NASA CR-122331

MARTIN MARIETTA AEROSPACE GROUP

DENVER DIVISION, POST OFFICE BOX 179, DENVER, COLORADO 80201

SPACE TRAJECTORIES ERROR ANALYSIS (STEAP) PROGRAMS

Volume I - Analytic Manual (Update)

December 1971

MARTIN MARIETTA CORPORATION
DENVER DIVISION
Denver, Colorado 80201

MCR-71-3

Volume I of Three Volumes

Final Report

Contract NAS 5-11795

Computer Program for Mission Analysis of Lunar and Interplanetary Missions

Prepared by

E. D. Vogt
G. L. Adams
M. M. Working
J. B. Ferguson

Space Navigation Technology Section
Martin Marietta Corporation
Denver, Colorado

For

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland

March, 1971

Acknowledgements

The authors wish to acknowledge the invaluable support of several persons whose help was instrumental in the completion of this contract. Dr. Al Bradt, serving as Program Manager, efficiently solved the many administrative problems that arose during the course of the contract. Joanne Spofford did her usual excellent job of typing the voluminous documents. Deborah Bower was responsible for all the graphics work involved with the production of the flowcharts and mathematical analyses. A utility computer program developed by Jim Schnelker aided greatly in the program conversion from CDC single precision to IBM double precision. All of their work was greatly appreciated.

FOREWORD

STEAP II is a series of three computer programs developed by the Martin Marietta Corporation for the mathematical analysis of interplanetary or lunar navigation and guidance. *STEAP* is an acronym for *Space Trajectory Error Analysis Programs*. The first series of programs under this name was developed under Contract NAS1-9745 for Langley Research Center and was documented in two volumes (*STEAP Users' Manual, STEAP Analytical Manual*) as NASA Contract Report 66818. Under contracts NAS5-11795 and NAS5-11873, the STEAP series was extensively modified and expanded for Goddard Space Flight Center. This second-generation series of programs is referred to as STEAP II.

STEAP II is composed of three independent yet related programs: NOMNAL, ERRAN, and SIMUL. All three programs require the integration on n-body trajectories for both interplanetary and lunar missions. The virtual-mass technique is the scheme used for this purpose in all three programs.

The first program named NOMNAL is responsible for the generation of n-body nominal trajectories (either lunar or interplanetary) performing a number of deterministic guidance events. These events include initial or injection targeting, midcourse retargeting, orbit insertion, and miniprobe targeting. A Variety of target parameters are available for the targeting events. The actual targeting is done iteratively either by a modified Newton-Raphson algorithm or by a steepest-descent/conjugate gradient scheme. Planar and nonplanar strategies are available for the orbit insertion computation. All maneuvers may be executed either by a simple impulsive model or by a pulsing sequence model.

ERRAN, the second program of STEAP II, is used to conduct linear error analysis and generalized covariance analysis studies along specific targeted trajectories. The targeted trajectory may, however, be altered during flight by retargeting events (computed either by linear or nonlinear guidance) and by an orbit insertion event. Knowledge and control covariances are propagated along the trajectory through a series of measurements and guidance events in a totally integrated fashion. The knowledge covariance is processed through measurements using a Kalman-Schmidt or equivalent recursive weighted-least-squares filter with arbitrary solve-for/consider augmentation. Execution of guidance events may be modeled either by an impulsive approximation or by a pulsing sequence model. The resulting knowledge and control covariances can be analyzed by the program at various events to determine statistical data, including

probabilistic midcourse correction sizing and effectiveness, probability of impact, and biased aimpoint requirements. Probe release events are also available for studying missions employing multi-probe spacecraft.

The third and final program in the STEEP II series is the simulation program SIMUL. SIMUL is responsible for the testing of the mathematical models used in the navigation and guidance process. An "actual" dynamic model is used to propagate an "actual" trajectory. Noisy measurements from this "actual" trajectory are then sent to the estimation algorithm. Here the actual measurement, the statistics associated with that measurement, and an "assumed" dynamical model are blended together to generate the filter estimate of the trajectory state. This process is repeated continually through the measurement schedule. At guidance events, corrections are computed based on the estimate of the current state. These corrections are then corrupted by execution errors and added to the "actual" trajectory. The statistics and augmentation of the filter, the mismatches in the "actual" and "assumed" dynamics, and the execution errors and measurement biases may then be varied to determine the effects of these parameters on the navigation and guidance process. All guidance and probe release event options defined for ERRAN are also available in SIMUL.

The documentation for STEAP II consists of three volumes: the Analytic, Programmers' and User's Manuals. Each of these documents is self-contained.

The *STEAP Analytic Manual* consists of two major divisions. The first section provides a unified treatment of the mathematical analysis of the STEAP II programs. The general problem description, formulation, and solution are given in a tutorial manner. The second section of this report supplies the detailed analysis of those subroutines of STEAP II dealing with technical tasks.

The *STEAP Programmers' Manual* provides the reader with the information he needs to modify the programs. Both the overall structure of the programs as well as the computational flow and analysis of the individual subroutines is described in this manual.

The *Users' Manual* contains the information necessary to operate the programs. The input and output quantities of the programs are described in detail. Example cases are also given and discussed.

CONTENTS

1. INTRODUCTION

This Analytic Manual is intended to provide the reader with the mathematical analysis, assumptions and restrictions upon which the STEAP II programs are based. The volume consists of three major divisions: an introductory section, an overview of the four basic modes of STEAP II, and a collection of detailed analyses of each of the technical subroutines of STEAP II.

The introductory section includes the Foreword summarizing the basic modes of STEAP II and the three formal documents describing STEAP II. This chapter describes the contents of the Analytic Manual. The following chapter defines the abbreviations and notation used throughout this document.

The second division of this report including chapters three through seven provides a convenient overview of the mathematical foundation of the four basic modes or subprograms of STEAP. Unified discussions of the virtual mass n-body propagation, the trajectory targeter NOMNAL, and the measurement processing and guidance modeling of the error analysis and simulation programs ERRAN and SIMUL are given. Therefore the reader desiring to know the general assumptions and procedures used by the major modes of STEAP II will find these chapters very helpful.

The third major division of this manual comprising the bulk of the volume is Chapter 8 which contains a detailed analysis of the STEAP II programs at the subroutine level. Each of the technical subroutines is analyzed individually and in detail. A cross reference of the subroutines by general categories is supplied. Thus the analyst who needs to make modifications or extensions to STEAP II may use this chapter to great advantage.

2. NOMENCLATURE

A. Arabic symbols

Symbol	Definition
a	Semi-major axis of conic
$B \cdot T$	Impact plane parameter
$B \cdot R$	Impact plane parameter
C_{xx_s}	Correlation between position/velocity state and solve-for parameters
C_{xu}	Correlation between position/velocity state and dynamic consider parameters
C_{xv}	Correlation between position/velocity state and measurement consider parameters
$C_{x_s u}$	Correlation between solve-for parameters and dynamic consider parameters
$C_{x_s v}$	Correlation between solve-for parameters and measurement consider parameters
e	Eccentricity of conic
E	Eccentric anomaly
f	True anomaly on conic
G	Observation matrix relating observables to dynamic consider parameter state
H	Observation matrix relating observables to position/velocity state
i	Inclination of conic (reference body equatorial)
J	Measurement residual covariance matrix
K	Kalman gain constant for position/velocity state
L	Observation matrix relating observables to measurement consider parameter state Mean longitude
M	Observation matrix relating observables to solve-for parameter state Mean anomaly

n_1	Dimension of solve-for parameter state
n_2	Dimension of dynamic consider parameter state
n_3	Dimension of measurement consider parameter state
p	Semilatus rectum of conic Probability density function
P	Position/velocity covariance matrix
$\hat{P}$	Unit vector to periapsis of conic
P_s	Solve-for parameter covariance matrix
Q	Dynamic noise covariance matrix
$\tilde{Q}$	Execution error matrix
$\hat{Q}$	Unit vector in plane of motion normal to P
r	Radius
r_{CA}	Radius of closest approach
r_{SI}	Radius of sphere of influence
R	Measurement noise covariance matrix
$\underline{R}$	Actual noise covariance matrix
$\hat{R}$	Unit vector normal to T in plane perpendicular to approach asymptote directed south ($R = S \times T$)
R_c	Target planet capture radius
S	Kalman gain constant for solve-for parameters
S_j	Velocity correction covariance matrix
$\hat{S}$	Approach or departure asymptote
t_{CA}	Time of closest approach to target body
t_{SI}	Time of intersection with sphere of influence of target body
Δt	Time interval
$\hat{T}$	Unit vector lying in ecliptic plane normal to $\hat{S}$. ($\hat{T} = \frac{\hat{S} \times \hat{K}}{\lvert \hat{S} \times \hat{K} \rvert}$ where $\hat{K}$ is unit normal to ecliptic plane.)
U_o	Dynamic consider parameter covariance matrix

v	Velocity
V_o	Measurement consider parameter covariance matrix
W_j	Target parameter covariance matrix
$\hat{W}$	Unit normal to orbital plane
X	Actual position/velocity state
$\bar{X}$	Targeted nominal position/velocity state
$\tilde{X}$	Most recent nominal position/velocity state

B. Greek Symbols

α	Auxiliary parameters
Γ_j	Guidance matrix
Γ	Flight path angle
δ	Declination of vector
Δv	Velocity increment
ϵ	Measurement residual Errors in target parameters
η_j	Variation matrix relating position/velocity variations to target conditions
θ_{xx_s}	State transition matrix partition associated with solve-for parameters
θ_{xu}	State transition matrix partition associated with dynamic consider parameters
θ	Longitude or right ascension
Λ_j	Projection of target condition covariance matrix W_j into the impact plane
μ	Gravitational constant of body
$\vec{\mu}$	Biased aimpoint
ν	Sampled measurement noise True anomaly

ρ	Magnitude of gaussian approximation for midcourse correction Correlation coefficient
σ	Standard deviation
Σ	Launch azimuth
$\vec{\tau}$	Target parameters
Φ	Targeting matrix State transition matrix for position/velocity state Latitude
χ	Sensitivity matrix
ψ_j	Matrix relating guidance corrections to target condition deviations
Ω	Longitude of ascending node
ω	Argument of periapsis
$\tilde{\omega}$	Longitude of periapsis

C. Subscripts

C	Control variable (P_C)
CA	Closest approach (r_{CA})
f	Final variable (t_f)
i	Initial variable (t_i)
j	Index of current guidance event (P_j)
k	Index of current measurement (P_k)
K	Knowledge variable (P_K)
s	Solve-for parameter (x_s)
SI	Sphere of influence (t_{SI})

D. Superscripts

A	Augmented variable (Φ^A)
T	Matrix transpose (Φ^T)
-1	Matrix inverse (Φ^{-1})

-	Variable immediately before instant (P_k^- or v^-)
+	Variable immediately after instant (P_k^+ or v^+)

E. Abbreviations

AU	Astronomical unit
CA	Closest approach to reference body
ERRAN	Error analysis program
FTA	Fixed time of arrival guidance policy
GHA	Greenwich hour angle
J.D.	Julian date (referenced either 0^{yr} or 1900^{yr})
km	Kilometers
M/C	Midcourse correction
NOMNAL	Nominal trajectory generation program
POI	Probability of impact
Q-L	Quasilinear filter event
S/C	Spacecraft
SF/C	Solve-for/consider
SIMUL	Simulation program
SOI	Sphere of influence
STM	State transition matrix
STEAP	Space Trajectories Error Analysis Programs
VM	Virtual Mass
2VBP	Two variable B-plane guidance policy
3VBP	Three variable B-plane guidance policy

3. TRAJECTORY PROPAGATION ANALYSIS

The trajectory mode of STEAP computes an n-body trajectory for an infinitesimal spacecraft through the use of the varicentric or virtual mass concept. As explained in detail by Novak in reference 15, the essential idea of virtual mass n-body trajectory computations is that, at any instant of time, the gravitational forces exerted by all the governing bodies can be resolved into one effective vector emanating from a virtual mass whose position and magnitude are uniquely determined. Over small time intervals, therefore, the motion of the spacecraft can be represented as a two-body conic section arc around the moving and varying virtual mass. The computational algorithm of the STEAP trajectory mode uses this concept in determining the n-body spacecraft trajectory.

Novak's original work proved the validity of the virtual mass approach for the restricted three-body problem. The trajectory mode of STEAP extends its applicability to general n-body problems. Modeled in the trajectory mode are the best available mean conic section orbital elements of each of the planets in the solar system plus the Earth's moon. These are available to the trajectory mode through an ephemeris subroutine and permit the determination of realistic interplanetary trajectories.

The basic concepts of virtual mass n-body trajectory computation are reviewed here for reference. In addition, the computational algorithm at each interval along the trajectory is presented, step by step, just as it appears in the trajectory mode of STEAP. For more details concerning the underlying concepts, see reference 15.

Consider the vector differential equations for the motion of an infinitesimal spacecraft under the influence of n attracting bodies to be given by:

$$\ddot{\vec{r}}_s = \sum_{i=1}^{n} \frac{\mu_i \, (\vec{r}_i - \vec{r}_s)}{|\vec{r}_i - \vec{r}_s|^3} \tag{3.1}$$

where $\vec{r}_s$ is the position vector of the spacecraft in some reference coordinate system, μ_i is the gravitational attraction of the i^{th} governing body, and $\vec{r}_i$ is the position vector of the i^{th} body in the same reference system. It is easy to show that an equivalent set of equations can be written as:

$$\ddot{\vec{r}}_s = \vec{M} - \vec{r}_s \, M_s = \frac{\mu_v \, (\vec{r}_v - \vec{r}_s)}{|\vec{r}_v - \vec{r}_s|^3} \tag{3.2}$$

where the quantities M, M_s, μ_v, and $\vec{r}_v$ are defined by:

$$\vec{M} = \sum_{i=1}^{n} \frac{\mu_i \vec{r}_i}{|\vec{r}_i - \vec{r}_s|^3} \tag{3.3}$$

$$M_s = \sum_{i=1}^{n} \frac{\mu_i}{|\vec{r}_i - \vec{r}_s|^3} \tag{3.4}$$

$$\vec{r}_v = \frac{\vec{M}}{M_s} \tag{3.5}$$

$$\mu_v = |\vec{r}_v - \vec{r}_s|^3 M_s \tag{3.6}$$

The final form of equation (3.2), which is easily recognized as the differential equation for two-body motion, suggests the essential idea of virtual mass computation for n-body trajectories. Over intervals where μ_v and $\vec{r}_v$ can be treated as constants, the motion is two-body with respect to this magnitude and position of the virtual mass. The computation of the n-body orbit in the trajectory mode results from piecing together two-body arcs around varying magnitudes and locations for the virtual mass. The way in which each two-body arc is calculated is discussed in the computational algorithm to follow.

A close inspection of the above equations demonstrates that the location and magnitude of the virtual mass has the desired limiting properties. When the spacecraft is within the influence of one dominant body, the virtual mass position and magnitude approximate those of the dominant body. In a transition region, two or more bodies may contribute significantly to the location and magnitude of the virtual mass.

The computational scheme used within the trajectory mode of STEAP is presented in the following paragraphs. Emphasis is placed on the procedure used for determining each individual two-body arc in the sequence. The decision concerning the length of the interval is made before entering the computational algorithm and is based on a fixed true anomaly passage with respect to the virtual mass. Thus, when the spacecraft is near the virtual mass (near a planet for interplanetary applications), smaller steps are taken to ensure the accuracy of the computation. Over the heliocentric portions of an interplanetary flight, when the spacecraft is far from the virtual mass location and the trajectory is essentially a heliocentric ellipse, larger computational intervals are automatically used.

Within each computing interval, the motion of the virtual mass is assumed to be constant velocity with a constant mass magnitude. Two approaches to determining this constant velocity and mass magnitude were analyzed by Novak (ref. 15). One is called the iterative method and the other noniterative. The noniterative computation uses the values for the

virtual mass velocity and mass magnitude at the beginning of the time step for the entire computation interval. Then, at the beginning of the new time step, new values are calculated and assumed consistent with the new position of the spacecraft. This method results in position discontinuities in the virtual mass trajectory since the initial values, rather than any computed mean values, are used over the step. The spacecraft trajectory itself is still continuous, but it is being based on a discontinuous virtual mass trajectory.

In the iterative method "average" values for the virtual mass velocity and mass magnitude are used over the computing interval. The values of $\vec{r}_v$ and μ_v at the end of the interval are initially estimated and then iteratively improved to force consistency between the virtual mass and spacecraft trajectories. The iterative method is used in STEAP; its computational algorithm is presented here, step by step.

1) At the beginning of the n^{th} time interval the acceleration terms from the previous time interval are calculated as

$$\ddot{\vec{r}}^{\,n}_{v_{av}} = \frac{\dot{\vec{r}}^{\,n-1}_{v_E} - \dot{\vec{r}}^{\,n-1}_{v_B} - \ddot{\vec{r}}^{\,n-1}_{v_B}\,(\Delta t^{n-1})}{(\Delta t^{n-1})^2}$$

$$\ddot{\mu}^{\,n}_{v_{av}} \quad \frac{\dot{\mu}^{\,n-1}_{v_E} \quad \dot{\mu}^{\,n-1}_{v_B} \quad \ddot{\mu}^{\,n-1}_{v_B}\,(\Delta t^{n-1})}{(\Delta t^{n-1})^2} \qquad (3.7)$$

At the first time interval $\ddot{\vec{r}}_{v_{av}} = 0$ and $\ddot{\mu}_{v_{av}} = 0$.

2) Assuming a second order variation with time, an initial guess for the final position and magnitude of the virtual mass is made by using the equations,

$$\vec{r}_{v_E} = \vec{r}_{v_B} + \dot{\vec{r}}_{v_B}\,(\Delta t) + \ddot{\vec{r}}_{v_{av}}\,(\Delta t)^2$$

$$\mu_{v_E} = \mu_{v_B} + \dot{\mu}_{v_B}\,(\Delta t) + \ddot{\mu}_{v_{av}}\,(\Delta t)^2 \qquad (3.8)$$

The superscripts n have been dropped for convenience.

All the succeeding equations are for the n^{th} time interval unless otherwise specified. The subscripts B and E refer to the beginning and end of the computational interval.

3) Using the assumed position and mass magnitude of the virtual mass at the end of the time interval, the assumed average velocity and mass magnitude for the virtual mass over the computational interval may be calculated as

$$\dot{\vec{r}}_{v_{av}} = \frac{\vec{r}_{v_E} - \vec{r}_{v_B}}{\Delta t}$$

$$\mu_{v_{av}} = C_1 \mu_{v_E} + (1 - C_1) \mu_{v_B} \tag{3.9}$$

where C_1 linearly interpolates the virtual mass magnitude to some value between the initial and final values $(0 \leq C_1 \leq 1)$.

The initial velocity of the spacecraft with respect to the virtual mass is now based on this assumed average velocity and is given by

$$\dot{\vec{r}}_{vs_B} = \dot{\vec{r}}_{s_B} - \dot{\vec{r}}_{v_{av}} \tag{3.10}$$

4) The Keplerian vector (represents twice the areal rate) for the computing interval is next computed as

$$\vec{k} = \vec{r}_{vs_B} \times \dot{\vec{r}}_{vs_B} \tag{3.11}$$

Then the eccentricity vector is determined from

$$\vec{e} = - \frac{\vec{r}_{vs_B}}{r_{vs_B}} - \frac{\vec{k} \times \vec{r}_{vs_B}}{\mu_{v_{av}}} \tag{3.12}$$

The magnitude of the eccentricity vector, $\vec{e}$, represents the eccentricity of the conic section and the orientation of the vector is toward the conic section periapsis.

5) The final position and velocity of the spacecraft with respect to the virtual mass is calculated next. An intermediate variable $\Delta\tau$ is used which must be related to the desired Δt for the interval. The value $\Delta\tau$ determines the time or true anomaly increment along the conic section arc. Again assuming a second order variation,

$$\Delta\tau = \Delta t + \kappa \Delta t^2 \tag{3.13}$$

where κ is computed from information about the preceding interval as

$$\kappa = \frac{\Delta\tau - \Delta t}{(\Delta t)^2} \tag{3.14}$$

The final position $\vec{r}_{vs_E}$ must lie in the plane of motion defined by $\vec{r}_{vs_B}$ and $\dot{\vec{r}}_{vs_B}$, and hence can be expressed as a linear combination of the two,

$$\vec{r}_{vs_E} = B\left[\vec{r}_{vs_B} + \Delta\tau\ \dot{\vec{r}}_{vs_B}\right] \equiv B\,\vec{\sigma}_{vs_E} \tag{3.15}$$

where the quantity B is given by,

$$B = \frac{k^2/\mu_{v_{av}}}{\vec{e}\cdot\vec{\sigma}_{vs_E} + \sigma_{vs_E}} \tag{3.16}$$

Then the velocity of the spacecraft with respect to the virtual mass at the end of the interval is

$$\dot{\vec{r}}_{vs_E} = \frac{\vec{k}\ x\left(\vec{e} + \dfrac{\vec{r}_{vs_E}}{r_{vs_E}}\right)}{\left(k^2/\mu_{v_{av}}\right)} \tag{3.17}$$

6) The final position and velocity of the spacecraft in the reference coordinate system are now computed from

$$\begin{aligned} \vec{r}_{s_E} &= \vec{r}_{vs_E} + \vec{r}_{v_E} \\ \dot{\vec{r}}_{s_E} &= \dot{\vec{r}}_{vs_E} + \dot{\vec{r}}_{v_{av}} \end{aligned} \tag{3.18}$$

7) It is necessary to evaluate the conic section time of flight so that κ may be found to use in the next iteration. First, some preliminary orbit variables must be determined. The in-plane normal to the major axis is

$$\vec{n} = \begin{cases} \dfrac{\vec{k}\ x\ \vec{e}}{k\ e}, & e \neq 0 \\ \dfrac{\vec{k}\ x\ \vec{r}_{vs_B}}{k\ r_{vs_B}}, & e = 0 \end{cases} \tag{3.19}$$

The length of the semimajor axis is given by

$$b = \frac{k^2}{\mu_{v_{av}} \left(|1 - e^2|\right)^{\frac{1}{2}}}, \quad e \neq 1$$

$$b_i = \frac{2}{r_{vs_i} - k^2/\mu_{v_{av}}}, \quad \begin{matrix} e = 1 \\ i = B,E \end{matrix} \tag{3.20}$$

The projection of the radius vector orthogonal to the major axis, divided by b is given by

$$x_i = \frac{\vec{n} \cdot \vec{r}_{vs_i}}{b_i}, \quad i = B,E \tag{3.21}$$

The mean angular rate is

$$\omega_M = \begin{cases} \dfrac{\mu_{v_{av}} (1 - e^2)}{k\, b}, & e \neq 1 \\ \dfrac{k}{2}, & e = 1 \end{cases} \tag{3.22}$$

where $\omega_M < 0$ for hyperbolic orbits. The eccentric anomaly is given by,

$$E_i = \begin{cases} \sin^{-1} X_i, & e < 1 \\ \dfrac{k^2/\mu_{v_{av}} \cdot X_i}{3}, & e = 1 \\ \sinh^{-1} X_i, & e > 1 \end{cases} \tag{3.23}$$

Then

$$M_i = E_i - eX_i, \quad i = B,E \tag{3.24}$$

and finally the conic section time of flight is given as

$$\Delta\tau = t_2 - t_1 = \frac{M_E - M_B}{\omega_M} \tag{3.25}$$

The intermediate variable κ, to be used in the next interval is calculated as

$$\kappa = \frac{\Delta\tau - \Delta t}{(\Delta t)^2} \tag{3.26}$$

8) The final positions and velocities of the planets are now calculated from the ephemeris subroutine and returned to the virtual mass routine.

9) The final position and mass magnitude of the virtual mass are now recalculated from the assumed position of the spacecraft at the end of the interval and the planetary aphemerides by using equation (2). The velocity and magnitude rate of the virtual mass are computed from

$$\dot{\vec{r}}_{v_E} = \frac{\dot{\vec{M}} - \vec{r}_{vs_E}\dot{M}_s}{M_s}$$

$$\dot{\mu}_v = \mu_{v_E}\left[\frac{V_{vs_E}}{r_{vs_E}} + \frac{\dot{M}_s}{M_s}\right] \tag{3.27}$$

where

$$\dot{\vec{M}} = \sum_{i=1}^{n} \frac{\mu_i}{r_{is_E}}\left[\dot{\vec{r}}_{i_E} - \vec{r}_{i_E}\left(\frac{V_{is_E}}{r_{is_E}}\right)\right]$$

$$M_s = \sum_{i=1}^{n} \frac{\mu_i}{\left(r_{is_E}\right)^3}\left(\frac{V_{is_E}}{r_{is_E}}\right)$$

$$\frac{V_{is_E}}{r_{is_E}} = \frac{3\,\vec{r}_{is_E}\cdot\dot{\vec{r}}_{is_E}}{\left(r_{is_E}\right)^2}$$

10) After this last computation, Step 9, one complete iteration has been obtained. The values of the final position and magnitude of the virtual mass that was just calculated is compared to the one assumed in the computation of the spacecraft trajectory. If they do not agree to within a set tolerance, the new values of r_{v_E},

μ_{V_E}, and k are returned to Step 3 and another iteration is performed. However, it should be pointed out that Step 8 is to be omitted from future iterations. The final positions of the planets do not differ from one iteration to the next since the final time is fixed.

11) After two iterations, the required quantities are stored for the next time interval and the algorithm returns to Step 1.

More complete details for the above computational algorithm may be found in Novak's report (ref. 15). One point worth mentioning at this juncture concerns the use of the words "accuracy level" when referring to an orbit computed by the trajectory mode. As was mentioned earlier, the step size used in the virtual mass calculation of n-body trajectories refers to the true anomaly arc, with respect to the virtual mass, that is kept fixed throughout the trajectory. Thus, a fixed true anomaly arc of 1 mrad means that each individual computing interval, using the algorithm defined above, results in a two-body arc around the effective force center of 1 mrad. If the fixed true anomaly arc is 10 mrad, then clearly fewer computational intervals are used and the resulting trajectory, neglecting computer noise, is less accurate.

An external accuracy level is input to the program, where this value is subsequently changed into a fixed true anomaly arc for the computing intervals. An external accuracy of 2.5×10^{-5}, for example, corresponds to a true anomaly arc of 16.57 mrad; similarly, an external accuracy level of 1×10^{-6} corresponds to a fixed true anomaly of 3.6 mrad. For the n-body problem the external accuracy level is a dummy variable; it was initially set up to represent the accumulated percentage position error for a restricted three-body problem after one orbit. Thus lower accuracy levels imply lower amounts of arc for fixed time anomaly used in the computations and, consequently, more accurate trajectories that require more computer time. Throughout this report, when referring to the computational interval size used by the trajectory mode or subroutine, the phase accuracy level is employed.

4. NOMNAL ANALYSIS

4.1 Introduction

The NOMNAL program is responsible for the generation of a nominal trajectory from injection through midcourse corrections to orbit insertion. The program structure of NOMNAL is described in the companion volume STEAP II Programmer's Manual. A detailed analysis of the main program NOMNAL and each of its supporting subroutines is provided in the last section of this report. This section provides a unified discussion of the general analysis, assumptions, and modeling upon which the NOMNAL program is based.

NOMNAL has been built with a modular computational structure. The basic cycle of NOMNAL consists of propagating an n-body trajectory using the virtual mass technique described in the previous chapter. Computational blocks external to this basic cycle are called events. The first type of event is called a zero iterate event in which the desired trajectory is approximated to yield initial conditions for the n-body trajectory. This includes both lunar and interplanetary missions.

The second type of event is a targeting event. At a targeting event the impulsive velocity correction required to meet specific target conditions is computed. A targeting event may be entered immediately following a zero iterate event or at any point along the nominal trajectory.

The third type of event is an orbit insertion event. In this event the impulsive velocity correction and time of execution required to insert into a desired orbit are computed.

The targeting and orbit insertion maneuvers may be specified as executable or nonexecutable. At a nonexecutable event the velocity correction is simply computed and recorded; at an executable event the velocity correction is actually added to the nominal trajectory. An execution event controls the execution of such maneuvers. The maneuvers are executed either by a simple impulsive model or by a more involved thrusting arc model.

The major analytic problems solved in NOMNAL may be conveniently divided into the following categories:

1. Interplanetary zero iterate
2. Lunar zero iterate
3. Targeting
4. Orbit Insertion
5. Thrusting Arc Modeling

These topics will be discussed in this order in this chapter.

4.2 Interplanetary Zero Iterate

Iterative refinement procedures used in targeting trajectories require a zero iterate value for the initial trajectory state. In the interplanetary case this zero iterate state is computed from a two stage procedure in which the first stage approximates the heliocentric phase by a massless planet trajectory and the second stage approximates the launch phase by a simple conic leg.

4.2.1 Heliocentric Phase

For interplanetary missions the initial and final position vectors and the time required to traverse them determines the general characteristics of the transfer trajectory (see HELIO). Four options are available in specifying the two terminals:

1. Planet to planet
2. Planet to specific point
3. Specific point to planet
4. Specific point to second point

When one of the terminals is a planet, its location at the relevant time (either initial time or target time) is computed to determine the position vector.

Now suppose that the initial and final points $\vec{r}_i$ and $\vec{r}_f$ have been determined either by being read in or computed internally and that the transit time Δt is available. Lambert's theorem states that the transit time between any two points on an ellipse is a function of the sun of the distances of each point from the focus, the distances between the points, and the semimajor axis of the ellipse, or

$$\Delta t = f(r_i + r_f, \ |\vec{r}_i - \vec{r}_f| \ , \ a) \qquad (4.1)$$

Since the only unknown in this equation is the semimajor axis a, it may be solved for iteratively (see FLITE). Battin [2] has shown that the eccentricity e is actually a function of the semimajor axis and so it may be determined simultaneously.

The orientation of the transfer plane is indicated in figure 4.1 below. The unit normal to the transfer plane is

$$\hat{W} = \frac{\vec{r}_i \times \vec{r}_f}{|\vec{r}_i \times \vec{r}_f|} \qquad (4.2)$$

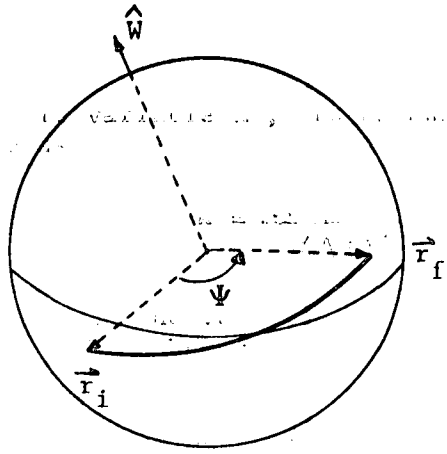


Figure 4.1. Transfer Trajectory Geometry

The inclination to that plane i is given by

$$\cos i = \hat{W}_z \tag{4.3}$$

The ascending node of the plane Ω may be computed from

$$\tan \Omega = \frac{\hat{W}_x}{-\hat{W}_y} \tag{4.4}$$

The central angle of the transfer Ψ is

$$\cos \Psi = \frac{\vec{r}_i \cdot \vec{r}_f}{r_i r_f} \tag{4.5}$$

The true anomaly at the initial and final points may be computed from

$$p = a(1 - e^2)$$

$$\cos f_f = \frac{p - r_f}{e\, r_f}$$

$$\cos f_i = \frac{p - r_i}{e\, r_i} \qquad \sin f_i = \frac{\cos f_i \cos \Psi - \cos f_f}{\sin \Psi}$$

$$f_f = f_i + \Psi \tag{4.6}$$

Finally the argument of periapsis is given by

$$\cos\ (\omega + f_i) = \frac{\vec{r}_i \cdot \hat{U}}{r_i} \tag{4.7}$$

where $\hat{U} = (\cos\Omega,\ \sin\Omega,\ 0)$ is the unit vector directed toward the ascending node.

Therefore, the elements defining the heliocentric conic $(a,\ e,\ i,\ \Omega,\ \omega)$ and the true anomaly at the initial and final times may all be computed from a knowledge of the initial and final radii vectors and the transfer time. Having these elements the position and velocity vectors and associated data at both the initial and final points may be determined.

4.2.2 The Launch Phase

The analysis to this point is called a massless planet solution, that is, the gravitational effects of the launch and target bodies have been ignored. In this section the effects of the launch body will be considered.

When the initial terminal is a planet, the transfer trajectory is automatically backed up to a realistic launch and injection phase (see LAUNCH). The heliocentric analysis has generated the velocity vector $\vec{v}_i$ on the transfer ellipse at the initial point. The hyperbolic excess velocity at the initial point is then given by

$$\vec{v}_{HE} = \vec{v}_i - \vec{v}_{LP} \tag{4.8}$$

where $\vec{v}_{LP}$ is the heliocentric velocity of the launch planet at the initial time. Up to this point it has been tacitly assumed that all computations have been done in ecliptic coordinates. From this point on launch planet equatorial coordinates will be used.

The model used for the launch phase is based on a simple one body point mass model. The spacecraft is to be launched from the launch body into a circular parking orbit from which it is injected into the escape hyperbola consistent with the desired hyperbolic excess velocity. The parking orbit and transfer plane are to be coplanar.

Let the $\vec{v}_{HE}$ vector be rotated into launch body equatorial coordinates. Then let $\hat{S}$ denote the unit vector in this direction; $\hat{S}$ is then the departure

asymptote. If the launch site latitude Φ_L and the launch azimuth Σ_L are specified there are at most two planes satisfying the latitude and azimuth constraints and containing $\hat{S}$ as indicated in Figure 4.2.

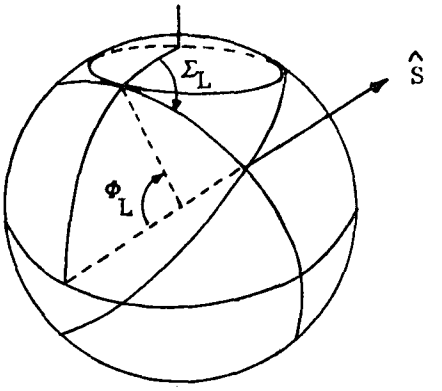


Figure 4.2 Launch Plane Determination

The determination of the transfer plane may be accomplished in the following way. The normal to the plane is given by

$$\hat{W} = \hat{r} \times \hat{v} = \hat{r} \times (\vec{v}_p + \vec{v}_r) = \hat{r} \times \hat{v}_p \qquad (4.9)$$

where $\hat{r}$, $\hat{v}$ are the launch radius and velocity vectors in equatorial coordinates. Since $\hat{r} \times \hat{r} = 0$ only the component of velocity normal to $\hat{r}$ need be considered in W. Suppose that launch occurs at a latitude of Φ, longitude of θ and azimuth Σ. Then the position and velocity terms may be written

$$\hat{r} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\Phi \\ \sin\Phi \end{bmatrix} \qquad \hat{v}_p = \begin{bmatrix} -\sin\theta\sin\Sigma - \sin\phi\cos\theta\cos\Sigma \\ \cos\theta\sin\Sigma - \sin\phi\sin\theta\cos\Sigma \\ \cos\phi\cos\Sigma \end{bmatrix}$$

The z-component of $\hat{W}$ may then be written

$$W_z = r_x v_y - r_y v_x = \cos\Phi \sin\Sigma \qquad (4.10)$$

To compute the remaining two components of $\hat{W}$ the conditions $\hat{W} \cdot \hat{S} = 0$, $\hat{W} \cdot \hat{W} = 1$ may be applied to yield

$$W_y = \frac{-W_z S_y S_z \pm S_x \sqrt{1 - (S_z^2 + W_z^2)}}{S_x^2 + S_y^2} \tag{4.11}$$

$$W_x = \frac{W_y S_y + W_z S_z}{S_x} \tag{4.12}$$

The y-component equation illustrates the necessity of a constraint on the launch azimuth :

$$\sin^2 \Sigma_L \leq \frac{\cos^2 \Phi_S}{\cos^2 \Phi_L} \tag{4.13}$$

where Φ_S is the declination of the departure asymptote.

The ambiquity in sign in the y-component distinguishes between the long and short coast time orbits. The positive sign designates the long coast solution; the negative sign, the short coast. The user specifies which of the two solutions he wants by input.

Having determined the plane of motion the in-plane characteristics must be computed. The energy C_3 and eccentricity of the departure hyperbola are given by

$$C_3 = v_{HE}^2 \tag{4.14}$$

$$e = 1 + \frac{r_p C_3}{\mu} \tag{4.15}$$

where r_p is the input desired parking orbit radius and μ is the launch planet gravitational constant. The orientation within the plane is specified by the argument of periapsis ω which may be determined from the true anomaly f_s of the departure asymptote $\hat{S}$ and the eccentricity of the hyperbola (4.15). The true anomaly at injection f_I is input by the user. Thus, the injection state may easily be computed from the standard conic formula.

To determine the required time of injection a realistic launch profile is now imposed. The size, shape, and orientation of the hyperbola within the launch plane has been determined above. The launch to injection phase is divided into three sections: a first burn arc from launch to parking orbit specified by a central angle Ψ_1 and a time interval Δt_1, a coasting arc in the parking orbit specified by k_ϕ (the inverse parking rate), and a second burn arc from parking orbit to injection specified by the central angle Ψ_2 and time interval Δt_2. The geometry is illustrated in Figure 4.3 below.

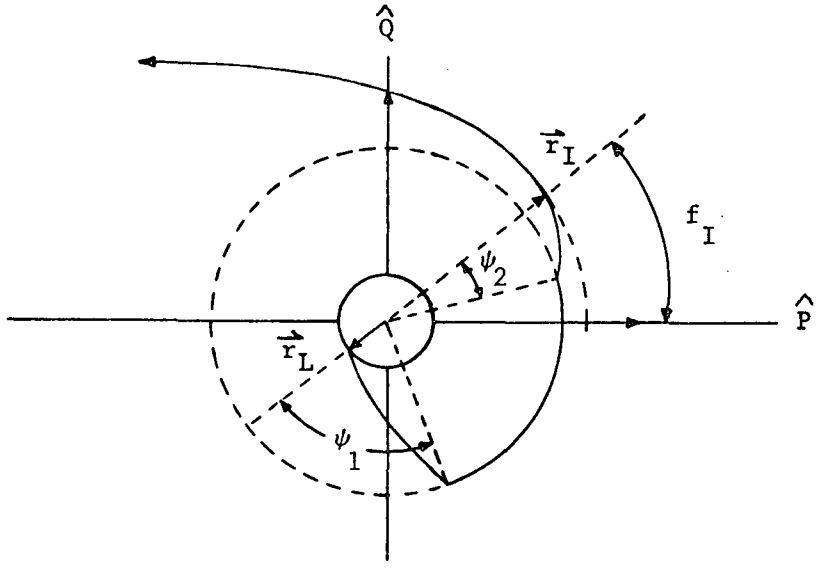


Figure 4.3 Launch Profile

The right ascension at launch Θ_L is determined from the fact that the launch plane satisfies the latitude, azimuth, and $\hat{S}$ constraints

$$\cos \Theta_L = \frac{W_x \sin\Phi_L \sin \Sigma_L + W_y \cos \Sigma_L}{W_z^2 - 1}$$

$$\sin \Theta_L = \frac{W_y \sin\Phi_L \sin \Sigma_L - W_x \cos \Sigma_L}{W_z^2 - 1} \qquad (4.16)$$

and the unit vector toward the launch site $\hat{r}_L$ is given by

$$\hat{r}_L = (\cos\Phi_L \cos\Theta_L,\ \cos\Phi_L \sin\Theta_L,\ \sin\Phi_L)^T \qquad (4.17)$$

Since $\hat{r}_I$ is also known the computations necessary to compute the time interval Δt_{LI} from launch to injection are straightforward and may be found in detail in LAUNCH. The time at which launch must occur coincides with the time that the launch site passes through the launch plane which is equivalent to the time at which the launch site latitude and launch right ascension are consistent. This may be computed from

$$t_L = \frac{(\Theta_L - \theta_L - GHA) \bmod 2\pi}{\omega} \qquad (4.18)$$

where ω is the rotation rate of the planet and where GHA is the Greenwich hour angle at 0^h U.T. on the launch date. The time of injection t_I is then given by

$$t_I = t_L + \Delta t_{LI} \qquad (4.19)$$

4.3 Lunar Zero Iterate

The generation of a lunar zero iterate proceeds in two steps. The first step involves the targeting of a patched conic trajectory; the second then targets a multi-conic trajectory to the desired conditions. The controlling subroutine for the lunar targeting is LUNA.

4.3.1 Patched Conic Targeting

The lunar patched conic phase (see LUNCON, LUNTAR) generates a patched conic trajectory which satisfies both launch conditions at earth and target conditions at closest approach to the moon. The launch conditions include launch from a site specified by latitude Φ_L and longitude θ_L at an azimuth Σ_L into an intermediate parking orbit of radius r_p until injection. The target conditions include time at closest approach t_{CA}, radius at closest approach r_{CA}, inclination to the lunar equator i_{CA}, and semimajor axis at the lunar hyperbola a_{CA}.

For the patched conic model, the moon's position is fixed at its location at the time the sphere of influence is pierced. The eccentricity of the lunar phase hyperbola is

$$e_{CA} = 1 - \frac{r_{CA}}{a_{CA}} \tag{4.20}$$

where $a_{CA} < 0$. The hyperbolic time Δt to go from R_{SI} (the radius of the lunar sphere of influence) to periapsis is computed from

$$\Delta t = f(\mu_M, a_{CA}, e_{CA}, R_{SI}) \tag{4.21}$$

where μ_M is the lunar gravitational constant. Thus, the time at which the probe should intersect the SOI is

$$t_{SI} = t_{CA} - \Delta t \tag{4.22}$$

Now any point on the lunar SOI can be described by giving two angular components: declination δ and right ascension θ for example. Denote the vector from the center of the earth to such a point on the SOI by $\vec{R}_I$. Then by setting

$$\hat{S} = \frac{\vec{R}_I}{R_I} \tag{4.23}$$

the plane including the vector $\hat{S}$ and satisfying the launch site latitude and azimuth conditions may be determined as it was in the previous section for the interplanetary case. The normal $\hat{W}$ to the transfer plane (identical to the parking orbit plane) is thus given by

$$\begin{aligned} W_z &= \cos\Phi_L \sin\Sigma_L \\ W_y &= \frac{-W_z S_y S_z \pm S_x \sqrt{1 - (S_z^2 + W_z^2)}}{S_x^2 + S_y^2} \\ W_x &= -\frac{W_y S_y + W_z S_z}{S_x} \end{aligned} \tag{4.24}$$

Now let the injection point on the parking orbit of radius r_p be located an angle α from the vector $-\vec{R}_I$. Denote the radius vector to that injection point by $\vec{r}_I$. Then the situation is illustrated in Figure 4.4 below.

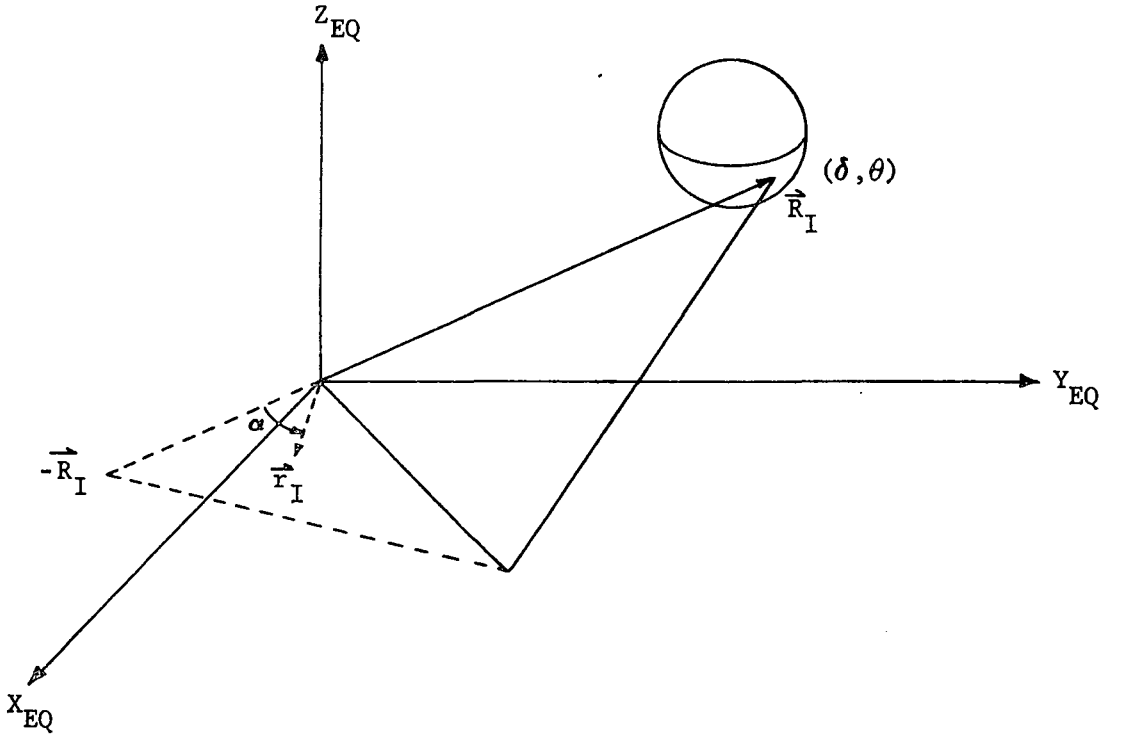


Figure 4.4 Lunar Patched Conic Targeting

The three controls α, δ, θ determine a unique patched conic trajectory (see LUNCON). The elements of the geocentric phase may be computed from the simultaneous solution of the equations

$$R_I = \frac{a(1 - e^2)}{1 + e \cos(\pi - \alpha)}$$

$$r_I = a(1 - e) \tag{4.25}$$

to yield the semimajor axis and eccentricity of the geocentric conic

$$e = \frac{R_I - r_I}{r_I - R_I \cos(\pi - \alpha)} \tag{4.26}$$

$$a = \frac{r_I}{1 - e} \tag{4.27}$$

The position and velocity $\vec{R}_I$, $\vec{V}_I$ on the geocentric conic may then be evaluated at the lunar SOI. The state relative to the moon at this patching point is given by

$$\vec{r}_{SI} = \vec{R}_I - \vec{R}_M \tag{4.28}$$

$$\vec{V}_{SI} = \vec{V}_I - \vec{V}_M \tag{4.29}$$

where $\vec{R}_M$, $\vec{V}_M$ are the position and velocity vectors of the moon with respect to the earth.

Given the state relative to the moon the target parameters r_{CA}, i_{CA}, and a_{CA} may be evaluated from the usual conic formulae. However, for linearity purposes, the impact plane parameters B·T and B·R are substituted for the parameters r_{CA} and i_{CA}.

The actual targeting procedure (see LUNTAR) may now be described. Suppose that on the k-th iterate the best values of the controls are denoted by $(\alpha_k, \delta_k, \theta_k)$. Suppose that the resulting target values are given by $(a_k, B{\cdot}T_k, B{\cdot}R_k)$. The Newton-Raphson scheme is then used to generate the k+1 st values for the controls.

The first control α_k is first perturbed by the amount $\Delta\alpha$ while holding the other controls at the k iterate values and the resulting trajectory

determined. Suppose that the resulting target values differ from the k-th iterate values by the amounts $(\Delta a_\alpha, \Delta B\cdot T_\alpha, \Delta B\cdot R_\alpha)$. Then the δ and θ controls are each perturbed in the same manner and the resulting perturbations in the target parameters denoted by $(\Delta a_\delta, \Delta B\cdot T_\delta, \Delta B\cdot R_\delta)$ and $(\Delta a_\theta, \Delta B\cdot T_\theta, \Delta B\cdot R_\theta)$ respectively. The "sensitivity" matrix is then defined to be

$$\Phi = \begin{bmatrix} \frac{\Delta a_\alpha}{\Delta \alpha} & \frac{\Delta a_\delta}{\Delta \delta} & \frac{\Delta a_\theta}{\Delta \theta} \\ \frac{\Delta B\cdot T_\alpha}{\Delta \alpha} & \frac{\Delta B\cdot T_\delta}{\Delta \delta} & \frac{\Delta B\cdot T}{\Delta \theta} \\ \frac{\Delta B\cdot R_\alpha}{\Delta \alpha} & \frac{\Delta B\cdot R}{\Delta \delta} & \frac{\Delta B\cdot R_\theta}{\Delta \theta} \end{bmatrix} \qquad (4.30)$$

The targeting matrix is then defined to be Φ^{-1}. The values of the controls to be used on the next iteration are then given by

$$\begin{bmatrix} \alpha \\ \delta \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} \alpha \\ \delta \\ \theta \end{bmatrix}_k + \Phi^{-1} \begin{bmatrix} a_{CA} - a_k \\ B\cdot T - B\cdot T_k \\ B\cdot R - B\cdot R_k \end{bmatrix} \qquad (4.31)$$

where B·T and B·R are the values of the impact parameters corresponding to the desired values i_{CA} and r_{CA}. (IMPACT)

When the values of the target parameters are within given tolerances of the desired values, the patched conic phase of the targeting is terminated and the multiconic stage begun. The time of injection of the final targeted patched conic is computed from conic formula to use in that second stage.

4.3.2 Multiconic Targeting

The second phase of the generation of a lunar zero iterate involves the targeting of a multi-conic trajectory (see MULTAR, MULCON).

Multi-Conic Trajectory Propagation

The multi-conic propagation scheme has recently been introduced as an effective intermediate betwen patched conic and precision integrated trajectories (Ref. 4).

A detailed analysis of the propagation of a trajectory by the multi-conic technique is provided in the analysis of subroutine MULCON. A heuristic description of this scheme will be discussed in this section.

The equations defining the motion of a spacecraft traveling under the influence of the earth and moon may be written

$$\ddot{\vec{r}}_e = -\frac{\mu_E \vec{r}_E}{r_E^3} - \frac{\mu_M \vec{r}_M}{r_M^3} - \frac{\mu_M \vec{R}_{EM}}{R_{EM}^3} \qquad (4.32)$$

where $\vec{r}_E$, $\vec{r}_M$, and $\vec{R}_{EM}$ are the position vectors of the spacecraft relative to Earth, the spacecraft relative to the moon, and the moon relative to the Earth and μ_E and μ_M are the gravitational constants of the Earth and moon respectively.

A graphic description of the multi-conic scheme is provided in Figure 4.5 below.

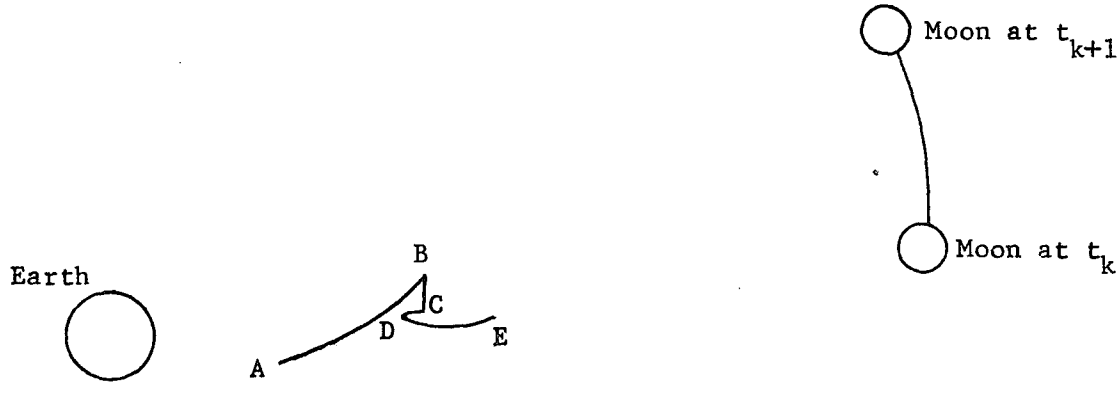


Figure 4.5 Schematic Representation of Multi-conic Propagation

The state of the spacecraft at the beginning of the n-th iterative step is indicated by the point A. The trajectory AB is a simple conic about the earth determined by the initial state at the time t_n and propagated over the time interval $\Delta t = t_{n+1} - t_n$. This corresponds to ignoring the last two terms in equation (4.32) above.

The second leg of the trajectory BC accounts for the "third term" or indirect force. Here the third term is evaluated at the beginning and at the end of the interval $[t_n, t_{n+1}]$. The net effect of the force is approximated by assuming that a constant force equal to the average of the two values of the indirect force acted over the interval. That perturbation is represented by the arc BC.

Finally, the effect of the direct lunar force is considered. The state of the spacecraft relative to the moon at the point C (corresponding to the time t_{n+1}) is first computed. This state is then propagated linearly backwards in time over the time interval Δt arriving at the point D. Then a simple forward conic propagation relative to the moon over the interval Δt generates the path DE. This may then be converted to geocentric coordinates to begin the next multi-conic step.

The multi-conic technique has been shown to efficiently approximate n-body trajectories to a very reasonable degree of accuracy. Therefore, an intermediate stage of targeting using this model was built into STEAP.

Multi-Conic Targeting

The actual targeting of the multi-conic trajectory is controlled by the subroutine MULTAR. The targeting scheme uses Newton-Raphson iteration to do the targeting. Let the k-th iterate values of the injection state in earth ecliptic coordinates and time be denoted $\hat{r}_k$, $\hat{v}_k$, t_k respectively. The problem then becomes the generation of an improved injection state and time $\hat{r}_{k+1}$, $\hat{v}_{k+1}$, t_{k+1}.

The k-th value of injection state $(\hat{r}_k, \hat{v}_k, t_k)$ is propagated forward using the multi-conic propagator to determine a final state near the moon. The resulting values of $B \cdot T_k$, $B \cdot R_k$, a_k, and $t_{CA,\,k}$ achieved on the trajectory as well as the target values B^*T and $B^*.R$ (consistent with the desired values of r_{CA} and i_{CA}) are computed. The errors in the four target conditions are then computed

$$\Delta\tau_k = \begin{bmatrix} \Delta a \\ \Delta B\cdot T \\ \Delta B\cdot R \\ \Delta t_{CA} \end{bmatrix} = \begin{bmatrix} a_k - a_{CA} \\ B\cdot T_k - B^*\cdot T \\ B\cdot R_k - B^*\cdot R \\ t_{CA,k} - t_{CA} \end{bmatrix} \tag{4.34}$$

If the error in each parameter is less than the allowable tolerance, the process is terminated. If convergence has not been achieved a Newton-Raphson iteration is made. The control variables are now

$$V_x,\ V_y,\ V_z,\ t_{inj}$$

and the target parameters are

$$a_{CA},\ B\cdot T,\ B\cdot R,\ t_{CA}$$

A complicating factor is introduced because of the time variable t_{inj}. The result of the patched conic targeting is a state $\vec{r}_o$, $\vec{v}_o$ in earth ecliptic coordinates at the time t_o. However, since the earth is rotating about the earth-moon barycenter, when the injection time is varied the injection state (both position and velocity) must be rotated to insure that an otherwise equivalent state is being used.

Thus, a sensitivity matrix Φ is computed by numerical differencing

$$\Phi = \begin{bmatrix} \dfrac{\Delta a_{CA_x}}{\Delta v_x} & \dfrac{\Delta a_{CA_y}}{\Delta v_y} & \cdots \\ \dfrac{\Delta B\cdot T_x}{\Delta v_x} & \vdots & \cdots \\ \dfrac{\Delta B\cdot R_x}{\Delta v_x} & & \\ \dfrac{\Delta t_{CA_x}}{\Delta v_x} & & \end{bmatrix} \tag{4.35}$$

The k+1 st iterate is then corrected by

$$\begin{bmatrix} \Delta V_x \\ \Delta V_y \\ \Delta V_z \\ \Delta t_{inj} \end{bmatrix}_{k+1} = \Phi^{-1} \Delta r_k \qquad (4.36)$$

The Newton-Raphson process is continued until the errors Δr_k are all within the desired tolerances.

4.4 N-Body Targeting

4.4.1 Parameters

N-Body targeting events may be required at the start of a trajectory or at some interior point along a trajectory. In either case the current position vector and time (perhaps generated as a zero iterate) are held at their original values while the velocity components are varied to meet three target conditions. The target parameters may be chosen from the list supplied in table 4.1 below.

Table 4.1 Target Parameters and Codes

1.	(available)	5.	B·T	9.	a_{SI}
2.	t_{SI}	6.	B·R	10.	x_f
3.	t_{CS}	7.	r_{CA}	11.	y_f
4.	t_{CA}	8.	i	12.	z_f

The parameter t_{SI} prescribes the time at which the target body SOI is intersected. The parameters t_{CS} and t_{CA} refer to time at closest approach to the target body: t_{CS} indicates that the integration is stopped at the SOI and the time at closest approach extrapolated from conic formula

while t_{CA} indicates that the time at closest approach is determined by actually integrating to closest approach. In lunar targeting a time variable may be replaced by the semimajor axis a_{SI} at the lunar SOI. In this case the n-body integration is stopped at the SOI for the evaluation of the target parameters. These four parameters thus specify the stopping conditions of the integration then. If one of the parameters t_{SI}, t_{CS}, or a_{SI} is triggered, the integration is stopped at the target body SOI and all target parameters are evaluated at that point using conic formulae if necessary. If t_{CA} is a target parameter, the integration is stopped at closest approach to the target body for the computation of parameters. If none of these four parameters are triggered, the integration stops at the target time.

The other seven parameters are automatically evaluated at the appropriate stopping condition. B·T and B·R are the impact plane parameters (see IMPACT), r_{CA} is the radius at closest approach to the target body, i is the inclination with respect to the target body equatorial system (see IMPACT) and x_f, y_f, z_f are the inertial ecliptic coordinates at the stopping conditions.

4.4.2 General Targeting Procedure

For efficiency it is sometimes necessary to introduce auxiliary parameters in place of the selected target parameters within the actual targeting procedure. Thus, if r_{CA} and i are specified as target parameters, auxiliary parameters of B·T and B·R are substituted for them. The reason for this is that the impact plane parameters are more linear functions of the velocity components than are r_{CA} and i. Therefore, the terms target parameters and auxiliary parameters are used. The target parameters $\vec{\tau}$ are those parameters for which the user has actually requested desired values; the auxiliary parameters $\vec{\alpha}$ are those parameters which the program uses to perform the targeting. Thus $\vec{\tau} = \vec{\alpha}$ unless i and r_{CA} are target parameters. In that case α_i = B·T if $\tau_i = r_{CA}$ and α_j = B·R if τ_j = i.

The general targeting program TARGET controls the refinement of the velocity components to meet the desired conditions. TARGET uses either of two iterative processes to perform the targeting: either the Newton-Raphson scheme or a steepest descent/conjugate gradient algorithm.

In either case suppose that on the k-th iteration the values of the velocity components are denoted $\hat{v}_k$. Denote the resulting values of the target and auxiliary parameters by r_k and α_k respectively. If the target values r_k agree with the desired values $\vec{r}^*$ to within the input tolerances the process is terminated. Otherwise the difference between the current auxiliary parameters and their desired values $\vec{\alpha}_k^*$ are computed

$$\vec{\Delta\alpha} = \vec{\alpha}_k - \vec{\alpha}_k^* \tag{4.37}$$

TARGET then calls on TARMAX or DESENT to compute the velocity correction $\Delta\hat{v}$ to be added to obtain the k+1 st iterate values by either the Newton-Raphson or steepest descent/conjugate gradient techniques discussed in the next two sections. In either case assume the correction is generated

$$\Delta\hat{v} = \hat{v}_{k+1} - \hat{v}_k = f\ (\Delta\alpha) \tag{4.38}$$

Two checks are available to guard against divergence in the targeting procedure: the maximum step check and the bad step check. In the maximum step scheme a maximum allowable velocity correction magnitude Δv_M is read in as input. If the correction determined by (4.38) has a magnitude larger than Δv_M, all components are scaled down to yield a correction within the maximum size. Therefore, if one wants to play safe he can read in a Δv_M that is perhaps one hundred times larger than the perturbation size dv used is predicting the step. The program will then insure that no steps larger than a hundred times the perturbation size will be allowed. Thus, the correction may be forced to stay within a region where the linear assumptions are hopefully valid.

The second scheme might be called an "a posteriori" method in comparison with the "a priori" method listed above. In the bad step check a scalar error or loss function ϵ is assigned to each iterate as

$$\epsilon_k = \vec{W} \cdot \vec{\Delta\alpha} \tag{4.39}$$

where $\vec{W}$ is a vector of weights. Whenever a step leads to a scalar error larger than the previous step, the current correction is reduced by one quarter recursively until a step with a smaller error is determined. Thus large steps that reduce the error will be allowed; only when they increase the error will they be decreased.

4.4.3 Newton-Raphson Iteration

In the Newton-Raphson technique, the errors in the auxiliary parameters are iteratively driven to zero by varying a set of three velocity controls, C_1, C_2, and C_3. Two different sets of these controls are available. In the first option, C_i is simply the increment to the ith Cartesian component of heliocentric ecliptic inertial velocity provided by the targeting impulse. The dependence of the targeting velocity increment $\underline{\Delta v}$ on these controls is then clearly

$$\underline{\Delta v} = C_1 \underline{i} + C_2 \underline{j} + C_3 \underline{k} \tag{4.40}$$

where $\underline{i}$, $\underline{j}$, and $\underline{k}$ are the respective unit vectors along the ecliptic inertial coordinate axes. The second control option is somewhat more complicated. It deals with the velocity relative to the launch planet in a rotating spherical coordinate system referenced to the trajectory. The current velocity vector determines the zero-latitude zero-longitude direction and the current angular momentum vector of the trajectory relative to the launch planet defines the +z or polar direction. The three controls then are respectively the increase in length, the latitude, and the longitude of the new velocity relative to the launch planet after addition of the targeting impulse. Figure 4.5a defines the controls pictorially for the case of the earth as launch planet.

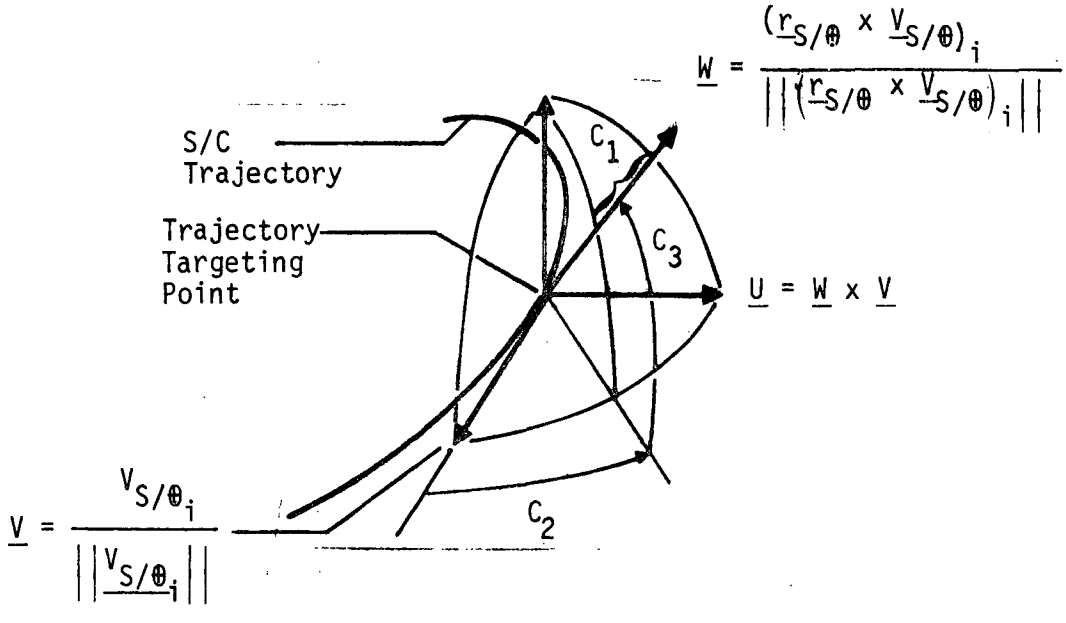


Figure 4.5a Pictorial Definition of Launch-Planetocentric Targeting Controls

The dependence of the targeting velocity increment on these controls is

$$\underline{\Delta v} = [(||\underline{v}|| + C_1) \cos C_2 \cos C_3 - ||\underline{v}||]\, \underline{V} + (||\underline{v}|| + C_1) \sin C_2 \cos C_3\, \underline{U} + (||\underline{v}|| + C_1) \sin C_3\, \underline{W} \qquad (4.41)$$

where $\underline{V}$ is a unit vector in the direction of the launch-planetocentric velocity just prior to the targeting impulse, $\underline{W}$ is a unit vector in the direction of the angular momentum vector of the launch-planetocentric trajectory and $\underline{U} = \underline{W} \times \underline{V}$.

Using the dependence of $\underline{\Delta v}$ on the particular set of controls at hand, a sensitivity (Jacobian) matrix Φ of the auxiliary parameters with respect to the controls is approximated by numerical differencing. The ijth element of Φ is by definition

$$\Phi_{ij} = \frac{\partial \alpha_i}{\partial C_j} \qquad \begin{matrix} i = 1, 2, 3 \\ j = 1, 2, 3 \end{matrix} \qquad (4.42a)$$

Now partitioning Φ into columns as follows

$$\Phi = \left(\Phi_1 \vdots \Phi_2 \vdots \Phi_3\right) \qquad (4.42b)$$

the jth column is given by

$$\Phi_j = \lim_{C_j \to 0} \frac{\Delta a}{C_j} \qquad (4.42c)$$

Thus the columns of the sensitivity matrix can be approximated by successively perturbing the three velocity controls. The control correction for the kth iteration is then given by the standard Newton-Raphson formula

$$\underline{\Delta C}_k = -\Phi_k^{-1}\, \underline{\Delta a}_k \qquad (4.43)$$

where $\underline{\Delta a}$ is the error vector of the auxiliary parameters on the previous iterate from their desired values.

The Newton-Raphson scheme may be slightly modified to yield a more efficient targeting algorithm. The modification is based on a premise that the characteristics of a low-integration-accuracy trajectory mirror those of a higher accuracy trajectory targeted to identical mission constraints. Thus, a targeting matrix computed about a given trajectory at a low-accuracy level should remain valid for similar trajectories at higher accuracy levels. This assumption has been verified by experimentation.

The modified targeting scheme then proceeds as follows. A trajectory is targeted to the desired target conditions at a low accuracy level, constructing the targeting matrix at each iteration. The targeting matrix Φ_L evaluated about the targeted trajectory at this low accuracy level is stored. The corresponding targeted velocity at this low level $\vec{v}_L$ is then used as the first iterate at a slightly higher accuracy. Because of the change in the integration step size the desired target conditions will not be realized. However, the matrix Φ_L may be used to predict an improved velocity $\vec{v}_I$ for that intermediate level. This process is continued until the desired accuracy level is reached. The the targeting matrix Φ_L is used repeatedly until a velocity is determined which yields a trajectory satisfying the desired target conditions.

4.4.4 Steepest Descent/Conjugate Gradient Iteration

An alternate scheme is provided for the generation of the next iterate by use of steepest descent/conjugate gradient method (DESENT). The current gradient $\vec{g}_c$ is computed by numerical differencing. For the x-component of $\vec{g}_c$ the corresponding component of velocity is perturbed by dv

$$\vec{v}_x = \vec{v}_k + [dv,\ 0,\ 0]^T \tag{4.44}$$

The initial state $(\vec{r},\ \vec{v}_x)$ is then propagated to the final stopping conditions. Let the auxiliary parameters of that trajectory be denoted α_x. The error (or loss function) associated with the perturbed state is then

$$\epsilon_x = \vec{W} \cdot (\vec{\alpha}_x - \vec{\alpha}^*) \tag{4.45}$$

where $\vec{W}$ represents the weighting factors and $\vec{\alpha}^*$ are the desired target conditions. The x-component of the current gradient is then

$$g_{c_x} = \frac{\epsilon_x - \epsilon}{dv} \tag{4.46}$$

The y- and z-components of the gradient are computed similarly by perturbing the y- and z-components respectively. The corrected gradient is then given by

$$\vec{p}_c = \vec{g}_c \quad , \text{ steepest descent step}$$

$$= \frac{|g_c|^2}{|g_p|^2} \vec{p}_p + \vec{g}_c \quad , \text{ conjugate gradient step} \qquad (4.47)$$

where the subscript c refers to a current parameter, p refers to a previous step. The unit vector in the direction of the next step is then given by

$$\vec{q}_c = - \frac{\vec{p}_c}{p_c} \qquad (4.48)$$

The step size is now determined as the correction leading to the minimization of the loss function in the direction predicted by (4.48). The directional derivative of the scalar error in the direction $\vec{q}_c$ is

$$d = \vec{q}_c \cdot \vec{g}_c \qquad (4.49)$$

The nominal optimal step size $\bar{h}$ is computed from a linear approximation to null the error

$$\bar{h} = \frac{\epsilon}{-d} \qquad (4.50)$$

The initial state corresponding to this correction is then propagated to the final stopping conditions and the resulting error ϵ computed. The three conditions

$$\begin{aligned} y(0) &= \epsilon \\ y(\bar{h}) &= \bar{\epsilon} \\ y'(0) &= d \end{aligned} \qquad (4.51)$$

may now be applied to the formula of a parabola

$$y - \epsilon^* = a(x - h^*)^2 \qquad (4.52)$$

to predict the optimal step size h^* yielding the minimum error ϵ^*

$$h^* = \frac{d\bar{h}^2}{2(d\bar{h} + \epsilon - \bar{\epsilon})} \tag{4.53}$$

The correction for the current step is then given by

$$\Delta\vec{v} = h^* \bar{q}_c \tag{4.54}$$

4.4.5 Outer Targeting

All of the targeting problems that involve stopping conditions at the SOI or at closest approach to the target body require a trajectory that at least intersects the target body SOI.

Since the massless planet initial conditions will not always satisfy this requirement some provision must be made to refine those conditions to obtain acceptable ones. An effective algorithm has been constructed to accomplish this.

When any trajectory has not encountered the target body SOI within a prescribed time, the closest approach conditions $\vec{r}_{CA}$, $\vec{v}_{CA}$ are noted (see Figure 4.6). An "artificial" sphere of influence is then constructed about the target body having a radius 1.2 times the value r_{CA} just noted. The trajectory will then intersect this artificial SOI even with small perturbations in the initial velocity. The normal targeting procedure is now used with target values

$$B \cdot T_A = B \cdot R_A = 0$$

$$t_{SI_A} = t_{SI} - \frac{1.2\, r_{CA} - R_{SI}}{V_{CA}} \tag{4.55}$$

where the subscript A indicates "artificial" target conditions and R_{SI} indicates the actual SOI radius. The initial conditions consistent with these "artificial" conditions will then yield a trajectory headed straight for the center of the target body when the artificial SOI is intersected which should result in a trajectory hitting the actual SOI.

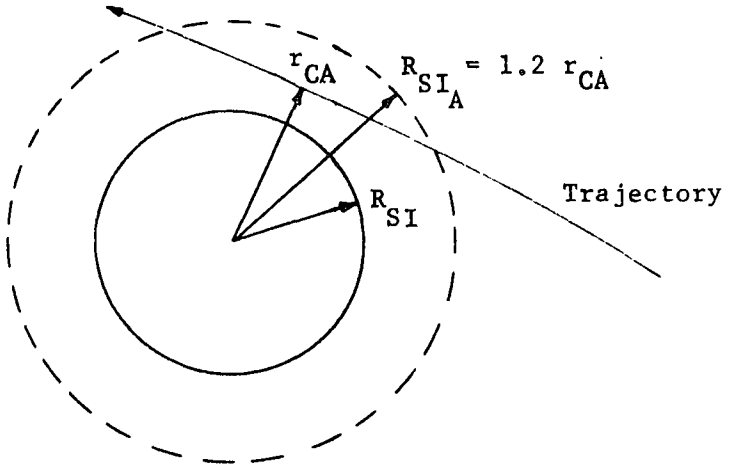


Figure 4.6 Outer Targeting Scheme

4.5 Orbit Insertion

4.5.1 Orbit Insertion Structure

The second type of guidance maneuver is an orbit insertion event. The orbit insertion event is divided into two distinct subevents: an orbit insertion decision event and an orbit insertion execution event occurring at a later time. There are two possible strategies available for the orbit insertion decision event: one specifying that the desired orbit is automatically coplanar with the incoming hyperbola and one specifying the orientation of the desired final orbit.

The orbit insertion decision event occurs at a time specified by the user (generally when the spacecraft is well within the target body SOI.) The general procedure begins after exiting from the trajectory propagation cycle. The state of the spacecraft with respect to the target body is computed and the conic extension of the trajectory is computed. The possible intersection point of this trajectory with the desired orbit are then investigated. If there are no points of intersection, a series of modifications are checked to determine an optimal modification which leads to a tangential or intersecting solution for the coplanar or nonplanar strategies respectively. The impulsive velocity correction corresponding to this solution is then computed along with the time interval (from conic formulae) from the time of the decision to the required time of execution.

After making these computations the trajectory propagation program is re-entered. If the insertion event is to be executed, an execution event is set up for the appropriate time before re-entering the trajectory mode. At that time the trajectory program is exited and the impulsive correction added on.

4.5.2 Coplanar Strategy

The details of the coplanar orbit insertion decision are available in COPINS. A heuristic description of this option will be given here. In the coplanar option the desired orbit is selected within the orbital plane of the approach hyperbola. The target parameter specified by the user include the elliptical semimajor axis a, eccentricity e, and periapsis shift $\Delta\omega$. The periapsis shift is defined as the angle from the periapsis of the approach hyperbola to the periapsis of the desired orbit measured positive in the direction of motion (see Figure 4.7).

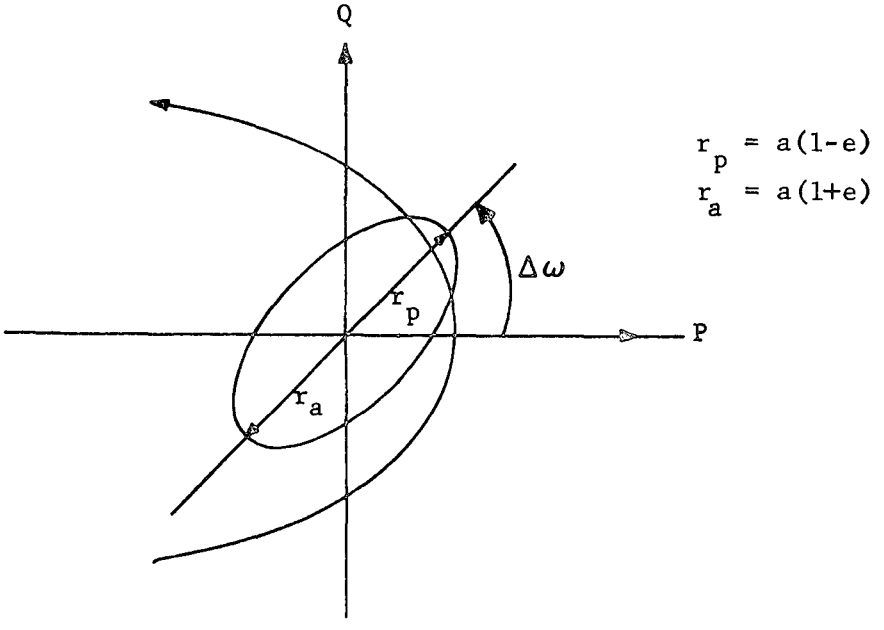


Figure 4.7 Definition of Coplanar Insertion Parameters

There are three possible situations which may arise in the determination of the points of intersection: there may be one, two, or no solutions. If there is one solution, that solution is analyzed to compute the impulsive Δv and time interval Δt before execution. If there are two solutions, the minimum velocity correction magnitude solution is computed for execution.

If there are no solutions the desired orbit is modified to obtain a tangential solution. Three modifications (see Figure 4.8) are investigated:

1) Vary r_p while holding r_a at desired value

2) Vary r_a while holding r_p at desired value

3) Vary a while holding e at desired value

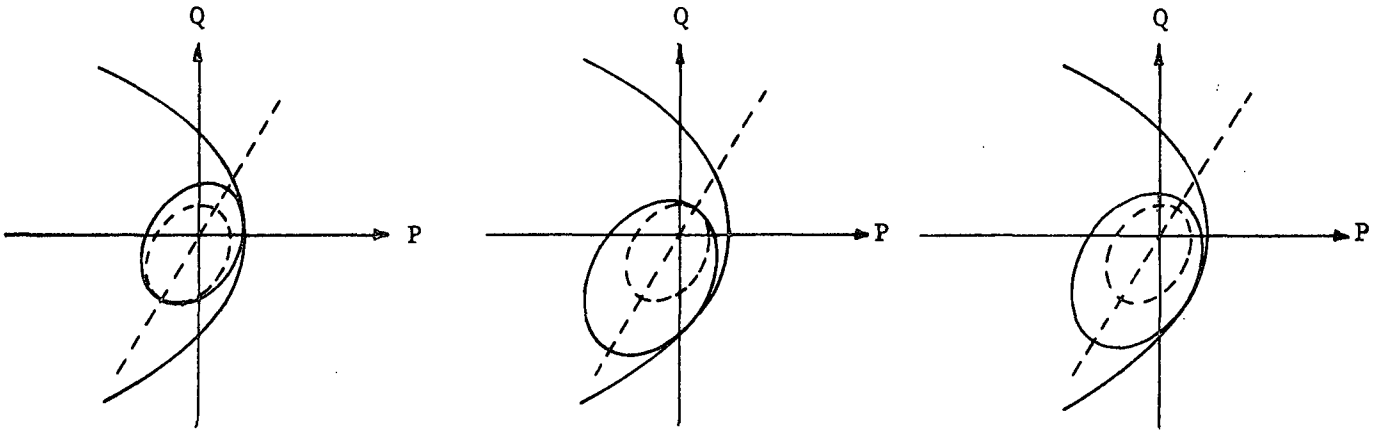


Figure 4.8 Coplanar Orbit Modifications

The details of the computations of these modifications may be found in the analysis of COPINS. Now for the unweighted errors evaluate the differences

$$\epsilon = |\Delta r_p| + |\Delta r_a| \tag{4.56}$$

where Δr_p, Δr_a denote the discrepancies in the desired and modified values of r_p and r_a respectively. The unweighted errors are then multiplied by a weighting factor W_i chosen to rate the preference for the given modification. The first option requires one subsequent orbit trim at apoapsis to trim to the desired orbit; it therefore is given a weight factor $W = 1$. The second option requires a subsequent trim at periapsis which is a less satisfactory maneuver: therefore it has a weighting factor of $W = 2$. The third option requires two subsequent maneuvers at both periapsis and apoapsis: it therefore is assigned a weighting factor of $W = 3$.

After evaluating the weighted errors of each of the maneuvers, the minimum one is selected for the actual insertion execution.

4.5.3 Nonplanar Strategy

The analytic details of the nonplanar orbit insertion strategy are given in the analysis of subroutine NONINS. The overview of this strategy will be discussed here.

The target parameters prescribed under this option are the angles specifying the desired plane in target body equatorial coordinates: the inclination i and the right ascension Ω. The orientation of the ellipse within the plane is fixed by the specification of the argument of periapsis ω. Nominal values for the semimajor axis a and eccentricity e are also read in; however these are altered during the course of the insertion decision to obtain an impulsive solution. These parameters are illustrated in Figure 4.9.

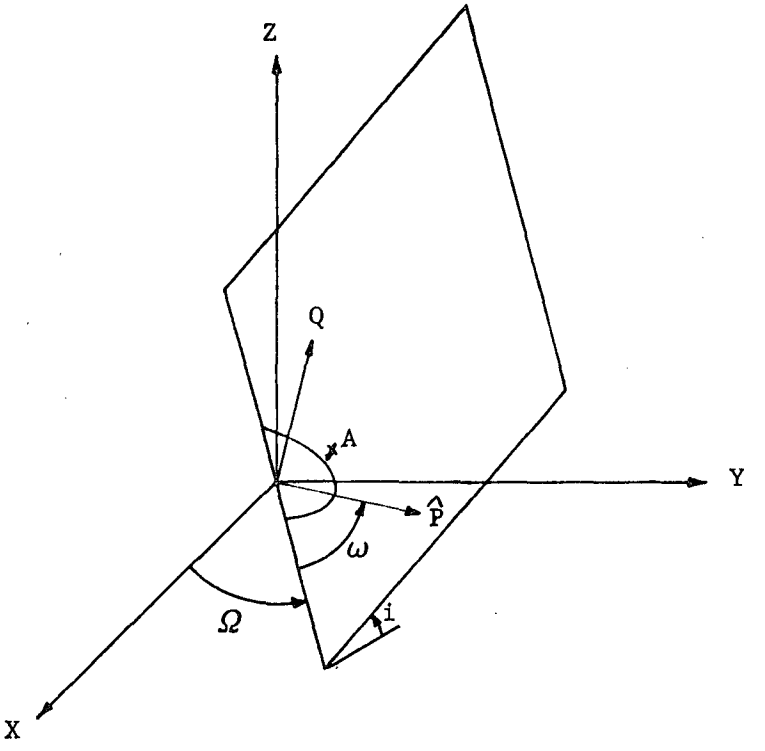


Figure 4.9 Orientation of Desired Orbit

Now suppose that the approach hyperbola intersects the desired plane at two points A and B. The probability is that neither A nor B lie on the desired ellipse. Thus, the desired ellipse must be varied to allow the intersection of those points.

The geometry within the plane of the desired ellipse is indicated in Figure 4.10. For each of the points A and B three modifications of the desired orbit are made:

1) Vary r_a holding r_p fixed

2) Vary r_p holding r_a fixed

3) Vary a holding e fixed

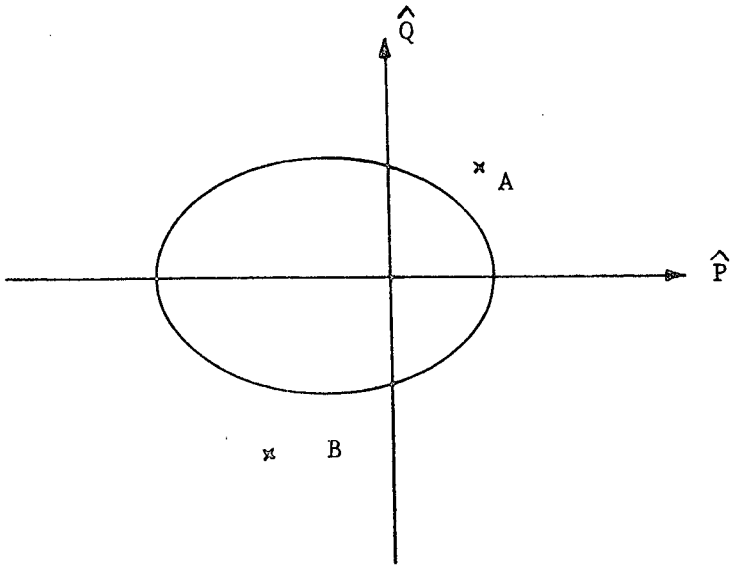


Figure 4.10 Nonplanar Orbit Insertion Modifications

Then loss functions are assigned to each of the solutions according to

$$\epsilon = W_M W_R \left(|\Delta r_a| + |\Delta r_p| \right) \tag{4.57}$$

where Δr_a and Δr_p are the differences in periapsis and apoapsis radii from their desired values. W_M is a weighting function equal to the index of the modifications listed above indicating the preferences for the modifications. W_R is a weighting factor to encourage the choice of a solution on the incoming ray ($W_R = 1$) over the outgoing ray ($W_R = 2$). The solution with the minimum ϵ is then selected for the execution of the insertion maneuver.

The states on the hyperbola $(\vec{r}_h, \vec{v}_h)$ and the modified ellipse $(\vec{r}_e, \vec{v}_e)$ at the selected intersection point are then computed and used to determine the insertion $\vec{\Delta v}$

$$\vec{\Delta v} = \vec{v}_e - \vec{v}_h \tag{4.58}$$

and time before excution Δt.

4.6 Thrusting Arc Modeling

4.6.1 Introduction

The result of the midcourse correction targeting event is the velocity increment $\vec{\Delta v}$ which must be added impulsively to the current nominal state $(\vec{r}, \vec{v})$ in order to yield a trajectory satisfying given target conditions.

If the midcourse correction is to be executed impulsively on the nominal, the state immediately following the maneuver is given by

$$\vec{r}^{+} = \vec{r}^{-}$$

$$\vec{v}^{+} = \vec{v}^{-} + \vec{\Delta v} \qquad (4.59)$$

The midcourse may also be executed by an alternative technique known at the thrusting arc model.

The thrusting arc is intended to add to STEAP the capability to model pulsing type radial engines. The radial engine is mounted on a spin-stabilized spacecraft; thrusts are added periodically as the spacecraft rotates into the proper direction. Thus, the velocity corrections are added as a series of pulses over an extended time interval. The order of magnitude of parameters is such that a maneuver may require up to a thousand pulses over a ten day interval.

The general scheme for the transformation of the impulsive velocity increment into an equivalent series of pulses is indicated in Figure 4.11. The impulsive correction Δv is divided into a number of equal pulses Δv_i all in the same direction as Δv and with the sum of their magnitudes equal to the magnitude of the single impulse Δv.

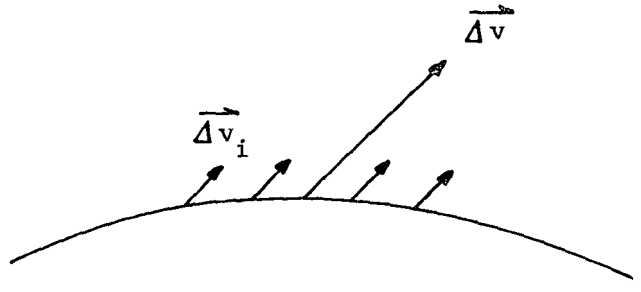


Figure 4.11 Thrusting Arc Modeling

4.6.2 Computation of Sequence

Suppose that the targeting event has generated the required impulsive velocity correction $\vec{\Delta v}$. The problem then becomes to determine a sequence of bounded pulses $\Delta\vec{v}_i$ such that the net effect of adding these pulses sequentially over a time interval is equivalent to adding the single increment $\vec{\Delta v}$ impulsively.

The procedure used in STEAP proceeds as follows. Introduce the engine parameters

T	Thrust magnitude of single pulse
m	Nominal mass of spacecraft
Δt	Duration of single pulse
Δt_i	Time interval between successive pulses

The velocity increment imparted by a single pulse is

$$\Delta v_i = \frac{T\Delta t}{m} \tag{4.60}$$

The number of pulses required then is

$$N_p = \left[\frac{\Delta v}{\Delta v_i}\right] + 1 \tag{4.61}$$

where $[\cdot]$ denotes the greatest integer function. The magnitude of the final pulse must be adjusted to insure that the sum of magnitudes is correct. Thus, the final pulse Δv_f is set to

$$\Delta v_f = \Delta v - (N_p - 1)\,\Delta v_i \tag{4.62}$$

The nominal pulse of the sequence $\vec{\Delta v}_i$ and the final pulse in the sequence $\vec{\Delta v}_f$ are forced to have the same direction as the original impulsive increment

$$\vec{\Delta v}_1 = \Delta v_1 \frac{\vec{\Delta v}}{\Delta v}$$

$$\vec{\Delta v}_f = \Delta v_f \frac{\vec{\Delta v}}{\Delta v} \qquad (4.63)$$

The time interval of the pulsing arc is then given by

$$\Delta T = (N_p - 1)\, \Delta t_1 \qquad (4.64)$$

4.6.3 Perturbed Heliocentric Propagation

Because of the large number of pulses possible in a single maneuver, it is expedient to determine an efficient means of propagating the ballistic trajectory between pulses. A perturbed heliocentric propagator (PERHEL) including the effects of the sun and the launch and target bodies was developed.

The equations of motion of a body moving under the influence of the sun and a perturbing body may be written

$$\ddot{\vec{r}} = -\frac{\mu_o \vec{r}}{r^3} - \frac{\mu(\vec{r} - \vec{r}_m)}{|\vec{r} - \vec{r}_m|^3} - \frac{\mu \vec{r}_m}{r_m^3} \qquad (4.65)$$

where

$\vec{r}$ is the vector radius from the sun to the spacecraft

$\vec{r}_m$ is the vector radius from the sun to the perturbative mass

μ_o, μ are the gravitational constants of the sun and mass respectively

The last term in (4.65) representing the indirect force is discarded as being insignificant at this point. Conic formulae are used to determine the state $\vec{r}_{of}$, $\vec{v}_{of}$ at the end of the interval Δt ignoring the second term. This is simply heliocentric conic propagation then. Perturbations resulting from the second term of the right hand side of (4.65) are then brought in. The assumption is made that the vector $\vec{r} - \vec{r}_m$ is linear in time over the interval Δt

$$\vec{r} - \vec{r}_m = \vec{A}t + \vec{B} \qquad (4.66)$$

With this assumption the perturbation equations may be solved in closed form for the direct term perturbations δr, δv. The state at the end of the interval is then given by

$$\begin{aligned} \vec{r}_f &= \vec{r}_{of} + \vec{\delta r} \\ \vec{v}_f &= \vec{v}_{of} + \vec{\delta v} \end{aligned} \tag{4.67}$$

The detailed derivation of these equations are given in PERHEL. In the actual program the direct term perturbations for both the launch and target bodies are added to the final solution.

Investigating of equation (4.66) indicates that some way of propagating the launch and target bodies over the given interval must be available. To do this the f and g series (see PREPUL) for both the launch and target planets are evaluated at the nominal time of correction. The position of the target planet at some point Δt relative to that time is then given by

$$\begin{aligned} \vec{r}_T\ (\Delta t) &= f_T\ (\Delta t)\ \vec{r}_T\ (0) + g_T\ (\Delta t)\ \vec{v}_T\ (0) \\ f_T\ (t) &= \sum_{k=0}^{6} f_k t^k \\ g_T\ (t) &= \sum_{k=0}^{6} g_k t^k \end{aligned} \tag{4.68}$$

with similar equations for the launch body.

4.6.4 Execution of Sequence

The actual propagation of the spacecraft through the series of pulses may now be explained. The state of the spacecraft, the launch planet and the target are all recorded at the nominal time of the correction t_o. The thrusting arc parameters given by (4.60) to (4.64) are computed for the required $\vec{\Delta v}$. The positions of the launch and target bodies are computed at the beginning of the thrusting arc t_B

$$t_B = t_o - \frac{\Delta T}{2} \tag{4.69}$$

The spacecraft is then propagated backwards to t_B using the perturbed heliocentric propagator PERHEL. The nominal pulse of the arc is then added impulsively

$$(\vec{r}_B, \vec{v}_B^{+}) = (\vec{r}_B, \vec{v}_B^{-} + \overrightarrow{\Delta v}_i) \tag{4.70}$$

The positions of the launch and target bodies an interval Δt_i later are computed and the spacecraft is propagated over that interval by PERHEL. This process is repeated until the final pulse $\overrightarrow{\Delta v}_f$ is added.

The final spacecraft state relative to the sun and the updated time $t + \frac{\Delta T}{2}$ are then sent to the trajectory mode to continue the n-body propagation from the end of the thrusting arc.

5. ERRAN ANALYSIS

5.1 General Description of ERRAN

The error analysis program ERRAN is a preflight mission analysis tool and is concerned primarily with the propagation of covariance matrices along selected interplanetary or lunar trajectories. All random variables are assumed to have gaussian distributions, and linear theory is assumed for the propagation of all covariance matrices.

There are three main quantitative results that come from the error analysis program, all of which are very important for trajectory design during preflight mission analysis. The first output is the orbit determination or navigation uncertainty at selected trajectory times. The processed (knowledge) covariance matrix of orbit determination uncertainty gives a probabilistic answer, for a specific reference trajectory, to the question "how well will the actual trajectory be known after optimal processing of the tracking information?" The error analysis program can be used to study the effects of dynamic model errors, sensor errors, and measurement schedules and types on the orbit determination process.

A second result obtained from the error analysis program is equally important. Orbit determination uncertainties, although they are significant, do not by themselves answer all the pertinent questions related to mission success. Another question that must be answered is, "how close will the actual trajectory come to meeting the specified target conditions?" Because of injection errors and dynamic model errors the actual trajectory will depart from the original targeted nominal trajectory. The statistical measure of such dispersion is represented by the control covariance matrix which, unlike the knowledge covariance discussed above, is unaffected by the processing of tracking information. The propagation of this control covariance forward to the target will provide us with probabilistic information relating to target miss in the absence of midcourse guidance corrections. However, a midcourse guidance correction can be performed to reduce the actual trajectory dispersion about the target. Propagation of the sum of the knowledge covariance and the guidance execution error covariance forward from the midcourse correction time to the target will provide us with probabilistic information relating to target miss following a midcourse guidance correction.

The third main result from the error analysis program is concerned with the probabilistic determination of likely fuel budgets required for interplanetary or lunar missions. Without performing any estimation, the most likely magnitudes of the midcourse correction magnitudes can be computed

along with their variances. This computation permits the mission analyst to calculate reasonable fuel loading requirements that are critical in the design of an actual system.

Two matrix quantities are carried throughout the error analysis program. One is the nominal or reference state vector, which is needed for many computations, and the second is the covariance matrix of navigation uncertainties associated with the state vector. The state vector is comprised of spacecraft position and velocity plus any augmentation parameters included in the analysis. The covariance matrix is a square, symmetric, positive definite matrix of associated uncertainties whose dimension corresponds to that of the state vector.

The computational operation of the error analysis program may be separated into two distinct calculation procedures. The first of these is called the basic cycle and refers to the process of propagating uncertainties from one measurement to the next. A Kalman recursive filtering algorithm with a consider option is used to process the measurement and compute the state vector associated covariance matrix that begins the next step in the basic cycle. Events refer to computations in the error analysis program that are not simply propagations of the navigation uncertainty covariance matrix from one measurement to the next and subsequent optimal filtering of the new measurement. In the error analysis program, three kinds of events are permitted.

The three events allowed in the error analysis program are eigenvector events, prediction events, and guidance events. At an eigenvector event, the position and velocity covariance matrix partitions are diagonalized to reveal geometric information about the size and orientation of the position and velocity navigation uncertainties. At a prediction event, the most recent covariance matrix is propagated forward to some critical trajectory time, usually a guidance correction time, to determine predicted orbit determination uncertainties in the absence of further measurements. When a guidance event occurs, a rather lengthy computational process determines the likely magnitude of the guidance correction together with execution error statistics based on an underlying physical model for the correction process. An option is also available for computing bias aimpoints and bias velocity corrections to satisfy planetary quarantine constraints at each midcourse guidance event. Orbital insertion guidance events are also available.

The next section of this chapter details the Kalman recursive estimation algorithm that is assumed to be the underlying orbit determination procedure. Section 5.3 discusses dynamic and measurement noise covariance matrices. Section 5.4 treats the methods used in the error analysis program for computing state transition matrices. Section 5.5 presents

2-5

the equations required for the computation of observation matrices for each type of measurement. Finally, section 5.6 discusses eigenvector and prediction events. The guidance event is not covered in chapter 5. See chapter 7 for a comprehensive treatment of the guidance event.

5.2 Recursive Estimation Algorithm

The recursive estimation algorithm refers to the computational procedure which combines dynamic model and measurement information to generate estimates of spacecraft position and velocity deviations from the nominal trajectory, estimates of certain dynamic and measurement parameters, and the knowledge covariances associated with these estimates. The error analysis program treats the estimation process in an ensemble sense. Only the knowledge covariances are generated in ERRAN, and not the estimates themselves. The Kalman recursive estimation algorithm with a consider option is modeled in the STEAP programs. But before presenting this estimation algorithm, the linear dynamic and observation models will be described.

The linearized system is assumed to be described by the augmented state vector

$$x^A = \begin{bmatrix} x \\ x_s \\ u \\ v \end{bmatrix} \tag{5.1}$$

where

x = spacecraft position/velocity state (dimension 6)

x_s = solve-for parameter state (dimension n_1)

u = dynamic consider parameter state (dimension n_2)

v = measurement consider parameter state (dimension n_3)

All the above state vectors represent deviations from nominal state vectors and all parameters are assumed to be constant. The distinction between solve-for and consider parameters will be clarified subsequently.

The linearized dynamic model is assumed to have form

$$x_{k+1} = \Phi(t_{k+1}, t_k)x_k + \Theta_{xx_s}(t_{k+1}, t_k)x_{s_k} + \Theta_{xu}(t_{k+1}, t_k)u_k + q_k \qquad (5.2)$$

where $\Phi(t_{k+1}, t_k)$, $\Theta_{xx_s}(t_{k+1}, t_k)$, and $\Theta_{xu}(t_{k+1}, t_k)$ are state transition matrices over the time interval $[t_k, t_{k+1}]$ relating changes in x, x_s, and u, respectively, at time t_k to changes in x at time t_{k+1}. The variable q_k represents the effect of dynamic noise over the interval.

The linearized observation model is assumed to have form

$$y_k = H_k x_k + M_k x_{s_k} + G_k u_k + L_k v_k + \eta_k \qquad (5.3)$$

where observation matrices H_k, M_k, G_k, and L_k relate changes in x, x_s, u, and v, respectively, to changes in the observable y. All observation matrices are evaluated at the nominal condition. The variable η_k represents measurement noise.

Under the usual assumption of white noise, the dynamic and measurement noise statistics are described by

$$E[q_k] = E[\eta_k] = 0$$

$$E[q_k q_j^T] = Q_k \delta_{jk}$$

$$E[\eta_k \eta_j^T] = R_k \delta_{jk}$$

The equations constituting the recursive estimation algorithm are of two types: state estimation equations and knowledge covariance equations. Only the latter will be presented below. The state estimation equations will be presented in chapter 6.

An estimation algorithm with no consider option treats all assumed dynamic and measurement parameters as "solve-for" parameters, i.e., the estimation algorithm generates estimates of the parameters as well as estimates of the spacecraft position and velocity. Continued processing of measurements will often reduce knowledge covariances to unrealistically low values, a situation which can induce divergence in the estimation algorithm. Divergence is said to occur when the actual estimation error grows without bound. One method used to prevent divergence is to incorporate a consider option into the algorithm and divide all assumed parameter into either solve-for or consider parameters. Consider parameters are not estimated by the algorithm, nor can their knowledge covariances be reduced by measurement processing. In essence, by not solving for all parameters in the assumed parameter set the algorithm acknowledges the fact that its assumed set of dynamic and measurement parameters do not fully describe the real world, and that it is impossible to reduce parameter uncertainties indefinitely.

The knowledge covariance for the augmented state is defined as

$$P_k^A = E\left[(\hat{x}^A - x^A)\ (\hat{x}^A - x^A)^T \right] \qquad (5.4)$$

where $\hat{x}$ indicates estimated values and x indicates actual values. Introducing equation (5.1) into equation (5.4) and expanding the result permits us to write the covariance matrix in the following partitioned form:

$$P_k^A = \begin{bmatrix} P_k & C_{xx_{s_k}} & C_{xu_k} & C_{xv_k} \\ C^T_{xx_{s_k}} & P_{s_k} & C_{x_s u_k} & C_{x_s v_k} \\ C^T_{xu_k} & C^T_{x_s u_k} & U_o & C_{uv_k} \\ C^T_{xv_k} & C^T_{x_s v_k} & C^T_{uv_k} & V_o \end{bmatrix} \qquad (5.5)$$

Covariance matrix partitions P, P_s, U_o, and V_o are all symmetric and represent the covariances of the spacecraft position/velocity state, solve-for parameters, dynamic consider parameters, and measurement consider parameters, respectively. The off-diagonal covariance matrix partitions represent the correlations between the two variables indicated by the subscripts. Thus, $C_{x_s u}$ represents the correlation between solve-for parameters and dynamic consider parameters.

The assumptions implicit in the consider option entail that covariances U_o and V_o remain constant with time. Estimates u and v are always zero. Although the consider option does not require it, it is realistic to assume no correlation between dynamic consider parameters and measurement consider parameters exists, so that C_{uv} is always zero.

The covariance equations involved in the estimation algorithm are of two types: prediction equations and filtering equations. The prediction equations describe the behavior of the covariance matrix partitions as they are propagated forward in time with no measurement processing. The filtering equations define the covariance updating procedure whenever a measurement is processed. Details of their derivation can be found in reference 10.

The prediction equations are summarized below:

$$P^-_{k+1} = (\Phi P^+_k + \Theta_{xx_s} C^{+T}_{xx_{s_k}} + \Theta_{xu} C^{+T}_{xu_k})\Phi^T + C^-_{xx_{s_{k+1}}} \Theta^T_{xx_s} + C^-_{xu_{k+1}} \Theta^T_{xu} + Q_k \quad (5.6)$$

$$C^-_{xx_{s_{k+1}}} = \Phi C^+_{xx_{s_k}} + \Theta_{xx_s} P^+_{s_k} + \Theta_{xu} C^{+T}_{x_s u_k} \quad (5.7)$$

$$P^-_{s_{k+1}} = P^+_{s_k} \quad (5.8)$$

$$C^-_{xu_{k+1}} = \Phi C^+_{xu_k} + \Theta_{xx_s} C^+_{x_s u_k} + \Theta_{xu} U_o \quad (5.9)$$

$$C^{-}_{x_s u_{k+1}} = C^{+}_{x_s u_k} \tag{5.10}$$

$$C^{-}_{xv_{k+1}} = \Phi C^{+}_{xv_k} + \Theta_{xx_s} C^{+}_{x_s v_k} \tag{5.11}$$

$$C^{-}_{x_s v_{k+1}} = C^{+}_{x_s v_k} \tag{5.12}$$

A minus superscript on covariance partitions indicates the covariance partition immediately prior to processing a measurement; a plus superscript, immediately after processing a measurement.

The filtering equations involve equations for the measurement residual covariance matrix J, Kalman gain matrices K and S, and covariance updating. The measurement residual covariance matrix is given by

$$J_{k+1} = H_{k+1} A_{k+1} + M_{k+1} B_{k+1} + G_{k+1} D_{k+1} + L_{k+1} E_{k+1} + R_{k+1} \tag{5.13}$$

where

$$A_{k+1} = P^{-}_{k+1} H^{T}_{k+1} + C^{-}_{xx_{s_{k+1}}} M^{T}_{k+1} + C^{-}_{xu_{k+1}} G^{T}_{k+1} + C^{-}_{xv_{k+1}} L^{T}_{k+1}$$

$$B_{k+1} = P^{-}_{s_{k+1}} M^{T}_{k+1} + C^{-T}_{xx_{s_{k+1}}} H^{T}_{k+1} + C^{-}_{x_s u_{k+1}} G^{T}_{k+1} + C^{-}_{x_s v_{k+1}} L^{T}_{k+1}$$

$$D_{k+1} = C^{-T}_{xu_{k+1}} H^{T}_{k+1} + C^{-T}_{x_s u_{k+1}} M^{T}_{k+1} + U_o G^{T}_{k+1}$$

$$E_{k+1} = C^{-T}_{xv_{k+1}} H^{T}_{k+1} + C^{-T}_{x_s v_{k+1}} M_{k+1} + V_o L^{T}_{k+1}$$

The Kalman gain matrices for both position/velocity state and solve-for parameters are given by

$$K_{k+1} = A_{k+1} J_{k+1}^{-1} \tag{5.14}$$

$$S_{k+1} = B_{k+1} J_{k+1}^{-1} \tag{5.15}$$

The covariance partitions immediately after processing a measurement are given by

$$P_{k+1}^{+} = P_{k+1}^{-} - K_{k+1} A_{k+1}^{T} \tag{5.16}$$

$$C_{xx_{s}}^{+}{}_{k+1} = C_{xx_{s}}^{-}{}_{k+1} - K_{k+1} B_{k+1}^{T} \tag{5.17}$$

$$P_{s}^{+}{}_{k+1} = P_{s}^{-}{}_{k+1} - S_{k+1} B_{k+1}^{T} \tag{5.18}$$

$$C_{xu}^{+}{}_{k+1} = C_{xu}^{-}{}_{k+1} - K_{k+1} D_{k+1}^{T} \tag{5.19}$$

$$C_{x_{s}u}^{+}{}_{k+1} = C_{x_{s}u}^{-}{}_{k+1} - S_{k+1} D_{k+1}^{T} \tag{5.20}$$

$$C_{xv}^{+}{}_{k+1} = C_{xv}^{-}{}_{k+1} - K_{k+1} E_{k+1}^{T} \tag{5.21}$$

$$C_{x_{s}v}^{+}{}_{k+1} = C_{x_{s}v}^{-}{}_{k+1} - S_{k+1} E_{k+1}^{T} \tag{5.22}$$

It should be noted that the covariance matrices themselves are not printed out in STEAP. Rather, all variances appearing along the diagonal of the augmented covariance matrix defined by equation (5.5) are converted to standard deviations and all off-diagonal covariances are converted to correlation coefficients. Thus, if covariance a_{ij} is an element

of the augmented covariance matrix, then the correlation coefficient is given by

$$\rho_{ij} = \frac{a_{ij}}{\sigma_i \sigma_j} \quad , \quad i \neq j$$

where standard deviations σ_i and σ_j are given by $\sigma_i = a_{ii}^{1/2}$ and $\sigma_j = a_{jj}^{1/2}$. Following these transformations all standard deviations and correlation matrix partitions are then printed out.

5.3 Dynamic and Measurement Noise Covariance Matrices

The problem of filter divergence has been mentioned in the previous section in connection with the consider option. The basic cause of divergence is modeling insufficiency and many separate categories of this insufficiency can be enumerated. The causes of the divergence problem and possible solutions to it are given in greater depth in the analytical discussion of the simulation program. The purpose of including a dynamic noise matrix Q in the error analysis program is to examine the effect of dynamic model insufficiency on the key outputs of the error analysis program. Some dynamic or unmodeled noise always corrupts an interplanetary trajectory; what is interesting, from the point of view of the error analysis program, is how the primary quantitative outputs are affected by various levels of dynamic noise.

The dynamic noise model used in the error analysis program is somewhat arbitrary and its interpretation is difficult. Over any time interval Δt between measurements, the dynamic noise matrix Q is computed from three input constants that remain the same throughout a trajectory run. These three constant inputs K_1, K_2, and K_3, whose units are km^2/sec^4, roughly correspond to variances of assumed unmodeled accelerations. The dynamic noise matrix Q added over any interval Δt is diagonal. Specifically, if Δt is the interval between measurements, the six nonzero terms of Q are given by

$$
\begin{aligned}
Q_{11} &= \tfrac{1}{4} K_1 \Delta t^4 \\
Q_{22} &= \tfrac{1}{4} K_2 \Delta t^4 \\
Q_{33} &= \tfrac{1}{4} K_3 \Delta t^4 \\
Q_{44} &= K_1 \Delta t^2 \\
Q_{55} &= K_2 \Delta t^2 \\
Q_{66} &= K_3 \Delta t^2
\end{aligned}
\qquad (5.23)
$$

Some explanation of this form for the dynamic noise is necessary. It was decided early in the design of the program that the physical interpretation of arbitrary dynamic noise must be made possible by relating the Q matrix, in some fashion, to unmodeled accelerations. Similarly, it appeared that the magnitude of the dynamic noise should be a function of the specific time interval over which it was added; in other words, the dynamic noise added when two days were between measurements should be greater than that added when only two hours separated the two measurements.

The first attempt to satisfy these two constraints resulted in the assumption that the unmodeled accelerations could be represented as biases with zero mean and variances K_1, K_2, K_3. Consider, for example, a vector random variable $(\delta\ddot{X}, \delta\ddot{Y}, \delta\ddot{Z})^T$ with variances

$$\sigma^{2..}_{\delta X} = K_1 \qquad \sigma^{2..}_{\delta Y} = K_2 \qquad \sigma^{2..}_{\delta Z} = K_3$$

and correlation coefficients set equal to zero. If these accelerations represent biases, then over any interval Δt they are related to position and velocity uncertainties through

$$\delta\dot{X} = \delta\ddot{X}\ (\Delta t); \quad \delta X = \frac{1}{2}\ (\delta\ddot{X})\ (\Delta t)^2$$

and similarly for the other components. Under this model for the dynamic noise, the Q matrix would be the same as that given in equation (5.23) except for the completely correlated off-diagonal terms resulting in

$$Q_{14} = \frac{1}{2} K_1 \Delta t^3, \quad Q_{25} = \frac{1}{2} K_2 \Delta t^3, \quad Q_{36} = \frac{1}{2} K_3 \Delta t^3$$

Clearly, if the unmodeled accelerations are indeed biases, the δX and $\delta \dot{X}$ uncertainties due strictly to the dynamic noise must be completely correlated.

This initial model for the dynamic noise was unsatisfactory for two reasons. First, the resulting error analysis was forced to assume that the unmodeled acceleration was a constant bias throughout the trajectory as well as over each interval. The physics of the problem suggests that unmodeled accelerations are probably constant biases over short periods, but over an entire trajectory they probably vary considerably. Secondly, if the values for K_j are large enough for the dynamic noise to significantly affect the processed covariance matrices, their total correlation induces an unrealistically high correlation between the same terms in the resulting uncertainty matrices.

A more careful modeling of the stochastic process was discarded due to the arbitrary nature of the Q matrix. The dynamic noise matrix was chosen as in equation (5.23) because uncoupling the position and velocity uncertainties due to unmodeled accelerations retained a physical feel for the meaning of Q and permitted its computation to be viewed as a combination of random and bias error in the unmodeled accelerations.

The measurement noise covariance matrix R requires little comment. We simply assume the measurement noise for each measurement type has constant statistics, and hence constant covariance matrix R, for a given mission.

5.4 State Transition Matrices

State transition matrices describe the dynamic behavior of linear systems. Before presenting the different techniques that are available in the STEAP programs for computing state transition matrices, the derivation of the general form of the linear system modeled in STEAP will be summarized.

The nonlinear equations describing the motion of the spacecraft have form

$$\dot{X} = f(X, W, t) \qquad (5.24)$$

where X denotes the spacecraft position/velocity state and W is a vector of dynamic parameters which define the dynamic model. The linearized version of equation (5.24) is given by

$$\dot{x} = \frac{\partial f}{\partial X} x + \frac{\partial f}{\partial W} w \qquad (5.25)$$

where x and w represent linear deviations from nominal states $\overline{X}$ and $\overline{W}$, respectively. Partial derivative matrices $\frac{\partial f}{\partial X}$ and $\frac{\partial f}{\partial W}$ are evaluated along the nominal state.

The discrete solution of equation (5.25) over the time interval $[t_k, t_{k+1}]$ is given by

$$x_{k+1} = \Phi(t_{k+1}, t_k)\, x_k + \theta(t_{k+1}, t_k)\, w_k \qquad (5.26)$$

where state transition matrices $\Phi(t_{k+1}, t_k)$ and $\theta(t_{k+1}, t_k)$ are required to define the solution. In STEAP the parameter deviation vector w is assumed to be constant. By dividing parameters into solve-for and consider parameters, we could expand equation (5.26) into equation (5.2).

Three methods are available in STEAP for computing the 6 x 6 state transition matrix Φ. The first two methods, which are analytical methods, are analytical patched conic and analytical virtual mass. The third method uses numerical differencing to compute Φ. In the analytical techniques it is assumed that the spacecraft trajectory is a two-body conic section over a small time interval, and that perturbations about the nominal trajectory can be related by using the basic analytical two-body matrizant. To increase the accuracy of these analytical techniques over longer time intervals a state transition matrix cascading option is provided in STEAP (see subroutine CASCAD analysis for more details).

In the present version of STEAP the state transition matrix θ, relating parameter deviations to position/velocity deviations, is always computed using the numerical differencing technique.

5.4.1 Analytical Patched Conic

The basic idea in using an analytic patched conic state transition matrix is that over a small time interval of an interplanetary flight, the motion of a spacecraft is essentially a two-body conic section. Based on the foregoing assumption, Danby (ref. 6) has developed a set of general equations for determining the state transition matrices by the use of matrizants. The matrizant of two-body motion is used in STEAP for both analytical methods of computing state transition matrices. The basic fundamentals and equations of Danby's method will be presented here. Complete derivations are given in reference 5.

Letting $x(t)$ represent a column vector composed of position and velocity deviations at time t, $x(t_o)$ the same for time t_o, and g the deviation of a set of six geometrical elements, an equation that relates small deviations in position and velocity at two different times can be written as

$$x(t) = M(t)g = M(t)M^{-1}(t_o)\, x\, (t_o) \tag{5.27}$$

Thus, the state transition matrix is given by

$$\Phi(t, t_o) = M(t)M^{-1}(t_o) \tag{5.28}$$

The reference coordinate system considered here has the X-axis pointing toward periapsis for the conic, the Z-axis along the angular momentum vector, and Y forming the triad. Danby (ref. 5) calls this the "orbital reference system".

The geometrical orbital elements defined by g may be set in a column vector as,

$$g = \begin{bmatrix} \dfrac{\delta \ell_o + \eta \delta r}{n} \\ \dfrac{a\delta e}{h} \\ \delta p \\ \dfrac{\delta a}{2a} \\ \dfrac{ae\delta r}{h} \\ \delta q \end{bmatrix} \tag{5.29}$$

where ℓ_o is the mean anomaly at an arbitrary epoch; a is the semimajor axis of the orbit; e is the eccentricity of the orbit; δp, δq, and δr

are infinitesimal rotations about the reference axis; h is the angular momentum per unit mass; and n is the mean motion of the orbit. The auxiliary parameter η is defined by $(1 - e^2)^{1/2}$. Avoiding the algebraic manipulations, the resultant M matrix as given by Danby (ref 5) has the following form,

$$M(t) = \begin{bmatrix} \dot{X} & \dot{Y}X-h & 0 & 2X-3t\dot{X} & \dot{Y}Y & 0 \\ \dot{Y} & -\dot{X}X & 0 & 2Y-3t\dot{Y} & -\dot{Y}X-2h & 0 \\ 0 & 0 & Y & 0 & 0 & -X \\ \ddot{X} & \dot{Y}\ddot{X}+\ddot{Y}X & 0 & -\dot{X}-3t\ddot{X} & \dot{Y}^2+\ddot{Y}Y & 0 \\ \ddot{Y} & -\dot{X}^2-\ddot{X}X & 0 & -\dot{Y}-3t\ddot{Y} & -\dot{X}\dot{Y}-\ddot{Y}X & 0 \\ 0 & 0 & \dot{Y} & 0 & 0 & -\dot{X} \end{bmatrix} \qquad (5.30)$$

where X, Y, and Z are the components of position and velocity along the particular orbit, t is some specified epoch, and the accelerations are given by

$$\ddot{X} = \frac{-\mu X}{R^3}, \quad \ddot{Y} = \frac{-\mu Y}{R^3}, \quad \ddot{Z} = \frac{-\mu Z}{R^3}$$

R is the magnitude of the position deviation represented by $(X^2 + Y^2 + Z^2)^{1/2}$ and μ is the gravitational constant of the dominant body used in the two-body approximation.

The inverse of the M matrix at the initial time t_o is given by

$$M^{-1}(t_o) = A\gamma M^T(t)\,\gamma^T \qquad (5.31)$$

where A is a diagonal matrix of dimension 6 x 6 and has diagonal components $(a/\mu, a/\mu h, 1/h, a/\mu, a/\mu h, 1/h)$. γ is given as the matrix

$$\gamma = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix},$$

with I being the identity matrix of appropriate dimension.

The state transition matrix that relates perturbations about some nominal state vector between two arbitrary times can now be determined by combining equations (5.30) and (5.31). The resulting matrix is referenced to the orbit plane coordinate system and thus, because all computations in STEAP are performed in the ecliptic frame, a rotation needs to be included so that

$$\Phi(t, t_o)_{\text{ecliptic}} = R\,\Phi(t, t_o)_{\text{orbit plane}}\,R^T \qquad (5.32)$$

where R is the rotation matrix.

In using the foregoing method for analytical patched conic determination of the state transition matrices, an automatic check is made in the program to determine what sphere of influence the vehicle is in at the time of computation. The sphere of influence determines what gravitational mass and dominant body location will be used to compute the matrizant. It should be stressed that the particular gravitational constant being used at the time of computing Φ is chosen at the beginning of the time interval. In other words, if a check is made at t_1 and the sphere of influence is that of the Sun and the trajectory at t_2 is inside the sphere of influence of the target planet, then μ_{Sun} will be used in the algorithm. No significant problems have resulted by using this approximate strategy, primarily because most of the time intervals are small when state transition matrices are computed near the spheres of influence.

The method of computing state transition matrices by the analytical patched conic technique is assumed in the program unless otherwise specified by the input.

5.4.2 Analytical Virtual Mass

Computation of state transition matrices by the analytical virtual mass technique is similar to the patched conic method. The same general equations developed by Danby (ref. 5) are also used in determining state transition matrices using the virtual mass concept.

The virtual mass technique requires that the location and magnitude of the virtual mass, as calculated by the virtual mass subroutine, be stored for use in the computation of Φ. Once the computational intervals and values for the location and magnitude of the virtual mass have been determined for the nominal trajectory these same quantities are used to

generate the state transition matrix. Hence, after determining the nominal trajectory, the nominal state vector $X(t)$ is available along with a set of values $\vec{r}_v(t)$ and $\mu_v(t)$ representing the position and magnitude of the virtual mass.

As mentioned previously, the equations for the two-body matrizant are also employed in this second method of computing the state transition matrix. However, now the dominant body is assumed to be the effective force center. Recall that in the analytic patched conic method, a check was made to determine what sphere of influence the vehicle was in at the beginning of the time interval. In using virtual mass concepts to compute the state transition matrices, a sphere of influence check is avoided. When calling the state transition matrix module, the gravitational parameter of the virtual mass μ_v is used instead of the μ of the dominant body. The location of the virtual mass is likewise used in the determination of Φ under this method.

5.4.3 Numerical Differencing

The method used to compute Φ and θ using numerical differencing will be presented in this section. Consider first the computation of Φ. Let $\zeta_j(t_{k+1}, t_k)$ represent the j-th column of $\Phi(t_{k+1}, t_k)$. We assume we have available nominal states $\bar{X}_k$ and $\bar{X}_{k+1}$. To obtain ζ_j we increment the j-th element of $\bar{X}_k$ by the numerical differencing factor Δx_j and numerically integrate equation (5.24) over the time interval $[t_k, t_{k+1}]$ to obtain the new spacecraft state X^j_{k+1}. (The j-superscript indicates X^j_{k+1} was obtained by incrementing the j-th element of $\bar{X}_k$.) Then

$$\zeta_j(t_{k+1}, t_k) = \frac{X^j_{k+1} - \bar{X}_{k+1}}{\Delta x_j} \qquad (5.33)$$

$$j = 1, 2, \ldots 6$$

The computation of θ is quite similar. Let ξ_j represent the j-th column of $\theta(t_{k+1}, t_k)$. Again we assume nominal states $\bar{X}_k$ and $\bar{X}_{k+1}$ are available. To obtain ξ_j we increment the j-th element of the parameter vector $\bar{W}$ by the numerical differencing factor Δw_j and numerically integrate equation (5.24) over the time interval $[t_k, t_{k+1}]$ to obtain the

new spacecraft state X^j_{k+1}. Then

$$\xi_j(t_{k+1}, t_k) = \frac{X^j_{k+1} - \bar{X}_{k+1}}{\Delta w_j} \qquad (5.34)$$

$$j = 1, 2, \ldots, n_p$$

where n_p is the number of dynamic parameters.

5.5 Observation Matrices

Observation matrices relate deviations in spacecraft position/velocity state and deviations in dynamic and measurement parameters from nominal values to deviations in observables from their nominal values. Before discussing the observation or measurement types available in STEAP and the technique used to construct observation matrices, the derivation of the linearized observation equation will be summarized.

The general nonlinear observation equation has form

$$Y = f(X, W, t) \qquad (5.35)$$

where Y denotes the observable, X denotes the spacecraft position/velocity state, and W is a vector of dynamic and measurement parameters. The linearized version of equation (5.35) is given by

$$y = \frac{\partial f}{\partial X} x + \frac{\partial f}{\partial W} w \qquad (5.36)$$

where y, x, and w represent deviations from nominal $\bar{Y}$, $\bar{X}$, and $\bar{W}$, respectively, and partial derivative matrices $\frac{\partial f}{\partial X}$ and $\frac{\partial f}{\partial W}$ are evaluated at the nominal condition.

If we partition the parameter vector w into a solve-for parameter vector x_s, a dynamic consider parameter vector u, and a measurement consider vector v, then equation (5.36) can be written as

$$y = Hx + Mx_s + Gu + Lv \qquad (5.37)$$

where we have defined $H = \frac{\partial f}{\partial X}$, and partitioned $\frac{\partial f}{\partial W}$ into three sub-matrices M, G, and L. Adding measurement noise to this equation, we would obtain equation (5.3).

Two categories of observables or measurements are available in STEAP: earth-based range and range-rate measurements and onboard optical measurements. Earth-based range and range-rate measurements can be taken from 4 tracking stations, one of which is an idealized station located at the center of the earth, while the remaining three can be positioned at arbitrary locations on the surface of the earth. The relevant geometry for such measurements is depicted in Figure 5.1. The X, Y, Z coordinate system represents the inertial ecliptic coordinate system, which can be centered at the Sun or the barycenter according to the nature of the

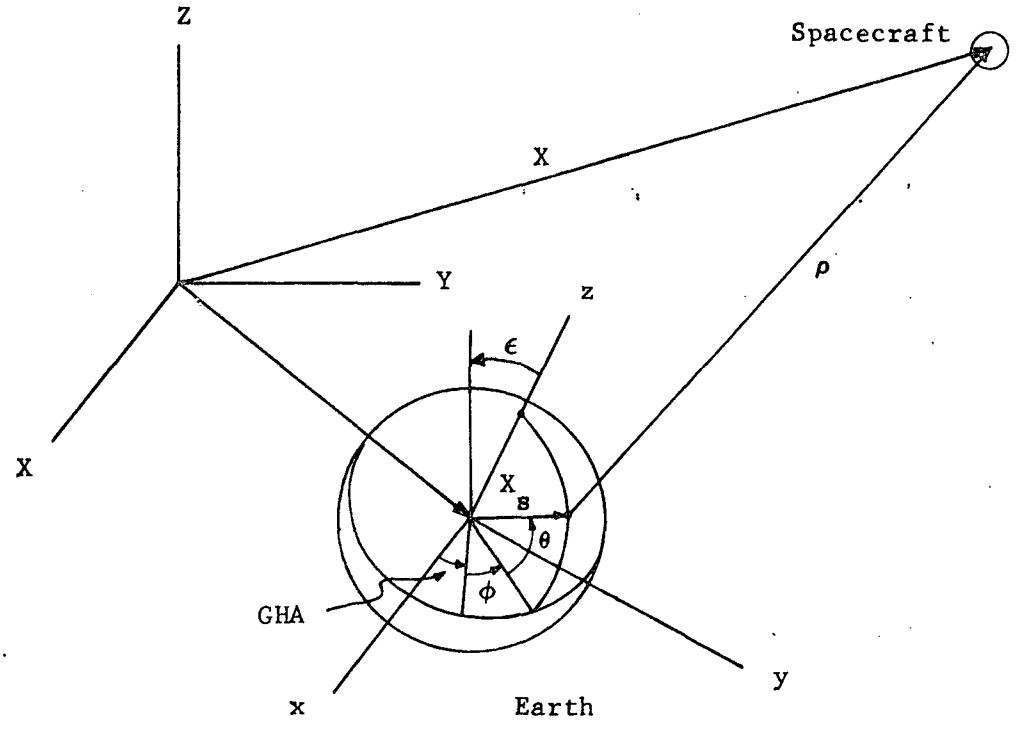


Figure 5.1 Earth-based Tracking

mission. The x, y, z coordinate system represents the geocentric equatorial coordinate system. Axis x is always aligned with axis X. The rotation of this coordinate system relative to the X, Y, Z system is defined by ϵ, the obliquity of the ecliptic. The states of the spacecraft and the Earth relative to inertial space are given by X and X_E, respectively. The tracking station state relative to the center of the Earth is denoted by X_S. The geographical location of the station is defined by radius $R = |X_S|$, latitude Θ, and longitude ϕ. Longitude ϕ is measured positive east from the Greenwich meridian. The hour angle of Greenwich is denoted by GHA. Finally, the position of the spacecraft relative to the tracking station is given by the vector ρ. The scalar observables are range, also denoted by ρ, and range-rate $\dot{\rho}$.

Onboard optical measurements modeled in STEAP are 3 star-planet angle measurements and an apparent planet diameter measurement. The relevant geometry for such measurements is depicted in Figure 5.2. The position

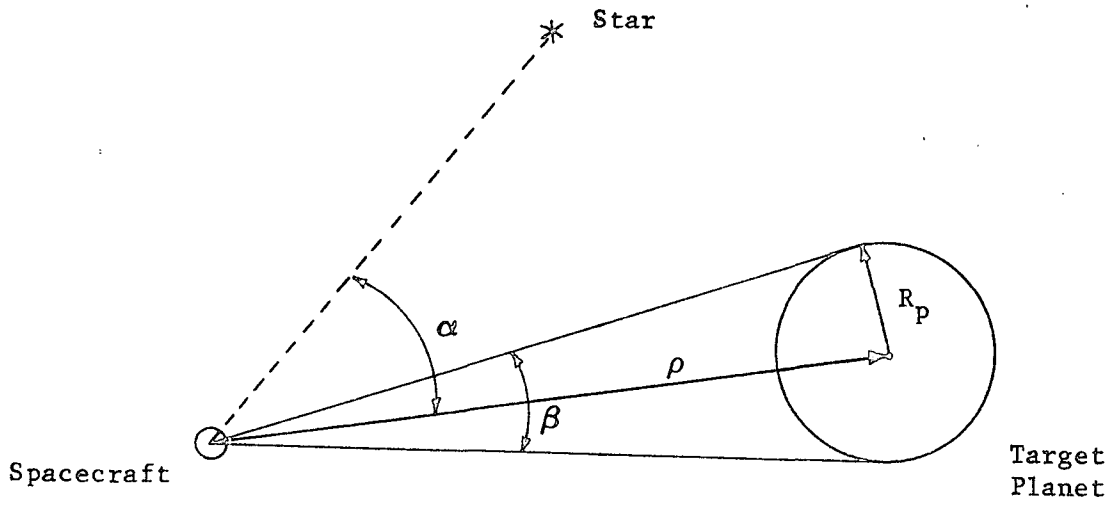


Figure 5.2 Onboard Tracking

of the target planet center relative to the spacecraft is denoted by the vector ρ. Target planet radius is given by R_p. The star-planet angle α is defined to be the angle between ρ and the spacecraft/star line of sight. The locations of three arbitrary stars are modeled in STEAP. The apparent planet diameter measurement is indicated by the angle β.

The nonlinear observation equations for all the measurement types discussed above are summarized in the subroutine TRAKS analysis section. Also presented there are expressions for the partial derivatives required to construct the observation matrix partitions H, M, G, and L.

5.6 Eigenvector and Prediction Events

At an eigenvector event we simply transform the knowledge or navigation uncertainty covariance matrix P into useful geometrical information, which includes eigenvalues, eigenvectors, and hyperellipsoids. Define t_k as the time of the last processed measurement before the eigenvector event and let $\bar{X}_k$ and P_k^+ be, respectively, the nominal trajectory and the orbit determination uncertainty covariance matrix after processing the measurement at t_k. If t_j is the time of the eigenvector event, then $\bar{X}_j$, the nominal state vector at t_j, is computed from the virtual mass trajectory subroutine. The navigation uncertainty covariance matrix at t_j defined by P_j is given by equation (5.6) with subscript k+1 replaced by j. All state transition matrix partitions are understood to be defined over the time interval $[t_k, t_j]$.

The eigenvalues, eigenvectors, and hyperellipsoid could be obtained for the 6 x 6 P_j matrix, but their geometrical interpretation is difficult. If, instead, we operate on the 3 x 3 position and velocity partitions of P_j we can obtain geometrical information which is both useful and readily interpreted.

Let P_R and P_v denote the position and velocity partitions, respectively, of covariance P_j. Then at an eigenvector event these partitions are diagonalized to produce position and velocity eigenvalues and eigenvectors. The principal axis associated with the minimum eigenvalue defines the direction of minimum uncertainty; the axis associated with the maximum eigenvalue defines the direction of maximum uncertainty. The method employed is described in more detail in the subroutine JACOBI analysis section. Next, 1σ or 3σ (or both) hyperellipsoids of uncertainty, in

both position and velocity space are computed to show the size and geometric orientation of the navigation uncertainties. These ellipsoids are then projected onto each of the two-dimensional planes to show additional geometric information. The analytical details of this procedure are presented in the subroutine HYELS analysis section.

At a prediction event at time t_j, the nominal trajectory $\bar{X}_j$ and associated knowledge covariance P_j are first computed just as at an eigenvector event. Now define t_p as the time to which the prediction is being made. Then the knowledge covariance at t_p, assuming no measurements over the time interval $[t_j, t_p]$, can be computed using equation (5.6) with $t_k = t_j$ and $t_{k+1} = t_p$.

Within the prediction event algorithm of the error analysis program, the resulting covariance matrix P_p at the prediction time is also diagonalized to produce eigenvector, eigenvalue, and hyperellipsoid information. Thus, by superimposing this geometrical information about P_p for different prediction event times t_j, one can observe the effect of additional tracking on predicted navigation uncertainties.

If time t_p occurs inside the target planet sphere of influence, Cartesian position and velocity uncertainties are transformed to uncertainties in the B-plane parameters B·T, B·R, S·R, and S·T, time of flight t_f, and energy C_3. In addition, the B·T, B·R covariance matrix is diagonalized and the maximum and minimum eigenvalues are computed. The square roots of the maximum and minimum eigenvalues can be identified with the (1σ) uncertainties in the semimajor axis SMAA and the semiminor axis SMIA, respectively, of the uncertainty ellipse in the B-plane. The orientation of this ellipse in the B-plane is also computed. The analytical details of this procedures are presented in the analysis sectors of subroutines PRESIM, BEPS, and BPLANE.

6. SIMUL ANALYSIS

6.1 General Description of SIMUL

There is an essential difference between the philosophies governing the error analysis and simulation programs of STEAP. The error analysis program is primarily a preflight mission analysis tool that gives information related to uncertainties about some specified nominal trajectory. By contrast, the simulation program is designed for a detailed analysis of the orbit determination procedure and its efficacy in the presence of a host of possible anomalies. The error analysis program might be used to determine the nominal trajectory design for a specific mission; the simulation program then "flies" the mission, within the computer, and can provide invaluable information for mission operations.

The computational structure of the simulation program is similar to that of the error analysis program. There is a basic cycle in which subsequent measurements are processed consecutively and there are events where calculations not specifically related to the measurement-processing cycle are made. Section 6.2 outlines in detail the equations and logic used within the basic cycle of the simulation program. In section 6.3 eigenvector, prediction, and quasi-linear filtering events are treated. Finally, section 6.4 includes a discussion of the problems of divergence and non-observability that can plague an orbit determination procedure. The guidance event is not covered in chapter 6, but is treated in chapter 7 together with the error analysis guidance event.

The computations themselves, within the simulation mode, are not any more difficult than in the error analysis mode. There are, however, many more of them and in the discussion to follow some of the most important features of the simulation program will be discussed.

6.2 The Basic Cycle

Recall that in the error analysis program only two quantities, the targeted nominal state vector $\bar{X}$ and the knowledge covariance matrix P, were carried along through each step in the basic cycle. Within the simulation program there are six key quantities carried from step to step in the basic cycle. These six quantities are summarized below:

1) $\bar{X}_k$ — Targeted nominal position/velocity state vector at time t_k; updated at each guidance event.

2) $\tilde{X}_k$ — Most recent nominal position/velocity state vector at time t_k; differs from $\bar{X}_k$ after a quasi-linear filtering event that updates the state vector by the estimate; identical to $\bar{X}_k$ immediately after a guidance event.

3) $\delta\tilde{X}_k$, or $\tilde{x}_k$ — Actual position/velocity state vector deviation from most recent nominal state.

4) $\delta\tilde{\hat{X}}_k$, or $\tilde{\hat{x}}_k$ — Estimated position/velocity state vector deviation from most recent nominal state.

5) $\delta\tilde{\hat{X}}_{s_k}$, or $\tilde{\hat{x}}_{s_k}$ — Estimated solve-for parameter deviations from nominal parameter values.

6) P_k — Knowledge covariance matrix after processing all measurements up to and including time t_k. See section 5.2 for definitions of the whole set of knowledge covariance matrix partitions which are carried from step to step in the basic cycle.

The basic cycle of the simulation program refers to the computational process by which all of the above quantities are propagated from time t_k (which may or may not have been a measurement time) to the next measurement time t_{k+1} and updated after the measurement is processed. We assume no events occur between t_k and t_{k+1}.

Before proceeding with a step-by-step discussion of this basic cycle, it is worthwhile to point out that, unlike the error analysis program, the simulation program is involved in actually processing data to estimate an interplanetary or lunar trajectory. Some "actual" trajectory is being flown within the computer and simulated measurements from Earth-based tracking stations are recorded, based upon this "actual" trajectory. These measurements are then processed in a recursive estimation algorithm; thus the simulation program provides a check of the orbit determination procedure's ability to reproduce the "actual" trajectory under a wide set of conditions that might be anticipated on an actual mission.

The structure of the basic cycle in the simulation program is depicted in figure 6.1 and shows the basic division within the simulation program between the modeled world and the real world. The modeled world consists of equations and parameters which are assumed to describe the motion of the spacecraft and the generation of observations. It also consists of assumed spacecraft injection statistics and assumed measurement noise statistics. Of necessity the modeled world always differs from the real world. The actual equations governing the spacecraft motion are never completely known. Dynamic and measurement parameters can never be known exactly. Non-zero injection errors always occur. Actual measurement noise statistics can only be approximated. The whole purpose of the simulation program is to provide a tool to study the effects of modeled world/real world differences on the navigation process. The same philosophy underlies the treatment of the guidance event in the simulation program (see chapter 7).

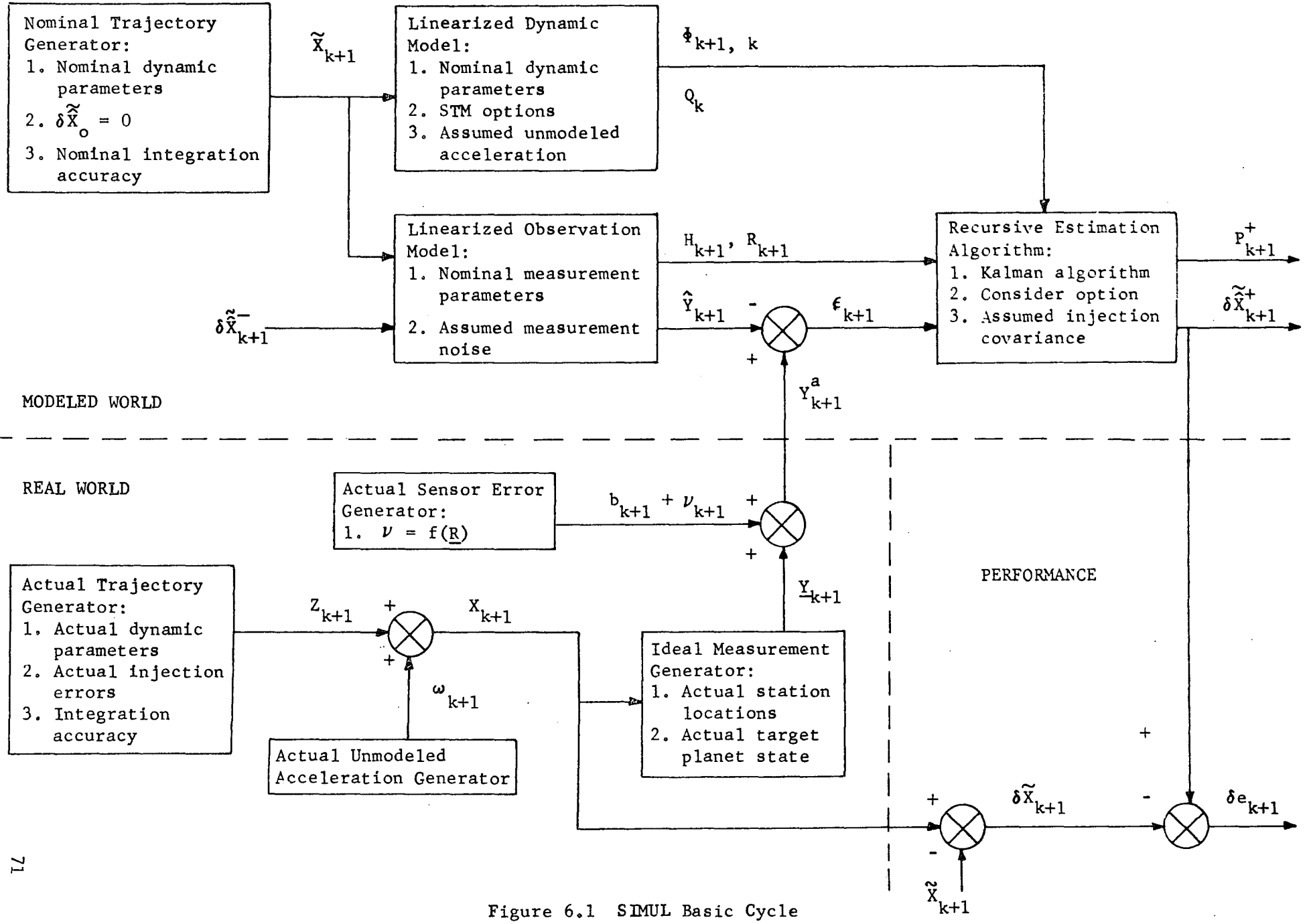


Figure 6.1 SIMUL Basic Cycle

Figure 6.1 will be used as a guide in the following discussion of the computational flow in the basic cycle. It should be pointed out that figure 6.1 has been simplified somewhat for the sake of clarity and does not show all the details of the basic cycle. For example, only the first member of all partitioned matrices is shown. We assume all required quantities are available at time t_k and that the next measurement occurs at time t_{k+1}. The first step in the basic cycle is to propagate the most recent nominal spacecraft state forward to t_{k+1} using assumed dynamic parameters and a nominal integration accuracy. Although not shown in figure 6.1, the targeted spacraft state is also propagated forward to t_{k+1} in the same way.

The next steps in the SIMUL basic cycle are the same as those in the basic cycle of the error analysis program. State transition and observation matrix partitions are computed, as are the assumed dynamic noise and measurement noise covariance matrices. Next, the knowledge covariance prediction and filtering equations given in section 5.2 are evaluated to determine the reduction in all knowledge covariance matrix partitions following the processing of the measurement. It should be pointed out that the measurement residual covariance matrix J_{k+1}, defined by equation (5.13), is very important in what is known as adaptive filtering, a topic to be discussed in section 6.4 of this chapter. An extension of the current program to permit the use of adaptive filtering would not be difficult.

We turn next to the generation of the actual spacecraft state at time t_{k+1}. The same virtual mass subroutines are used to compute the actual trajectory, except now different dynamic parameters are used. For example, different values for the gravitational constants of the Sun and the target planet could be used. The target planet ephemerides could be changed. Also, a different number of celestial bodies could be used in the generation of the actual trajectory, as well as a different integration accuracy. Returning from the virtual mass subroutines, we have available a quantity Z_{k+1}, which would be the actual state at t_{k+1} if there were no unmodeled accelerations acting on the spacecraft over the time interval $[t_k, t_{k+1}]$. However, the simulation program permits constant unmodeled accelerations to act on the spacecraft over this time interval which integrate into a state vector addition ω_{k+1}. The actual state and the actual state deviation from the most recent nominal are then given by

$$X_{k+1} = Z_{k+1} + \omega_{k+1} \tag{6.1}$$

$$\delta\tilde{X}_{k+1} = X_{k+1} - \hat{X}_{k+1} \tag{6.2}$$

Unmodeled accelerations are permitted to corrupt the actual state vector for a very definite purpose. Along an interplanetary flight, many possible sources of mechanical difficulty onboard the vehicle could give rise to small accelerations. It is important, for the purposes of the simulation, to determine how the orbit determination algorithm reacts in the presence of small accelerations about which the algorithm itself has no specific knowledge.

Another digression concerning the underlying philosophy of the simulation program is now warranted. Recall that its purpose is essentially to test a specific navigation and guidance process, insofar as is possible, under real conditions. Four key assumed statistical descriptions are used by the estimation algorithm to produce the optimal estimate of the state vector. These four are in injection covariance P_0, the dynamic noise covariance Q_k, the measurement noise covariance R_k, and the midcourse correction execution error covariances $\tilde{Q}_j$. All of the matrices represent assumed errors and their probabilistic descriptions. Obviously the convergence of the estimated trajectory to the actual trajectory, for a real flight, is a function of the accuracy of these a priori statistics. To test the orbit determination and guidance process within the simulation program, actual injection errors δX_0, actual midcourse execution errors, actual unmodeled accelerations, and actual measurement noise statistics $\underline{R}_k$ may be specified by the user. These specifications permit the study of the effect of bad a priori statistics on the success of the defined navigation and guidance algorithms.

The next step in the SIMUL basic cycle is concerned with the generation of the actual measurement Y^a_{k+1}. Referring to figure 6.1, it is apparent that the actual measurement provides the recursive estimation algorithm's only contact with the real world. All other inputs to the estimation algorithm are computed on the basis of the modeled world. To compute the actual measurement we first determine the ideal measurement $\underline{Y}_{k+1}$ which would be made for the actual spacecraft state in the absence of all sensor errors. The equation defining $\underline{Y}_{k+1}$ has form

$$\underline{Y}_{k+1} = F(X_{k+1}, p, t_{k+1}) \tag{6.3}$$

where X_{k+1} is the actual spacecraft state and p denotes a vector of actual dynamic and measurement parameters. Then, assuming the actual sensor error to be the sum of a bias b_{k+1} and a random noise ν_{k+1}, we compute the actual measurement from

$$Y^a_{k+1} = \underline{Y}_{k+1} + b_{k+1} + \nu_{k+1} \tag{6.4}$$

The random noise ν_{k+1} is obtained by randomly sampling the actual measurement noise covariance matrix $\underline{R}_{k+1}$.

The estimation algorithm operates on the measurement residual ϵ_{k+1}, where ϵ_{k+1} is defined as

$$\epsilon_{k+1} = Y^a_{k+1} - \hat{Y}_{k+1} \tag{6.5}$$

and $\hat{Y}_{k+1}$ is the estimated value of the measurement. To compute $\hat{Y}_{k+1}$ we first determine the ideal measurement $\tilde{Y}_{k+1}$ which would be made if the spacecraft were actually located at the most recent nominal state $\tilde{X}_{k+1}$. The equation defining $\tilde{Y}_{k+1}$ has form

$$\tilde{Y}_{k+1} = F(\tilde{X}_{k+1}, \tilde{p}, t_{k+1}) \tag{6.6}$$

where $\tilde{p}$ denotes a vector of assumed or nominal dynamic and measurement parameters. Of course, the estimated spacecraft state does not, in general, coincide with $\tilde{X}_{k+1}$. Further, some of the parameters are solve-for parameters which are being estimated along with the spacecraft state. The estimated measurement $\hat{Y}_{k+1}$ reflects these estimated deviations and is given by

$$\hat{Y}_{k+1} = \tilde{Y}_{k+1} + \delta\tilde{\hat{Y}}_{k+1} \tag{6.7}$$

where

$$\delta\tilde{\hat{Y}}_{k+1} = H_{k+1}\ \delta\tilde{\hat{X}}^-_{k+1} + M_{k+1}\ \delta\tilde{\hat{X}}^-_{s_{k+1}} \tag{6.8}$$

Observation matrices H_{k+1} and M_{k+1} are defined in equation (5.3). Position/velocity deviation estimate $\delta\tilde{\hat{X}}^{-}_{k+1}$ and solve-for parameter deviation estimate $\delta\tilde{\hat{X}}^{-}_{s_{k+1}}$ are computed using the prediction equations

$$\delta\tilde{\hat{X}}^{-}_{k+1} = \Phi\, \delta\tilde{\hat{X}}^{+}_{k} + \theta_{xx_s}\, \delta\tilde{\hat{X}}^{+}_{s_k} \tag{6.9}$$

$$\delta\tilde{\hat{X}}^{-}_{s_{k+1}} = \delta\tilde{\hat{X}}^{+}_{s_k} \tag{6.10}$$

The recursive estimation algorithm now has available all the information required to process the measurement and generate new estimates. The Kalman recursive estimation equations are given by

$$\delta\tilde{\hat{X}}^{+}_{k+1} = \delta\tilde{\hat{X}}^{-}_{k+1} + K_{k+1}\, \epsilon_{k+1} \tag{6.11}$$

$$\delta\tilde{\hat{X}}^{+}_{s_{k+1}} = \delta\tilde{\hat{X}}^{-}_{s_{k+1}} + S_{k+1}\, \epsilon_{k+1} \tag{6.12}$$

where Kalman gain constants K_{k+1} and S_{k+1} have been computed previously using equations (5.14) and (5.15).

The SIMUL basic cycle is complete with the computation of the actual estimation error δe_{k+1}. This error is defined as

$$\delta e_{k+1} = \delta\tilde{\hat{X}}^{+}_{k+1} - \delta\tilde{X}_{k+1} \tag{6.13}$$

and is a measure of the performance of the recursive estimation algorithm. A similar error quantity is defined for the solve-for parameter estimate. Divergence is said to occur when these estimation errors grow in an unbounded fashion.

6.3 Eigenvector, Prediction, and Quasi-Linear Filtering Events

Eigenvector and prediction events in the simulation program involve all the computations which are performed in eigenvector and prediction

events in the error analysis program (see section 5.6). In addition, the most recent nominal $\tilde{X}_k$, the actual state deviation $\delta\tilde{X}_k$, and the estimated deviation $\delta\tilde{\hat{X}}_k$ and $\delta\tilde{\hat{X}}_{s_k}$ are propagated forward from time t_k to event time t_j. These operations are performed by using equations (6.1), (6.2), (6.9), and (6.10) with $t_{k+1} = t_j$. Also, covariance matrix partitions are propagated along the most recent nominal, rather than along the targeted nominal as in ERRAN.

The prediction event, as it is treated in SIMUL, differs from an ERRAN prediction event in one other respect. In addition to propagating the knowledge covariance matrix partitions forward to time t_p, the time to which we wish to predict, the estimated deviations $\delta\tilde{\hat{X}}_j$ and $\delta\tilde{\hat{X}}_{s_j}$ are also propagated forward to t_p. Equations (6.9) and (6.10) are used for this purpose with $t_{k+1} = t_p$.

At a quasi-linear filtering event the most recent nominal trajectory is updated by using the most recent estimate. The purpose of the update is to combat divergence due to the possible invalidity of the linearity assumption that is the basis for the estimation algorithm being used. Specifically, updating the nominal trajectory results in better computations of the state transition and observation matrix partitions employed in the recursive estimation algorithm. The quasi-linear filtering event is defined only in the simulation program since actual state deviation estimates must be available.

Letting t_j denote the time of the quasi-linear filtering event, and using $(\ )^-$ and $(\ )^+$ notation to indicate values immediately before and after the event, respectively, the primary state vectors and deviations in the simulation program are updated as follows:

$$\begin{aligned}
\tilde{X}_j^+ &= \tilde{X}_j^- + \delta\tilde{\hat{X}}_j^- \\
\delta\tilde{\hat{X}}_j^+ &= 0 \\
\delta\tilde{X}_j^+ &= \delta\tilde{X}_j^- - \delta\tilde{\hat{X}}_j^- \\
\bar{X}_j^+ &= \bar{X}_j^-
\end{aligned} \tag{6.14}$$

Knowledge covariance matrix partitions do not change across a quasi-linear filtering event since nothing has occurred which would either increase or decrease the uncertainty of our estimates. Control covariance matrices, which are defined in chapter 7, likewise do not change across a quasi-linear filtering event. Finally, nominal solve-for parameter state vectors are not updated at a quasi-linear filtering event since all parameters will be reasonably well known initially.

6.4 Divergence and Other Problems

One of the purposes of creating such a detailed and extensive simulation program was to study the problem of filter divergence. The problem of divergence in a recursive navigation process and a companion difficulty, computational nonobservability, are the subjects of this section.

Strictly speaking, when divergence occurs in a navigation process, the navigation is failing to navigate properly. The phenomenon of divergence never appears in an error analysis program because no actual estimation is taking place and only covariance matrices are being propagated. In a computer simulation such as the STEAP simulation program, where an "actual" trajectory is being flown and concurrently estimated by a navigation algorithm, filter divergence refers to the failure of the estimated trajectory to converge, within reasonable bounds specified by the covariance matrices, to the "actual" simulated trajectory. For real-world orbit applications, where the actual trajectory is never known, divergence is occurring when the residual difference between predicted and actual observation vectors becomes increasingly large.

In either computer simulation or a real orbit determination procedure, divergence in the recursive filter manifests itself as a statistical inconsistency between the measurement residuals and the filtering algorithm. Recall that at each step of the recursive process, the matrix J_{k+1} is computed (see equation (5.13)). This matrix defines the a priori statistics associated with the measurement residual ϵ_{k+1}. The measurement residual ϵ_{k+1} should represent a sample for the population defined by J_{k+1}. When divergence occurs, a group of successive residuals appear less and less likely, statistically, to have been sampled from their covariances J_{k+1}.

To illustrate the divergence manifestation in terms of measurement residuals inconsistent with their a priori covariances, assume that a scalar range-rate measurement is being taken along an interplanetary orbit. For scalar measurements, the matrix J_{k+1} is a scalar residual variance, call it σ_ϵ^2. Suppose that for the first two hundred measurements, each measurement residual was compared to its statistical variance by solving

$$\epsilon_j = K_j \sigma_{\epsilon_j}, \quad j = 1, 2, \ldots, 100 \qquad (6.15)$$

for the value K_j. Suppose further that a frequency histogram of the values K_j produced

Interval	No.	Percent	Theoretical Percent
$-\frac{1}{2} \leq K < \frac{1}{2}$	79	39.5	38.3
$-\frac{3}{2} \leq K < -\frac{1}{2}$	55	27.5	24.17
$-\frac{5}{2} \leq K < -\frac{3}{2}$	9	4.5	6.06
$K < -\frac{5}{2}$	4	2.0	0.62
$+\frac{1}{2} \leq K < \frac{3}{2}$	38	19.0	24.17
$\frac{3}{2} \leq K < \frac{5}{2}$	13	6.5	6.06
$K < \frac{5}{2}$	2	1.0	0.62

Without subjecting the above data to a rigorous chi-square test of hypothesis, it should be clear that the measurement residuals, each of which is assumed to be an uncorrelated, Gaussian, mean zero random variable, are more or less consistent with their statistics; that is, the ensemble values for the measurement residuals look reasonable in terms of their a priori variances used within the estimation algorithm.

Now suppose that the next nine values of K_j, determined in the same fashion from the nine measurements following the two hundredth, are given by the sequence

$$K_{201} = -2.4 \quad K_{204} = -4.7 \quad K_{207} = -6.2$$

$$K_{202} = -3.7 \quad K_{205} = -5.1 \quad K_{208} = -7.1 \qquad (6.16)$$

$$K_{203} = -4.4 \quad K_{206} = -5.5 \quad K_{209} = -7.8$$

From the underlying assumptions of the navigation process, each of these events, taken singly, is extremely unlikely. However, the sequence of values given is almost totally unlikely and should represent a dead giveaway that divergence is occurring. Without pursuing the mathematics too far, it should be stressed that if the nine values given in equation (6.16) were supposedly chosen at random from a normal distribution with near zero and unit variance, the governing distribution would fail every test of statistical hypothesis. Such values for K_j indicate that something in the estimation process is definitely wrong: the most likely candidate for the error is the assumed a priori J_{k+1} matrix used for weighting by the estimation algorithm.

The hypothetical example given above is typical of the divergence phenomenon that recurs in complex orbit determination processes. Often, the process converges initially and then, after many measurements have been taken, divergence begins. A general explanation for this is that the covariance matrices associated with the estimated state vector become overly optimistic and, subsequently, tend to disregard the new measurement data in the weighting process.

The general cause of divergence is modeling insufficiency. For most real problems, everything about the dynamical system and the observations being treated by the filter is not known exactly. Unless the estimation algorithm acknowledges, in some fashion, the incomplete understanding of the governing equations, divergence may result. A familiar source of insufficient modeling is the dynamic equations themselves. All the forces acting on spacecraft are never known exactly. In addition, the filtering algorithm operates on perturbation equations resulting from a linearization about some reference dynamic state. Thus, the procedure is working with approximate equations and unless dynamic noise is added to the computational algorithm, the Kalman filter "thinks" it knows the exact equations of motion, whereas in reality it does not.

Divergence can also result from other model inadequacies. Among the most frequent causes are failure to account for measurement nonlinearities when the measurements themselves are very accurate, neglect of correlated errors between sequences of measurements taken by the same instruments, and overly optimistic a priori error statistics describing

the measurement noise. Within the simulation program of STEAP, the effect of all these model inadequacies on a specific reference trajectory can be tested.

Many possible solutions to the problem of divergence have been postulated and investigated. Four of the methods of divergence prevention will be discussed in this section. These methods are (1) dynamic noise covariance modeling, (2) quasi-linear filtering, (3) consider mode filtering, and (4) adaptive filtering. All but the last method are available in the simulation program.

In an earlier section of this report the modeling of a dynamic noise matrix Q_k was discussed. Between measurements in the estimation algorithm of STEAP, the state vector associated covariance matrix is propagated according to equation (5.6). The Q matrix appearing in this equation increases the magnitude of the key diagonal elements in the covariance matrix. Because divergence generally occurs when the state vector associated covariances become unduly optimistic and additional measurements are weighted very slightly, the addition of Q represents an attempt to systematically downgrade the dynamics information in favor of the measurements.

Although the addition of a proper Q matrix will impede divergence, unless its size is determined by physical considerations it can also slow convergence. Most often the Q matrix is somewhat arbitrary and its exact influence on the estimation algorithm is not clearly understood. Thus, attempts should be made, based on the modeling for a particular problem, to ensure that the elements of Q are realistic.

A second method of divergence prevention included in the simulation program is the method of quasi-linear filtering. The fundamental estimation process assumes that variations about the nominal trajectory and nominal measurements are linear. In the case of highly accurate measurements, which is usually the case of interplanetary spacecraft tracked by the DSIF, measurement nonlinearities become significant model inadequacies when the actual trajectory is only slightly different from the nominal. Quasi-linear filtering essentially permits more accurate computation of state transition and observation matrices. This is accomplished by updating the most recent nominal trajectory, based on the estimated state vectors coming from the navigation algorithm, and then computing both the state transition and observation matrices in terms of linear perturbations about the updated nominal.

Treatment of errors in dynamic or measurement parameters without actually estimating them is generally called consider mode filtering. This is the third method of divergence prevention available in STEAP.

Consider mode filtering acts to prevent divergence in much the same way as does dynamic noise covariance modeling, discussed earlier. The consider parameter covariance matrices are not influenced by measurement processing and, in fact, remain constant. Since consider parameter covariance matrices cannot be reduced, the effect of consider mode filtering is to create a residue of uncertainty which can never be eliminated from the assumed dynamic model. This, in essence, defines a lower bound for the position/velocity covariance matrix.

Of the other methods for handling filter divergence that have been suggested in the literature, the strongest appears to be adaptive filtering. Reference 7 explains the theoretical basis for several kinds of adaptive filtering schemes. The essential idea of adaptive filtering is the feedback of actual measurement residuals into the covariance matrix propagation process. Earlier it was pointed out that a sign of filter divergence is a statistical inconsistency between the measurement residuals ϵ_{k+1} and their assumed a priori covariance matrices J_{k+1} used by the estimation algorithm. In adaptive filtering, this statistical inconsistency is used to change the assumed a priori statistics, on both the dynamics and the measurements until the residuals and their updated covariances are more or less consistent. Optimal implementation of adaptive filtering is being pursued by several researchers in the field.

Another problem associated with interplanetary orbit determination that can be studied with the STEAP simulation mode is that of computational nonobservability. Because this problem threatens to occur whenever strictly Earth-based tracking is being used to determine the orbit of a spacecraft around the Moon or another planet, it warrants attention.

In classical batch-processing algorithms, observability does not exist when a key matrix inverse used to determine the estimate does not exist. In a recursive algorithm, nonobservability manifests itself when one of the correlation coefficients relating uncertainties in different elements of the state vector has unit magnitude. Physically this means that the navigation process cannot observe or estimate the two quantities that are either positively or negatively correlated uniquely. The orbit determination procedure has no unique convergence in this case.

When the correlation coefficients relating uncertainties in two elements of the state vector are very close to unity in magnitude, then the underlying estimation algorithm is very unstable. Although theoretically a unique solution still exists, any model inadequacies can produce wild gyrations in the estimated solutions. Preliminary studies with STEAP of orbit determination processes using Earth-based tracking for spacecraft in Moon or Mars orbits indicate that the above orbit determination instability, called computational nonobservability, is very much a real problem.

7. GUIDANCE ANALYSIS

7.1 Introduction

Of the many types of events available in the STEAP programs, guidance events are by far the most complex. The purpose of Chapter 7 is to provide a comprehensive and unified discussion of the analytical basis for all types of guidance events modeled in the error analysis program ERRAN and the simulation program SIMUL.

Guidance events yield much useful information for preflight mission analysis. Using ERRAN we can evaluate, in a statistical sense, the efficacy of the guidance process in achieving desired target conditions. Equally important is the determination of the statistical ΔV requirements for the mission. Using SIMUL we can determine the effect of modeled world/real world differences on the guidance process. Actual ΔV requirements and actual target errors can be computed for the simulated mission. In both ERRAN and SIMUL the coupling of the guidance and navigation processes has been carefully modeled.

Several types of guidance events are available in ERRAN and SIMUL. At a midcourse guidance event the user can choose from three midcourse guidance policies: fixed-time-of-arrival, two-variable B-plane, and three-variable B-plane. The midcourse guidance event can be subjected to planetary quarantine constraints. Two orbital insertion policies are available: coplanar insertion and non-planar insertion. Options are also available for changing target conditions in mid-flight and re-targeting the trajectory using nonlinear techniques, or for simply applying an externally-supplied or pre-computed ΔV at some arbitrary time along the trajectory. Two thrust models are available: impulse and impulse series.

In the following section the concept of control covariance will be presented, and all features of the guidance event which are independent of the specific guidance policy will be discussed. Section 7.3 treats the execution error model employed for impulsive ΔV's. Section 7.4 treats both linear and nonlinear midcourse guidance, as well as biased aimpoint guidance, which is required to satisfy planetary quarantine constraints. All remaining guidance event options are discussed in section 7.5

7.2 General Analysis

Most variables used in the general analysis have already been defined in Chapters 5 and 6. We shall assume an arbitrary guidance event is to be executed at guidance event time t_j. In the following analysis the notation $(\ )_j^-$ will be used to indicate the values of variables immediately prior to the execution of the event; $(\ )_j^+$, immediately after. Although denoted simply by P in Chapters 5 and 6, the knowledge covariance will now be denoted by P_K to distinguish it from the control covariance P_c.

Only the spacecraft position/velocity knowledge and control covariance partitions are required for guidance analysis, although the entire set of covariance prediction equations given in section 5.2 are used whenever covariances matrices are to be propagated over some interval of time.

Before proceeding with the general analysis of a guidance event, it is necessary to digress briefly to discuss the control covariance P_C and how it differs from knowledge covariance P_K. Recall that the knowledge covariance represents the statistical dispersions of the estimation errors about the spacecraft state estimate and is defined as

$$P_K = E\left[\delta e\ \delta e^T\right] \tag{7.1}$$

where estimation error δe is defined as

$$\delta e = \delta\tilde{\hat{X}} - \delta\tilde{X} . \tag{7.2}$$

Here $\delta\tilde{\hat{X}}$ and $\delta\tilde{X}$ denote the estimated and actual deviations, respectively, from the most recent nominal trajectory. Processing of measurements normally reduces the knowledge covariance, which, in geometrical terms, corresponds to a contraction of the knowledge covariance hyperellipsoid. The control covariance represents the statistical dispersions of the actual trajectory about the targeted nominal trajectory and is defined as

$$P_C = E\left[\delta X\ \delta X^T\right] \tag{7.3}$$

where δX denotes the actual deviation from the targeted nominal trajectory. The time behavior of the control covariance depends solely on modeled spacecraft dynamics and is in no way (except at a guidance event) influenced by measurement processing. Control covariances, like knowledge covariances, are propagated across an interval of time using the covariance prediction equations given in section 5.2. However, the covariance filtering equations, which are also presented in section 5.2, are never used to update control covariances. Control covarisnces are used in both ERRAN and SIMUL to predict statistical target miss dispersions. The control covariance is also important in the computation of statistical midcourse guidance corrections in ERRAN.

We return now to the discussion of a general guidance event. At each guidance event a commanded velocity correction $\Delta\hat{V}_j$ is computed. The nature of this computation is, of course, policy-dependent and will be treated in subsequent sections. In general, $\Delta\hat{V}_j$ will be a function

of the desired target conditions and the estimated spacecraft state. Except when midcourse guidance corrections are treated in an ensemble sense, as they are in ERRAN, the required $\Delta\hat{V}_j$ can always be computed. Otherwise, only the statistical "$E\left[\Delta\hat{V}_j\right]$" can be computed.

Due to execution errors the actual velocity correction will differ from the commanded correction. The actual velocity correction ΔV_j is given by

$$\Delta V_j = \Delta\hat{V}_j + \delta\Delta V_j \tag{7.4}$$

where $\delta\Delta V_j$ is the execution error. The guidance process acknowledges the existence of an execution error by generating the assumed statistics of the execution error. The execution error is assumed to have zero mean and covariance $\tilde{Q}_j$, which is defined as

$$\tilde{Q}_j = E\left[\delta\Delta V_j \; \delta\Delta V_j^T\right] \tag{7.5}$$

Both $\delta\Delta V_j$ and $\tilde{Q}_j$ are generated using the execution error model described in section 7.3.

The spacecraft state, estimated deviations, and covariance matrices are altered when a guidance event is executed. The remainder of this section develops all the equations required in this updating process for an impulsive velocity correction. If the velocity correction is modeled as an impulse series the modified equations presented in section 7.5.4 define the updating process.

In addition to the assumption of an impulsive velocity correction, we also assume that the targeted nominal trajectory is updated after a guidance correction. This is a reasonable assumption and permits the simplification of the control covariance update equation. The targeted nominal state update equation is given by

$$\bar{X}_j^+ = \bar{X}_j^- + \delta\tilde{\hat{X}}_j^- + \begin{bmatrix} - \\ \underline{0} \\ - \\ \Delta\hat{V}_j \end{bmatrix} \tag{7.6}$$

where $\bar{X}_j^- + \delta\tilde{\hat{X}}_j^-$ is the estimated spacecraft state just prior to the guidance event. Note that equation (7.6) also defines the estimated state immediately following the guidance correction. Thus, if we also update the most recent nominal state using equation (7.6), then the estimated state deviation is given by

$$\delta\tilde{\hat{X}}_j^+ = 0 . \tag{7.7}$$

At a guidance event our estimation error δe is increased by the execution error. Thus

$$\delta e_j^+ = \delta e_j^- - \begin{bmatrix} - \\ \delta\Delta V_j \end{bmatrix} \tag{7.8}$$

where

$$\delta e_j^- = \delta\tilde{\hat{X}}_j^- - \delta\tilde{X}_j^- . \tag{7.9}$$

The minus sign appears in equation (7.8) since, according to equation (7.4), $\delta\Delta V_j$ is defined as the actual minus the estimate, while the estimation error δe is defined as the estimate minus the actual. Then the new actual state deviation from the most recent nominal is defined by

$$\delta\tilde{X}_j^+ = \delta\tilde{\hat{X}}_j^+ - \delta e_j^+ \tag{7.10}$$

Combining equations (7.7) through (7.10) to eliminate all $(\ )^+$ quantities in equation (7.10) yields

$$\delta\tilde{X}_j^+ = \delta\tilde{X}_j^- - \delta\tilde{\hat{X}}_j^- + \begin{bmatrix} - \\ \delta\Delta V_j \end{bmatrix} \tag{7.11}$$

This is the actual state deviation update equation.

It remains to develop the update equations for the knowledge and control covariance matrices. The knowledge covariance immediately following the guidance correction is defined by

$$P_{K_j}^+ = E\left[\delta e_j^+ \, \delta e_j^{+T}\right] \tag{7.12}$$

Substitution of equation (7.8) into equation (7.12) readily yields the required knowledge covariance update equation:

$$P_{K_j}^+ = P_{K_j}^- + \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \tilde{Q}_j \end{array}\right] \tag{7.13}$$

The control covariance update equation is also easily derived. Substitution of equation (7.7) into equation (7.10) yields

$$\delta\tilde{X}_j^+ = -\delta e_j^+ \tag{7.14}$$

But the control covariance immediately following the guidance equation is defined by

$$P_{c_j}^+ = E\left[\delta\tilde{X}_j^+ \; \delta\tilde{X}_j^{+\,T}\right] \tag{7.15}$$

Substitution of equation (7.14) into equation (7.15) then yields the control covariance update equation

$$P_{c_j}^+ = P_{K_j}^+ \tag{7.16}$$

All the update equations just presented are used in the simulation program SIMUL. The error analysis program ERRAN treats only the ensemble characteristics of the navigation and guidance processes and, as such, does not generate estimated or actual state deviations. Thus, the state deviation update equations are undefined in ERRAN. Only $\bar{X}$, P_K, and P_c are updated at an ERRAN guidance event. The latter two quantities are updated using equations (7.13) and (7.16), respectively, while the targeted nominal is updated using

$$\bar{X}_j^+ = \bar{X}_j^- + \begin{bmatrix} 0 \\ - \\ \Delta\hat{V}_{UP_j} \end{bmatrix} \tag{7.17}$$

where $\Delta\hat{V}_{UP_j}$ represents the non-statistical component of $\Delta\hat{V}_j$. When $\Delta\hat{V}_j$ is wholly statistical, as is the case for ERRAN midcourse guidance events (with no aimpoint biasing), then of course $\Delta\hat{V}_{UP_j}$ is zero and the targeted nominal is not updated.

7.3 Execution Error Model

The computation of the actual execution error $\delta\Delta V$ and the execution error covariance matrix $\tilde{Q}$ is based on an execution error model defined by four independent error sources. The first error source is called the proportionality error and is in the direction of the velocity correction

vector ΔV with magnitude determined by the proportionality factor k. A second error source, in the direction of ΔV but independent of its magnitude, is the resolution error s that corresponds to a thrust tailoff error from the engines. Two pointing errors defined in terms of angles $\delta\alpha$ and $\delta\beta$ complete the error model. From this description of the error model, the equation for $\delta\Delta V$ can be written as

$$\delta\Delta V = k\,\Delta V + s\,\frac{\Delta V}{|\Delta V|} + \delta\Delta V_{pointing} \qquad (7.18)$$

where $\delta\Delta V_{pointing}$ is defined by two angular pointing errors, $\delta\alpha$ and $\delta\beta$.

For purposes of unique specification, assume that $\delta\alpha$ is a pointing error angle measured in a plane parallel to the ecliptic plane and along a vector orthogonal to the velocity correction vector ΔV. If $\delta\Delta V_1$ is the velocity error due to the angular pointing error $\delta\alpha$ and $\hat{i}$, $\hat{j}$, $\hat{k}$ form the unit triad in the ecliptic system, then for small angles $\delta\alpha$, $\delta\Delta V_1$ is given by

$$\delta\Delta V_1 = \rho\delta\alpha\left[\frac{\Delta V_Y}{(\Delta V_X^2 + \Delta V_Y^2)^{\frac{1}{2}}}\hat{i} - \frac{\Delta V_X}{(\Delta V_X^2 + \Delta V_Y^2)^{\frac{1}{2}}}\hat{j}\right] \qquad (7.19)$$

where ΔV_X and ΔV_Y are the X and Y ecliptic components of the velocity correction vector ΔV and ρ is the magnitude of ΔV . Note that the velocity error $\delta\Delta V_1$ resulting from $\delta\alpha$ has components only in a plane parallel to the ecliptic.

The second pointing angle $\delta\beta$ defines a velocity error $\delta\Delta V_2$ that is orthogonal to both $\delta\Delta V_1$ and the velocity correction vector ΔV . Again for small angles $\delta\beta$, the velocity error resulting from this pointing error, referenced to the ecliptic system, is given by

$$\delta\Delta V_2 = \frac{\Delta V_X \Delta V_Z \delta\beta}{(\Delta V_X^2 + \Delta V_Y^2)^{\frac{1}{2}}}\hat{i} + \frac{\Delta V_Y \Delta V_Z \delta\beta}{(\Delta V_X^2 + \Delta V_Y^2)^{\frac{1}{2}}}\hat{j} - \delta\beta(\Delta V_X^2 + \Delta V_Y^2)^{\frac{1}{2}}\hat{k} \qquad (7.20)$$

From these equations it is clear that the vector set ΔV, $\delta\Delta V_1$ and $\delta\Delta V_2$ satisfies the mutual orthogonality imposed by the model. The complete description of the execution error vector $\delta\Delta V$ may then be written in ecliptic coordinates as

$$\delta\Delta V = \left[\left(k + \frac{s}{\rho}\right)\Delta V_X + \frac{\rho\Delta V_Y\,\delta\alpha + \Delta V_X\,\Delta V_Z\,\delta\beta}{u}\right]\hat{i}$$
$$+ \left[\left(k + \frac{s}{\rho}\right)\Delta V_Y + \frac{\Delta V_Y\,\Delta V_Z\,\delta\beta - \rho\,\Delta V_X\,\delta\alpha}{u}\right]\hat{j}$$
$$+ \left[\left(k + \frac{s}{\rho}\right)\Delta V_Z - u\,\delta\beta\right]\hat{k} \qquad (7.21)$$

where ΔV_X, ΔV_Y, and ΔV_Z are the ecliptic coordinates of the velocity correction; $\hat{i}$, $\hat{j}$, $\hat{k}$ are unit vectors in the X, Y, and Z directions; ρ is the magnitude of ΔV; k, s, $\delta\alpha$, $\delta\beta$ are the four independent error sources; and u is an intermediate variable defined by

$$u = (\Delta V_X^2 + \Delta V_Y^2)^{\frac{1}{2}} \qquad (7.22)$$

In SIMUL the components of the commanded velocity correction $\Delta\hat{V}_j$ and actual values of the errors k, s, $\delta\alpha$, and $\delta\beta$ are used to evaluate equation (7.21) to determine the actual execution error $\delta\Delta V_j$. Equation (7.21) is not required in ERRAN.

The expression for the execution error covariance $\tilde{Q}_j$ is obtained by substituting equation (7.21) into equation (7.5). The equations which result from this operation are summarized in the subroutine QCØMP analysis and will not be presented here. However, these equations have form given by

$$\tilde{Q}_j = \tilde{Q}_j(\Delta V, \sigma_k^2, \sigma_s^2, \sigma_{\delta\alpha}^2, \sigma_{\delta\beta}^2) \qquad (7.23)$$

where σ_k^2 through $\sigma_{\delta\beta}^2$ are the assumed variances for the four error sources which define the error model. No cross-correlations appear in this equation since all the error sources are assumed to be independent.

In SIMUL the components of the commanded velocity correction $\Delta\hat{V}_j$ are used to evaluate equation (7.23). In ERRAN the sum of the statistical and non-statistical components of $\Delta\hat{V}_j$ are used.

7.4 Midcourse Guidance

7.4.1 Linear Midcourse Guidance

Linear impulsive midcourse guidance policies have form

$$\Delta\hat{V}_j = \Gamma_j \, \delta\hat{X}_j^- \tag{7.24}$$

where $\Delta\hat{V}_j$ is the commanded velocity correction required to null out deviations from the nominal target state, Γ_j is the guidance matrix, and $\delta\hat{X}_j^-$ is the estimated spacecraft deviation from the targeted nominal trajectory just prior to the guidance correction.

Three midcourse guidance policies are modeled in ERRAN and SIMUL: fixed-time-of-arrival (FTA), two-variable B-plane (2VBP), and three-variable B-plane (3VBP). The derivation of the Γ_j matrix for each policy will be summarized below.

The variation matrix η_j relates deviations in spacecraft state at t_j to target state deviations. If τ is a vector which defines the target state, then

$$\delta\tau = \eta_j \, \delta X_j \tag{7.25}$$

For FTA guidance the target state τ is the nominal closest approach position vector R_{CA} at nominal time of closest approach t_{CA}. State deviations at t_j are related to state deviations at t_{CA} by the equation

$$\delta X_{CA} = \Phi(t_{CA}, t_j) \, \delta X_j \tag{7.26}$$

where $\Phi(t_{CA}, t_j)$ is the state transition matrix over the interval $[t_j, t_{CA}]$. Thus, for FTA guidance the variation matrix is given by

$$\eta_j = \left[\phi_1 \;\vdots\; \phi_2 \right] \tag{7.27}$$

where ϕ_1 and ϕ_2 denote the two upper 3x3 partitions of Φ. We wish to select a $\Delta\hat{V}_j$ such that $\delta\tau = 0$ in equation (7.25). Employing equation (7.27), this condition reduces to the equation

$$0 = \left[\phi_1 \;\vdots\; \phi_2 \right] \left(\delta\hat{X}_j + \begin{bmatrix} - & 0 & - \\ & \Delta\hat{V}_j & \end{bmatrix} \right)$$

which, when solved for $\Delta\hat{V}_j$, yields

$$\Delta\hat{V}_j = \left[-\phi_2^{-1}\phi_1 \;\vdots\; -I \right] \delta\hat{X}_j \tag{7.28}$$

and

$$\Gamma_{FTA} = \left[-\phi_2^{-1}\phi_1 \;\vdots\; -I \right] . \tag{7.29}$$

The target state for 3VBP guidance is defined as $\tau = \left[B\cdot T,\ B\cdot R,\ t_{SI} \right]^T$, where t_{SI} is the time at which the nominal trajectory pierces the sphere of influence of the target planet. The variation matrix η_j cannot be easily defined in terms of a state transition matrix. Instead, a numerical differencing technique, which is described in the subroutine VARSIM analysis section, must be employed to construct the η_j matrix. Once the η_j matrix is available we can write

$$\eta_j = \left[\eta_1 \;\vdots\; \eta_2 \right] \tag{7.30}$$

and proceed along the lines of the previous FTA derivation to obtain

$$\Gamma_{3VBP} = \left[-\eta_2^{-1}\eta_1 \;\vdots\; -I \right] . \tag{7.31}$$

The target state for 2VBP guidance is defined as $\tau = \left[B\cdot T,\ B\cdot R \right]^T$, where, in contrast to 3VBP guidance, no time constraint is imposed. Target deviations are related to state deviations at t_{SI} by the equation

$$\delta\tau = M\,\delta X_{SI} . \tag{7.32}$$

The computation of the matrix M is described in the subroutine PARTL analysis section. Since δX_{SI} can be related to δX_j using

$$\delta X_{SI} = \Phi (t_{SI}, t_j) \delta X_j \tag{7.33}$$

it is easy to show that the variation matrix is given by

$$\eta_j = M \Phi (t_{SI}, t_j) \tag{7.34}$$

The variation matrix will be partitioned as follows

$$\eta_j = \left[A \mid B \right] \tag{7.35}$$

where partitions A and B are 2x3 matrices. We wish to select a $\Delta \hat{V}_j$ such that $\delta \tau = 0$ in equation (7.25). Employing equation (7.35), and defining $\delta \hat{X}_j = \left[\delta \hat{R}_j, \delta \hat{V}_j \right]^T$, this condition reduces to the equation

$$A \delta \hat{R}_j + B(\delta \hat{V}_j + \Delta \hat{V}_j) = 0 \tag{7.36}$$

This equation has no unique solution for $\Delta \hat{V}_j$ since the inverses of A and B do not exist. Non-uniqueness of $\Delta \hat{V}_j$ is to be expected since three components of $\Delta \hat{V}_j$ can be varied to satisfy the two components of τ. One degree of freedom remains and it will be used to minimize the magnitude of $\Delta \hat{V}_j$, which is equivalent to minimizing $\Delta \hat{V}_j^T \Delta \hat{V}_j$. Using standard constrained minimization techniques, the solution for $\Delta \hat{V}_j$ is given by

$$\Delta \hat{V}_j = \Gamma_{2VBP} \delta \hat{X}_j$$

where

$$\Gamma_{2VBP} = \left[-B^T(BB^T)^{-1} A \mid -B^T(BB^T)^{-1} B \right]. \tag{7.37}$$

This concludes the derivation of the guidance matrices for the three midcourse guidance policies modeled in ERRAN and SIMUL.

A quantity which is particularly useful in ERRAN since it provides the basis for the computation of statistical ΔV's is the velocity correction covariance matrix S_j, defined as follows:

$$S_j = E\left[\Delta\hat{V}_j \, \Delta\hat{V}_j^T\right] \tag{7.38}$$

A useful expression for S_j will be developed below. The derivation follows Reference 2.

Substitution of equation (7.24) into equation (7.38) yields

$$S_j = \Gamma_j E\left[\delta\hat{X}_j^- \, \delta\hat{X}_j^{-T}\right] \Gamma_j^T \tag{7.39}$$

But according to equation (7.2)

$$\delta\hat{X}_j^- = \delta X_j^- + \delta e_j^- \tag{7.40}$$

Substituting equation (7.40) into equation (7.39) and expanding yields

$$S_j = \Gamma_j \Big\{ E\left[\delta X_j^- \, \delta X_j^{-T}\right] + E\left[\delta e_j^- \delta X_j^{-T}\right] + E\left[\delta X_j^- \delta e_j^{-T}\right] + E\left[\delta e_j^- \, \delta e_j^{-T}\right] \Big\} \Gamma_j^T \tag{7.41}$$

Employing the definitions given by equations (7.1) and (7.3) the preceding equation reduces to

$$S_j = \Gamma_j \left\{ P_{c_j}^- + E\left[\delta e_j^- \, \delta X_j^{-T}\right] + E\left[\delta X_j^- \, \delta e_j^{-T}\right] + P_{K_j}^- \right\} \Gamma_j^T \tag{7.42}$$

Pre-multiplying the transpose of equation (7.40) by δe_j^-, and taking the expected value of the result yields

$$E\left[\delta e_j^- \, \delta X_j^{-T}\right] = E\left[\delta e_j^- \, \delta\hat{X}_j^{-T}\right] - P_{K_j} \tag{7.43}$$

If we assume that the estimate $\delta\hat{X}_j^-$ and the error in the estimate δe_j^- are orthogonal, as is the case if the recursive estimation algorithm is optimal, then

$$E\left[\delta e_j^- \delta\hat{X}_j^{-T}\right] = 0 \tag{7.44}$$

so that equation (7.43) reduces to

$$E\left[\delta e_j^- \delta X_j^{-T}\right] = -P_{K_j} \tag{7.45}$$

If we substitute equation (7.45) into equation (7.42), we obtain the desired result:

$$S_j = \Gamma_j \left(P_{c_j}^- - P_{K_j}^-\right) \Gamma_j^T \tag{7.46}$$

It was stated previously that ERRAN treats the midcourse guidance correction in an ensemble sense. State estimates are not generated in ERRAN, so that equation (7.24) cannot be used to determine $\Delta\hat{V}_j$. Instead, we compute a statistical or effective velocity correction in ERRAN. Simply taking the expected value of equation (7.24) does not yield useful information. The expected value of $\Delta\hat{V}_j$ is zero since $E\left[\delta\hat{X}_j\right]$ is zero, which is a consequence of the fact that our recursive estimation algorithm is an unbiased estimator. However, if we define the effective velocity correction to be

$$\text{"}E\left[\Delta\hat{V}_j\right]\text{"} = \rho_j \frac{\alpha_j}{|\alpha_j|} \tag{7.47}$$

where

$$\rho_j = E\left[\left|\Delta\hat{V}_j\right|\right] \tag{7.48}$$

and $\alpha_j / |\alpha_j|$ is a unit vector aligned with the most likely direction of the velocity correction, then information which is useful for fuel

sizing studies can be obtained. This effective velocity correction is also used to evaluate the execution error covariance $\tilde{Q}_j$ in ERRAN.

It remains to define expressions for magnitude ρ_j and direction α_j. Hoffman and Young (reference 6) have shown that a good approximation for ρ_j is given by

$$\rho_j = \sqrt{\frac{2A}{\pi}} \left(1 + \frac{B\ (\pi-2)}{A^2 \sqrt{5.4}} \right) \tag{7.49}$$

where

$$A = \text{trace } S_j,$$

$$B = \lambda_1 \lambda_2 + \lambda_1 \lambda_3 + \lambda_2 \lambda_3,$$

and λ_1, λ_2, λ_3 are the eigenvalues of the covariance matrix S_j given by equation (7.46).

The statistical variance of the magnitude of the j-th midcourse correction, also derived in approximate form in reference 6 , is given by the equation,

$$\sigma^2_{|\hat{\Delta V}_j|} = \text{trace } S_j - \rho_j^2 \tag{7.50}$$

Velocity correction covariance S_j can also be used to determine the direction α_j. Let λ_1, λ_2, and λ_3 be the eigenvalues of S_j. It can be shown that, under the assumption that some correction takes place, the most likely direction for the midcourse maneuver, defined probabilistically, is the direction of the eigenvector associated with the maximum eigenvalue of S_j. Define α_j as this eigenvector associated with the maximum eigenvalue. An alternate model for α_j assumes α_j is aligned with the vector $\left[\lambda_1, \lambda_2, \lambda_3\right]^T$. The validity of this latter model is questionable.

It should be stressed that the computation of the effective midcourse correction vector "$E\left[\hat{\Delta V}_j\right]$" within ERRAN is only an artifice

to permit a realistic, a priori computation of the execution error covariance $\tilde{Q}_j$. The nominal trajectory returned to the basic cycle is not affected by the computation. However, the calculated information concerning likely magnitudes and directions for the maneuvers is critical for fuel sizing studies.

The effective velocity correction just discussed is not defined in SIMUL since equation (7.24) can be used directly to obtain the commanded velocity correction $\Delta\hat{V}_j$. Also computed in SIMUL is the perfect velocity correction given by

$$\underline{\Delta V}_j = \Gamma_j \delta X_j^- \tag{7.51}$$

and the error in the correction due to navigation error given by

$$\Delta V_{e_j} = \underline{\Delta V}_j - \Delta\hat{V}_j \tag{7.52}$$

The perfect velocity correction $\underline{\Delta V}_j$ represents the velocity correction which is actually required to null out target errors (assuming linear guidance theory) since δX_j^- is the actual state deviation immediately prior to the correction. The error in the correction due to navigation error is indeed given by equation (7.52) since substitution of equations (7.24) and (7.51) into equation (7.52) shows that

$$\Delta V_{e_j} = \Gamma_j (\delta X_j^- - \delta X_j^-) = - \Gamma_j \delta e_j^- \tag{7.53}$$

To determine the efficacy of the midcourse correction at time t_j in meeting specified target conditions, it is necessary to compute the target condition covariance matrix W_j , both before and after the correction. Covariance W_j is defined by

$$W_j = E \left[\delta\tau \, \delta\tau^T \right] \tag{7.54}$$

where $\delta\tau$ represents the actual target state deviation. Thus W_j represents the statistical dispersions of actual target state deviations

about the nominal target state. Substitution of equation (7.25) shows that

$$W_j = \eta_j E\left[\delta X_j \; \delta X_j^T\right] \eta_j^T = \eta_j P_{c_j} \eta_j^T$$

Thus, immediately prior to the midcourse correction

$$W_j^- = \eta_j P_{c_j}^- \eta_j^T , \tag{7.55}$$

while immediately after the correction

$$W_j^+ = \eta_j P_{c_j}^+ \eta_j^T . \tag{7.56}$$

Recall that control covariance $P_{c_j}^-$ is obtained by propagating $P_{c_{j-1}}^+$ over the time interval $\left[t_{j-1}, t_j\right]$, where t_{j-1} is the time of the previous guidance event, using the standard covariance prediction equations in section 5.2. Recall also that $P_{c_j}^+$ is obtained from equation (7.16).

In SIMUL we also compute actual target errors resulting from actual spacecraft state deviations at time t_j^+ just after the guidance correction. Using equation (7.25), we obtain the total actual target error

$$\epsilon_{tot_j} = \eta_j \delta\tilde{X}_j^+ \tag{7.57}$$

Combining equations (7.8) and (7.14), we obtain

$$\delta\tilde{X}_j^+ = -\delta e_j^- + \begin{bmatrix} 0 \\ \hline \delta\Delta V_j \end{bmatrix} \tag{7.58}$$

which, when substituted into equation (7.57) yields

$$\epsilon_{tot_j} = -\eta_j \delta e_j^- + \eta_j \begin{bmatrix} 0 \\ \hline \delta\Delta V_j \end{bmatrix} \tag{7.59}$$

Thus, the total target error can be divided into the target error due to the navigation error

$$\epsilon_{nav_j} = - \eta_j \, \delta e_j^- \, . \tag{7.60}$$

and the target error due to the execution error

$$\epsilon_{ex_j} = \eta_j \begin{bmatrix} - & - & 0 & - & - \\ & & \delta \Delta V_j & & \end{bmatrix} \tag{7.61}$$

It will be helpful to summarize all the quantities computed at a midcourse guidance event in each of the two programs ERRAN and SIMUL. A summary is presented below:

	ERRAN	SIMUL
t_j^-:	X_j^-, $P_{K_j}^-$, $P_{c_j}^-$, W_j^-	$\bar{X}_j^-$, $\tilde{X}_j^-$, $\delta\hat{\tilde{X}}_j^-$, $\delta\tilde{\tilde{X}}_j^-$, $P_{K_j}^-$, $P_{c_j}^-$, W_j^-
t_j:	Γ_j, "$E\left[\Delta \hat{V}_j\right]$", $\tilde{Q}_j$	Γ_j, $\Delta\hat{V}_j$, $\delta\Delta V_j$, ΔV_j, $\underline{\Delta V}_j$, ΔV_{e_j}, $\tilde{Q}_j$
t_j^+:	$\bar{X}_j^+$, $P_{K_j}^+$, $P_{c_j}^+$, W_j^+	$\bar{X}_j^+$, $\tilde{X}_j^+$, $\delta\hat{\tilde{X}}_j^+$, $\delta\tilde{\tilde{X}}_j^+$, $P_{K_j}^+$, $P_{c_j}^+$, W_j^+, ϵ_{nav_j}, ϵ_{ex_j}, ϵ_{tot_j}

7.4.2 Biased Aimpoint Guidance

The target condition covariance W_j^+ given by equation (7.56) can be used to compute the probability of the spacecraft impacting the target planet. If planetary quarantine constraints are in effect and if the probability of impact exceeds the allowable probability of impact, then the nominal aimpoint must be biased so that the planetary quarantine constraints are satisfied. This section describes the biased aimpoint guidance technique which is used in ERRAN and SIMUL to select the biased aimpoint. The technique is based on the technique presented in reference 13.

The linear impulsive midcourse guidance policy given by equation (7.24) can be generalized to include deviations $\delta\mu_j$ from the nominal target state:

$$\Delta\hat{V}_j = \Gamma_j \delta\hat{X}_j + \Psi_j \delta\mu_j \tag{7.62}$$

The derivation of the Ψ_j guidance matrix is very similar to the derivation Γ_j (section 7.4.1). Only the details of the derivation of Ψ_j for the 2VBP policy will be presented here. We wish to select a $\Delta\hat{V}_j$ such that $\delta\mathcal{T} = \delta\mu_j$ in equation (7.25). Employing equation (7.35) and defining $\delta\hat{X}_j = \left[\delta\hat{R}_j, \delta\hat{V}_j\right]^T$, this requirement reduces to the equation

$$(A\,\delta\hat{R}_j - \delta\mu_j) + B(\delta\hat{V}_j + \Delta\hat{V}_j) = 0 \tag{7.63}$$

This equation is solved in the same manner as equation (7.36) was solved. The solution is given by

$$\Delta\hat{V}_j = \Gamma_j \delta\hat{X}_j + B^T(BB^T)^{-1}\,\delta\mu_j \tag{7.64}$$

Thus, for 2VBP guidance,

$$\Psi_j = B^T(BB^T)^{-1} \tag{7.65}$$

All aimpoint biasing will be constrained to lie in the impact plane. This requirement is automatically satisfied for the 2VBP policy. The other two policies, however, use 3-dimensional target states, which means we simply use the projection of the aimpoint in the impact plane for biased aimpoint guidance. The definition of the FTA impact plane and equations for the Ψ_j matrix for each policy are given in the subroutine BIAIM analysis section.

In computing the probability of impact PØI, we first project the target condition covariance matrix W_j^+ into the impact plane to obtain the covariance matrix Λ_j. Then, assuming the probability density function associated with Λ_j is gaussian and nearly constant over the the target planet capture area permits us to compute PØI using

$$P\emptyset I = \pi R_c^2 \, p \tag{7.66}$$

where R_c is the target planet capture radius and p represents the gaussian density function evaluated at the target planet center and is given by

$$p = \frac{1}{2\pi \left| \Lambda_j \right|^{\frac{1}{2}}} \exp \left[-\frac{1}{2} \mu_j^T \Lambda_j^{-1} \mu_j \right] \tag{7.67}$$

The general statement of the biased aimpoint guidance problem is as follows: Find an aimpoint μ_j in the impact plane which satisfies the impact probability (or planetary quarantine)constraint

$$P\emptyset I \leq P_I \, , \tag{7.68}$$

where P_I is the allowable probability of impact, and minimizes a performance functional having form

$$J = (\mu_j - \mu^*)^T \tilde{A} (\mu_j - \mu^*) = \delta\mu_j^T \tilde{A} \delta\mu_j \tag{7.69}$$

where $\tilde{A}$ is a constant symmetric matrix that will be defined subsequently.

Using equations (7.66) and (7.67), we can write the constraint equation (7.68) as

$$\mu_j^T \Lambda_j^{-1} \mu_j \geq c^2 \tag{7.70}$$

where

$$c^2 = 2 \ln \left[\frac{R_c^2}{2 |\Lambda_j|^{\frac{1}{2}} P_I} \right] \tag{7.71}$$

If we define $\mu_j = \left[\mu_1, \mu_2\right]^T$, $\mu^* = \left[\mu_1^*, \mu_2^*\right]^T$,

$$\tilde{A} = \begin{bmatrix} a_1 & a_3 \\ a_3 & a_2 \end{bmatrix}, \quad \text{and} \quad \Lambda_j^{-1} = \begin{bmatrix} \lambda_1 & \lambda_3 \\ \lambda_3 & \lambda_2 \end{bmatrix},$$

and if we introduce a slack variable q^2, then equations (7.69) and (7.70) can be written as follows:

$$J = a_1 (\mu_1 - \mu_1^*)^2 + 2a_3(\mu_1 - \mu_1)(\mu_2 - \mu_2^*) + a_2 (\mu_2 - \mu_2^*)^2 \tag{7.72}$$

$$\phi = \lambda_1 \mu_1^2 + 2 \lambda_3 \mu_1 \mu_2 + \lambda_2 \mu_2^2 - c^2 - q^2 = 0 \tag{7.73}$$

Solution of this standard constrained minimization problem yields the following set of necessary conditions:

$$q = 0 \tag{7.74}$$

$$(a_1\lambda_3 - a_3\lambda_1)\,\mu_1^2 + (a_3\lambda_2 - a_2\lambda_3)\,\mu_2^2 + (a_1\lambda_2 - a_2\lambda_1)\,\mu_1\mu_2$$

$$+ (-a_1\lambda_3\mu_1^* - a_3\lambda_3\mu_2^* + a_3\lambda_1\mu_1^* + a_2\lambda_1\mu_2^*)\,\mu_1 +$$

$$(-a_1\lambda_2\mu_1^* - a_3\lambda_2\mu_2^* + a_3\lambda_3\mu_1^* + a_2\lambda_3\mu_2^*)\,\mu_2 = 0 \qquad (7.75)$$

The method used to obtain μ_1 and μ_2 from equations (7.73) through (7.75) is described in the subroutine BIAIM analysis section.

If t_j is the time of the final midcourse correction, μ_j is chosen so that the miss distance $|\mu_j - \mu^*|$ is minimized. Defining $\tilde{A} = I$ in equation (7.69) will provide us with the appropriate μ_j. If t_j is not the final midcourse correction, μ_j will be chosen so that the velocity correction required to remove bias $\delta\mu_j$ at the time t_{j+1} of the next midcourse correction is minimized. This velocity correction is given by

$$\Delta\hat{V}_{j+1} = -\Psi_{j+1}\,\delta\mu_j \,. \qquad (7.76)$$

The square of the magnitude of $\Delta\hat{V}_{j+1}$ is given by

$$\Delta\hat{V}_{j+1}^T\,\Delta\hat{V}_{j+1} = \delta\mu_j^T\,\Psi_{j+1}^T\,\Psi_{j+1}\,\delta\mu_j \qquad (7.77)$$

Thus, setting $\tilde{A} = \Psi_{j+1}^T\,\Psi_{j+1}$ in equation (7.69) will provide us with the appropriate μ_j. Once bias $\delta\mu_j$ has been obtained, the bias velocity correction is determined from

$$\Delta\hat{V}_j = \Psi_j\,\delta\mu_j \qquad (7.78)$$

7.4.3 Nonlinear Midcourse Guidance

Options are available in ERRAN and SIMUL for computing velocity corrections using nonlinear techniques. In ERRAN we use the targeted nominal state $\bar{X}_j^-$ as the zero-th iterate in the targeting process, while in SIMUL we use the most recent state estimate $\tilde{X}_j^- + \delta\tilde{\hat{X}}_j^-$. In ERRAN we can use nonlinear guidance only for computing the bias component of the velocity correction. The statistical component "$E\left[\Delta\hat{V}_j\right]$" can only be computed using linear theory. This restriction, of course, does not apply in SIMUL. It should be noted, however, that the nonlinear two-variable B-plane policy, unlike the corresponding linear policy, constrains the z-component of $\Delta\hat{V}_j$ to be zero.

The analytical basis for nonlinear targeting is presented in section 4.4

7.5 Other Guidance Event Options

7.5.1 Re-targeting

In ERRAN and SIMUL a re-targeting event is defined to be the computation of a velocity correction $\Delta\hat{V}_{RT}$ required to achieve a new set of target conditions using nonlinear techniques. As with nonlinear midcourse guidance, we use $\bar{X}_j^-$ as the zero-th iterate in ERRAN; and $\tilde{X}_j^- + \delta\tilde{\hat{X}}_j^-$, in SIMUL. Thus, the new target conditions must be close enough to the original nominal to ensure a convergent targeting process. The analytical basis for nonlinear targeting is presented in section 4.4.

7.5.2 Orbital Insertion

An orbital insertion event is divided into a decision event and an execution event. At a decision event the orbital insertion velocity correction $\Delta\hat{V}_{\emptyset I}$ and the time interval Δt separating decision and execution are computed based on $\bar{X}_j^-$ in ERRAN and on $\tilde{X}_j^- + \delta\tilde{\hat{X}}_j^-$ in SIMUL. At an orbital insertion execution event, $\Delta\hat{V}_{\emptyset I}$ is executed and the new cartesian and orbital element states relative to the target planet are computed. The analytical basis for orbital insertion is presented in section 4.5.

7.5.3 External ΔV

An option is available in ERRAN and SIMUL for executing an externally-supplied (input) velocity correction $\Delta\hat{V}_{EX}$. In ERRAN $\Delta\hat{V}_{EX}$ is added to $\bar{X}_j^-$; in SIMUL, to $\tilde{X}_j^- + \delta\tilde{\hat{X}}_j^-$.

7.5.4 Impulse Series Thrust Model

All the analysis presented thus far in Chapter 7 is based on the assumption of an impulsive velocity correction. An option is available in ERRAN and SIMUL for modeling the impulsive velocity correction as a series of impulses. This section discusses the method of computing the effective execution error covariance $\tilde{Q}_{eff}$ and the modified update equations whenever an impulse series thrust model is used.

The modified update equations will be discussed first. For an impulsive velocity correction, execution errors increase only the velocity partition of the knowledge covariance. However, executing a velocity correction as an impulse series permits execution errors to influence the entire knowledge covariance. Thus, the knowledge covariance update equation for an impulse series is written as

$$P_{K_j}^{+} = P_{K_j}^{-} + \tilde{Q}_{eff} \tag{7.79}$$

We continue to use equation (7.16) to update the control covariance.

In ERRAN the targeted nominal state is updated as before. However, in SIMUL we compute effective estimated and actual spacecraft states at t_j. These states approximate the effect of the impulse series applied over the time interval ΔT, where ΔT brackets event time t_j. The analytical basis for the computation of effective states is presented in section 4.6. If we use $\hat{X}_{eff}$ and X_{eff} to denote the estimated and actual spacecraft states, respectively, then the state and deviation update equations employed in SIMUL are given by

$$\begin{aligned}
\tilde{\hat{X}}_j^{+} &= \hat{X}_{eff} \\
\bar{X}_j^{+} &= \hat{X}_{eff} \\
\delta\tilde{\hat{X}}_j^{+} &= X_{eff} - \hat{X}_{eff} \\
\delta\tilde{\hat{X}}_j^{+} &= 0
\end{aligned} \tag{7.80}$$

It remains to describe the computation of the effective execution error covariance $\tilde{Q}_{eff}$. We assume a series of n impulses ΔV_i, $i = 1,\ldots, n$, over the interval ΔT has been defined. Let $\tilde{Q}_i$ denote the execution error covariance associated with ΔV_i. Since ΔV_i is

impulsive we can use the impulsive execution error model described in section 7.3 to compute $\tilde{Q}_i$. The state transition matrix over the interval Δt_i separating the i-th and i+1-th impulses will be denoted by $\Phi(\Delta t_i)$. Then the effective execution error covariance after the second impulse has been applied can be written as

$$Q_{eff}^{(2)} = \Phi(\Delta t_1) \begin{bmatrix} 0 & 0 \\ 0 & \tilde{Q}_1 \end{bmatrix} \Phi^T(\Delta t_1) + \begin{bmatrix} 0 & 0 \\ 0 & \tilde{Q}_2 \end{bmatrix} \tag{7.81}$$

Proceeding recursively, we can write in similar fashion

$$\tilde{Q}_{eff}^{(i)} = \Phi(\Delta t_{i-1}) \tilde{Q}_{eff}^{(i-1)} \Phi^T(\Delta t_{i-1}) + \begin{bmatrix} 0 & 0 \\ 0 & \tilde{Q}_i \end{bmatrix} \tag{7.82}$$

where

$$\tilde{Q}_{eff}^{(1)} = \begin{bmatrix} 0 & 0 \\ 0 & \tilde{Q}_1 \end{bmatrix}$$

Then the effective execution error covariance matrix for the entire interval ΔT is given by

$$\tilde{Q}_{eff} = \tilde{Q}_{eff}^{(n)} . \tag{7.83}$$

8. GENERALIZED COVARIANCE ANALYSIS

8.1 Introduction

The performance of navigation filters for orbit determination depends on how well the physical environment and ground-based or onboard measurement instrumentation can be modeled. The design of a navigation filter involves not only selection of an algorithm for processing measurements, but also specification of error models for all error sources thought to be important. The use of an error analysis technique, such as the one described in Chapter 5, is not sufficient for determining actual filter performance in the presence of incorrectly modeled or unmodeled error sources. Although one could, of course, resort to a simulation technique such as SIMUL (Chapter 6) to study filter performance, the operation of simulation programs is expensive and only a single sample of the navigation process can be generated on each run. A generalized covariance program, however, can provide much useful information relating to the design and performance of navigation filters, with a significant reduction in program operating costs.

The generalized covariance technique described in this chapter is primarily concerned with the propagation and update (at a measurement) of both actual and assumed, i.e., filter-generated, estimation error statistics along a nominal trajectory. The deviation of the generalized covariance equations assumes linearity and gaussian statistics. Actual error statistics, however, are not required to have zero means. The equations are written in recursive form and are filter-independent, i.e., filter gains are not assumed to have been generated by any specific type of navigation filter.

The generalized covariance equations for the basic cycle (measurement processing) are derived in section 8.2. These equations can be used to determine filter sensitivity to differences between assumed (by filter) and actual:

1) Injection statistics;

2) Measurement noise statistics -- doppler, range, optical measurements;

3) Dynamic parameter statistics -- gravitational constants, target planet ephemerides;

4) Measurement parameter statistics -- instrument biases, station location errors;

5) Dynamic noise statistics.

The differences between assumed and actual error statistics can involve differences in means, standard deviations, and correlation coefficients. Actual error statistics can also be defined for parameters whose uncertainty has been ignored in filter design.

In section 8.3 the generalized covariance technique is extended to the guidance process. The equations presented there permit one to determine the sensitivity of the guidance process to differences between assumed and actual execution error statistics, as well as to differences in the previously described error statistics. Although execution errors are assumed to be uncorrelated, they are permitted to have nonzero means. The generalized covariance technique, as applied to the guidance process, primarily involves the computation of both assumed and actual target dispersions and velocity correction statistics.

The notation employed in this chapter is very similar to the notation used in previous chapters, except for the following differences:

1) Estimation errors are denoted by $\tilde{x}$, etc instead of by δe, etc;

2) Actual errors, deviations, means, covariances, etc are usually denoted by $(\ )'$.

8.2 Generalized Covariance Propagation and Update

8.2.1 The Basic Cycle

The generalized covariance basic cycle consists of the propagation of both actual and assumed estimation error means and covariances from the previous measurement time (or event) to the present measurement time, and the updating of each of these quantities after the measurement has been processed. The propagation and update of the assumed covariances was treated in Chapter 5 (assumed estimation error means are zero). The equations required to propagate and update the actual estimation error means and covariances are derived in this section. These equations are filter-independent and are expressed in terms of arbitrary filter gain matrices.

The filter employs an augmented state vector x^A partitioned as

$$x^A = \begin{bmatrix} x \\ x_s \\ u \\ v \end{bmatrix} \tag{8.1}$$

where x denotes assumed position/velocity deviations (from nominal); x_s, assumed solve-for parameter deviations; u, assumed dynamic consider parameter deviations; and v, assumed measurement consider parameter deviations. The assumed dynamics are described by

$$x_{k+1} = \Phi x_k + \theta_{xx_s} x_{s_k} + \theta_{xu} u_k + \omega_{k+1} \tag{8.2}$$

where state transition matrix partitions Φ, θ_{xx_s}, and θ_{xu} are defined over the time interval $[t_k, t_{k+1}]$, and ω_{k+1} denotes the contribution of assumed unmodeled accelerations over the same time interval. Parameter deviations are constant. The assumed measurement is given by

$$y_{k+1} = Hx_{k+1} + Mx_{s_{k+1}} + Gu_{k+1} + Lv_{k+1} + \nu_{k+1} \tag{8.3}$$

where H, M, G, and L are observation matrix partitions evaluated at time t_{k+1}, and ν_{k+1} denotes the assumed measurement noise.

The actual augmented state vector x'^A is partitioned as

$$x'^A = \begin{bmatrix} x' \\ x'_s \\ u' \\ v' \\ w' \end{bmatrix} \tag{8.4}$$

where x' denotes actual position/velocity deviations; x'_s, actual solve-for parameter deviations; u', actual dynamic consider parameter deviations; v', actual measurement consider parameter deviations; and w', actual dynamic and measurement ignore parameter deviations. The parameters x'_s, u', and v' correspond to x_s, u, and v, respectively, but have different statistical representations. Ignore parameters w' are parameters whose statistical uncertainty is completely ignored by the filter, but not by the actual estimation error mean and covariance propagation process. (Parameters not treated by either the filter or the actual propagation process will be referred to as neglect parameters.) The actual dynamics

are described by

$$x'_{k+1} = \Phi x'_k + \theta_{xx_s} x'_{s_k} + \theta_{xu} u'_k + \theta_{xw} w'_k + \omega'_{k+1} \qquad (8.5)$$

where ω'_{k+1} denotes the contribution of actual unmodeled accelerations over the time interval $[t_k, t_{k+1}]$. State transition matrix partition θ_{xw} relates changes in ignore parameters to changes in x'. All parameter deviations are constant. The actual measurement is given by

$$y'_{k+1} = Hx'_{k+1} + Mx'_{s_{k+1}} + Gu'_{k+1} + Lv'_{k+1} + Nw'_{k+1} + \nu'_{k+1} \qquad (8.6)$$

where ν'_{k+1} denotes the actual measurement noise at t_{k+1}.

The actual estimation errors are defined by

$$\tilde{x}'_{k+1} = \hat{x}_{k+1} - x'_{k+1} \qquad (8.7)$$

$$\tilde{x}'_{s_{k+1}} = \hat{x}_{s_{k+1}} - x'_{s_{k+1}} \qquad (8.8)$$

$$\tilde{u}'_{k+1} = \hat{u}_{k+1} - u'_{k+1} = -u'_o \qquad (8.9)$$

$$\tilde{v}'_{k+1} = \hat{v}_{k+1} - v'_{k+1} = -v'_o \qquad (8.10)$$

$$\tilde{w}'_{k+1} = \hat{w}_{k+1} - w'_{k+1} = -w'_o \qquad (8.11)$$

where equations (8.9), (8.10), and (8.11) have used the fact that estimates $\hat{u}$, $\hat{v}$, and $\hat{w}$ are always zero.

The estimates propagate over the time interval $[t_k, t_{k+1}]$ according to

$$\hat{x}^-_{k+1} = \Phi \hat{x}^+_k + \theta_{xx_s} \hat{x}^+_{s_k} \qquad (8.12)$$

and

$$\hat{x}^-_{s_{k+1}} = \hat{x}^+_{s_k} , \qquad (8.13)$$

3-5

where $(\ )^-$ denotes values immediately before processing a measurement and $(\ )^+$ immediately after. Substitution of equations (8.5) and (8.12) into equation (8.7) yields the following equation for the propagation of the actual estimation error:

$$\tilde{x}'^{-}_{k+1} = \Phi \tilde{x}'^{+}_{k} + \theta_{xx_s} \hat{x}'^{+}_{s_k} - \theta_{xu} u'_o - \theta_{xw} w'_o - \omega'_{k+1} \quad . \tag{8.14}$$

Similarly,

$$\tilde{x}'^{-}_{s_{k+1}} = \tilde{x}'^{+}_{s_k} \quad . \tag{8.15}$$

At measurement time t_{k+1} the estimates are updated using the equations

$$\hat{x}^+_{k+1} = \hat{x}^-_{k+1} + K_{k+1} \varepsilon'_{k+1} \tag{8.16}$$

$$\hat{x}^+_{s_{k+1}} = \hat{x}^-_{s_{k+1}} + S_{k+1} \varepsilon'_{k+1} \tag{8.17}$$

where K_{k+1} and S_{k+1} are the filter gain matrices (generated by an arbitrary filter). The actual measurement residual ε' is defined as the difference between the actual and predicted measurements

$$\varepsilon'_{k+1} = y'_{k+1} - H\hat{x}^-_{k+1} - M\hat{x}^-_{s_{k+1}} \quad . \tag{8.18}$$

Substitution of equation (8.6) into equation (8.18) yields

$$\varepsilon'_{k+1} = - H\tilde{x}'^{-}_{k+1} - M\tilde{x}'^{-}_{s_{k+1}} + Gu'_o + Lv'_o + Nw'_o + \nu'_{k+1} \quad . \tag{8.19}$$

The update equation for the actual estimation error is obtained by substituting equation (8.16) into equation (8.7). The resulting equation is

$$\tilde{x}'^{+}_{k+1} = \tilde{x}'^{-}_{k+1} + K_{k+1} \varepsilon'_{k+1} \quad . \tag{8.20}$$

Similarly,

$$\tilde{x}^{+\prime}_{s_{k+1}} = \tilde{x}^{-\prime}_{s_{k+1}} + S_{k+1}\,\varepsilon'_{k+1} \quad . \tag{8.21}$$

The propagation and update equations for the means of the actual estimation errors and the actual measurement residuals can now be derived. The filter assumes zero means for all estimates and all error sources. Except for actual dynamic and measurement noises, this is not the case for the actual propagation and update process. Thus,

$$\begin{aligned} E[\tilde{u}'_{k+1}] &= -\bar{u}'_o \\ E[\tilde{v}'_{k+1}] &= -\bar{v}'_o \\ E[\tilde{w}'_{k+1}] &= -\bar{w}'_o \\ E[\omega'_{k+1}] &= 0 \\ E[\nu'_{k+1}] &= 0 \quad . \end{aligned} \tag{8.22}$$

No generality is lost by setting the mean of the actual measurement noise ν' to zero, since a nonzero measurement mean can be absorbed into the mean of the actual measurement bias. The model for the actual dynamic noise ω' will be assumed to have the same form as the model for the assumed dynamic noise described in section 5.3 so the mean of ω' is also set to zero.

Applying the expectation operator to equations (8.14) and (8.15) yields the following equations for the propagation of the means of the actual estimation errors:

$$E[\tilde{x}^{-\prime}_{k+1}] = \Phi \cdot E[\tilde{x}^{+\prime}_{k}] + \theta_{xx_s} \cdot E[\tilde{x}^{+\prime}_{s_k}] - \theta_{xu}\,\bar{u}'_o - \theta_{xw}\,\bar{w}'_o \tag{8.23}$$

$$E[\tilde{x}^{-\prime}_{s_{k+1}}] = E[\tilde{x}^{+\prime}_{s_k}] \quad . \tag{8.24}$$

To initiate the propagation process described by the previous two equations requires initial values for the means of $\tilde{x}'$ and $\tilde{x}'_s$. At initial time t_o we have

$$E[\tilde{x}'_o] = E[\hat{x}_o] - E[x'_o] \tag{8.25}$$

and

$$E[\tilde{x}'_{s_o}] = E[\hat{x}_{s_o}] - E[x'_{s_o}] \quad . \tag{8.26}$$

Because initial estimates are always assumed to be zero, equations (8.25) and (8.26) become

$$E[\tilde{x}'_o] = -\bar{x}'_o \tag{8.27}$$

$$E[\tilde{x}'_{s_o}] = -\bar{x}'_{s_o} \tag{8.28}$$

where $\bar{x}'_o$ and $\bar{x}'_{s_o}$ are the initial means of the actual position/velocity and solve-for parameter deviations, respectively.

Applying the expectation operator to equation (8.19) yields the following equation for the mean of the actual measurement residual:

$$E[\varepsilon'_{k+1}] = - H \cdot E[\tilde{x}^{-\prime}_{k+1}] - M \cdot E[\tilde{x}^{-\prime}_{s_{k+1}}] + G\bar{u}'_o + L\bar{v}'_o + N\bar{w}'_o \quad . \tag{8.29}$$

The update equations for the means of the actual estimation errors are obtained by applying the expectation operator to equations (8.20) and (8.21). The resulting equations are:

$$E[\tilde{x}^{+\prime}_{k+1}] = E[\tilde{x}^{-\prime}_{k+1}] + K_{k+1} \cdot E[\varepsilon'_{k+1}] \tag{8.30}$$

$$E[\tilde{x}^{+\prime}_{s_{k+1}}] = E[\tilde{x}^{-\prime}_{s_{k+1}}] + S_{k+1} \cdot E[\varepsilon'_{k+1}] \quad . \tag{8.31}$$

The remainder of this section will treat the derivation of the propagation and update equations for the actual knowledge covariance matrix partitions. Since the actual estimation errors do not, in general, have zero means, it becomes more convenient, from both an analytical and a computational standpoint, to develop propagation and update equations for the 2nd moment matrices rather than for the covariance matrices, and then simply convert the 2nd moment matrices to covariance matrices using the standard relationship

$$\text{cov}\ (x,y) = E[xy^T] - \bar{x}\,\bar{y}^T \tag{8.32}$$

where cov (x,y) denotes the covariance of x and y, and $E[xy^T]$ denotes the 2nd moment matrix of x and y.

The required actual 2nd moment matrix partitions are defined in the following pages. Note that primes have been dropped from the 2nd moment variables to make the equations more readable in the remainder of this section. The 2nd moment matrix partitions that must be updated whenever a measurement is processed are listed first.

$$\begin{aligned}
P &= E[\tilde{x}'\ \tilde{x}'^T] & P_s &= E[\tilde{x}'_s\ \tilde{x}'^T_s] \\
C_{xx_s} &= E[\tilde{x}'\ \tilde{x}'^T_s] & C_{x_s u} &= E[\tilde{x}'_s\ \tilde{u}'^T] \\
C_{xu} &= E[\tilde{x}'\ \tilde{u}'^T] & C_{x_s v} &= E[\tilde{x}'_s\ \tilde{v}'^T] \\
C_{xv} &= E[\tilde{x}'\ \tilde{v}'^T] & C_{x_s w} &= E[\tilde{x}'_s\ \tilde{w}'^T] \\
C_{xw} &= E[\tilde{x}'\ \tilde{w}'^T]
\end{aligned} \tag{8.33}$$

The remaining 2nd moment matrix partitions do not change with time:

$$\begin{aligned}
C_{uv} &= E[\tilde{u}'\ \tilde{v}'^T] = C_{uv_o} & U &= E[\tilde{u}'\ \tilde{u}'^T] = U_o \\
C_{uw} &= E[\tilde{u}'\ \tilde{w}'^T] = C_{uw_o} & V &= E[\tilde{v}'\ \tilde{v}'^T] = V_o \\
C_{vw} &= E[\tilde{v}'\ \tilde{w}'^T] = C_{vw_o} & W &= E[\tilde{w}'\ \tilde{w}'^T] = W_o
\end{aligned} \tag{8.34}$$

The 2nd moment matrix propagation equations for the time interval $[t_k, t_{k+1}]$ are obtained by substituting equations (8.14) and (8.15) into (8.33) and expanding. All equations are simplified by assuming ω_k' and $(\tilde{x}_k', \tilde{x}_{s_k}', \tilde{u}_k', \tilde{v}_k', \tilde{w}_k')$ are uncorrelated. Thus, for example,

$$E[\tilde{x}_k' \, \omega_k'^T] = E[\tilde{x}_k'] \cdot E[\omega_k'^T] = 0$$

since the mean of ω_k' has been assumed to be zero. The final propagation equations are summarized below:

$$P_{k+1}^- = \left(\Phi P_k^+ + \theta_{xx_s} C_{xx_s}^{+T}{}_k + \theta_{xu} C_{xu}^{+T}{}_k + \theta_{xw} C_{xw}^{+T}{}_k\right)\Phi^T + C_{xx_s}^-{}_{k+1} \theta_{xx_s}^T + C_{xu}^-{}_{k+1} \theta_{xu}^T + C_{xw}^-{}_{k+1} \theta_{xw}^T + Q_{k+1} \quad (8.35)$$

$$C_{xx_s}^-{}_{k+1} = \Phi C_{xx_s}^+{}_k + \theta_{xx_s} P_{s_k}^+ + \theta_{xu} C_{x_s u}^{+T}{}_k + \theta_{xw} C_{x_s w}^{+T}{}_k \quad (8.36)$$

$$C_{xu}^-{}_{k+1} = \Phi C_{xu}^+{}_k + \theta_{xx_s} C_{x_s u}^+{}_k + \theta_{xu} U_o + \theta_{xw} C_{uw}^T{}_o \quad (8.37)$$

$$C_{xv}^-{}_{k+1} = \Phi C_{xv}^+{}_k + \theta_{xx_s} C_{x_s v}^+{}_k + \theta_{xu} C_{uv}{}_o + \theta_{xw} C_{vw}^T{}_o \quad (8.38)$$

$$C_{xw}^-{}_{k+1} = \Phi C_{xw}^+{}_k + \theta_{xx_s} C_{x_s w}^+{}_k + \theta_{xu} C_{uw}{}_o + \theta_{xw} W_o \quad (8.39)$$

$$P_{s_{k+1}}^- = P_{s_k}^+ \quad (8.40)$$

$$C_{x_s u}^-{}_{k+1} = C_{x_s u}^+{}_k \quad (8.41)$$

$$C^{-}_{x_s v_{k+1}} = C^{+}_{x_s v_k} \tag{8.42}$$

$$C^{-}_{x_s w_{k+1}} = C^{+}_{x_s w_k} \quad . \tag{8.43}$$

The actual dynamic noise 2nd moment matrix Q will be assumed to have form

$$Q_{k+1} = \text{diag}\left(\tfrac{1}{4} K'_1 \Delta t^4, \tfrac{1}{4} K'_2 \Delta t^4, \tfrac{1}{4} K'_3 \Delta t^4, K'_1 \Delta t^2, K'_2 \Delta t^2, K'_3 \Delta t^2\right) \tag{8.44}$$

where $\Delta t = t_{k+1} - t_k$, and K'_1, K'_2, and K'_3 are constants which roughly correspond to the variances of the actual unmodeled accelerations. The form of this equation is identical to that of equation (5.23).

The actual measurement residual 2nd moment matrix is defined by

$$J_{k+1} = E[\varepsilon'_{k+1} \varepsilon'^{T}_{k+1}] \quad . \tag{8.45}$$

Substituting equation (8.19) into equation (8.45) yields

$$J_{k+1} = HA_{k+1} + MB_{k+1} + GD_{k+1} + LE_{k+1} + NF_{k+1} + R_{k+1} \tag{8.46}$$

where observation matrix partitions H, M, G, L, and N have been defined previously, R is the actual measurement noise 2nd moment matrix defined by

$$R_{k+1} = E[\nu'_{k+1} \nu'^{T}_{k+1}] \tag{8.47}$$

and

$$A_{k+1} = P^{-}_{k+1} H^T + C^{-}_{xx_{s_{k+1}}} M^T + C^{-}_{xu_{k+1}} G^T + C^{-}_{xv_{k+1}} L^T + C^{-}_{xw_{k+1}} N^T \tag{8.48}$$

$$B_{k+1} = P^{-}_{s_{k+1}} M^T + C^{-T}_{xx_{s_{k+1}}} H^T + C^{-}_{x_s u_{k+1}} G^T + C^{-}_{x_s v_{k+1}} L^T + C^{-}_{x_s w_{k+1}} N^T \tag{8.49}$$

$$D_{k+1} = C^{-T}_{xu_{k+1}} H^T + C^{-T}_{x_s u_{k+1}} M^T + U_o G^T + C^{-}_{uw_o} N^T + C^{-}_{uv_o} L^T \quad (8.50)$$

$$E_{k+1} = C^{-T}_{xv_{k+1}} H^T + C^{-T}_{x_s v_{k+1}} M^T + C^{-}_{vw_o} N^T + V_o L^T + C^{-T}_{uv_o} G^T \quad (8.51)$$

$$F_{k+1} = W_o N^T + C^{-T}_{xw_{k+1}} H^T + C^{-T}_{x_s w_{k+1}} M^T + C^{-T}_{vw_o} L^T + C^{-T}_{uw_o} G^T \quad . \quad (8.52)$$

The 2nd moment matrix update equations, which correspond to the processing of a measurement, are obtained by substituting equations (8.20) and (8.21) into (8.33) and expanding. The final update equations are summarized as

$$P^{+}_{k+1} = P^{-}_{k+1} - K_{k+1} A^T - AK^T_{k+1} + K_{k+1} J_{k+1} K^T_{k+1} \quad (8.53)$$

$$C^{+}_{xx_{s_{k+1}}} = C^{-}_{xx_{s_{k+1}}} - K_{k+1} B^T - AS^T_{k+1} + K_{k+1} J_{k+1} S^T_{k+1} \quad (8.54)$$

$$C^{+}_{xu_{k+1}} = C^{-}_{xu_{k+1}} - K_{k+1} D^T \quad (8.55)$$

$$C^{+}_{xv_{k+1}} = C^{-}_{xv_{k+1}} - K_{k+1} E^T \quad (8.56)$$

$$C^{+}_{xw_{k+1}} = C^{-}_{xw_{k+1}} - K_{k+1} F^T \quad (8.57)$$

$$P^{+}_{s_{k+1}} = P^{-}_{s_{k+1}} - S_{k+1} B^T - BS^T_{k+1} + S_{k+1} J_{k+1} S^T_{k+1} \quad (8.58)$$

$$C^{+}_{x_s u_{k+1}} = C^{-}_{x_s u_{k+1}} - S_{k+1} D^T \quad (8.59)$$

$$C^{+}_{x_s v_{k+1}} = C^{-}_{x_s v_{k+1}} - S_{k+1} E^T \quad (8.60)$$

$$C^{+}_{x_s w_{k+1}} = C^{-}_{x_s w_{k+1}} - S_{k+1} F^T \quad (8.61)$$

where K_{k+1} and S_{k+1} are the filter gain constants.

8.2.2 Eigenvector and Prediction Events

The generalized covariance treatment of eigenvector and prediction events is quite similar to their treatment in an error analysis. At an eigenvector event, eigenvalues, eigenvectors, and hyperellipsoids are computed for both assumed and actual knowledge covariances. At a prediction event, both assumed and actual knowledge covariances are propagated forward to t_p, the time to which the prediction is to be made. If t_p occurs within the target planet sphere of influence, both assumed and actual covariances are converted from cartesian coordinates to B-plane coordinates (see section 5.6 for more details).

8.3 Generalized Midcourse Guidance Analysis

8.3.1 Target Condition Dispersion Analysis

To generate actual target condition dispersions (mean plus covariance) requires that equations be developed, first, for propagating actual deviation means (control means) and actual control 2nd moment matrix partitions over the time interval $[t_{j-1}, t_j]$ separating two successive guidance events. Second, equations must be developed for updating actual knowledge and control means and 2nd moment matrices following the execution of a guidance event. These equations are derived in this section.

The actual dynamics over the time interval $[t_{j-1}, t_j]$ are described by

$$x'^{-}_j = \Phi x'^{+}_{j-1} + \theta_{xx_s} x'_{s_o} + \theta_{xu} u'_o + \theta_{xw} w'_o + \omega'_j \qquad (8.62)$$

where actual parameter deviations x'_s, u', v', and w' do not change with time. Notation $(\,)^-$ and $(\,)^+$ indicates values immediately before and after a guidance correction, respectively. Applying the expectation operator to equation (8.62) yields the following mean propagation equation:

$$\bar{x}'^{-}_j = \Phi \bar{x}'^{+}_{j-1} + \theta_{xx_s} \bar{x}'_{s_o} + \theta_{xu} \bar{u}'_o + \theta_{xw} \bar{w}'_o \qquad (8.63)$$

where we have assumed that the mean of actual unmodeled acceleration ω'_j is zero.

The dispersions of the actual deviations about the targeted nominal trajectory are represented by the control 2nd moment matrix

$$P'_{c_j} = E\left[x'_j\ x'^{T}_j\right] \tag{8.64}$$

and related partitions. Equations (8.35) through (8.43) can be used to propagate all control 2nd moment matrix partitions over the interval $[t_{j-1},\ t_j]$ if we treat all 2nd moment variables appearing in these equations as control 2nd moment variables.

Initial control 2nd moment matrix partitions are identical to initial knowledge 2nd moment matrix partitions since all initial estimates are zero.

The updating of actual knowledge and control means and 2nd moment matrices following the execution of a guidance event reflects the introduction of actual execution error statistics into the mean and 2nd moment matrix propagation processes. The actual estimation error at a guidance event is increased by the actual execution error

$$\delta\Delta V'_j = \Delta V'_j - \Delta\hat{V}'_j \tag{8.65}$$

where $\Delta V'_j$ is the actual velocity correction and $\Delta\hat{V}'_j$ is the actual commanded velocity correction. Therefore, the estimation error immediately following the velocity correction is given by

$$\tilde{x}'^{+}_j = \tilde{x}'^{-}_j - A \cdot \delta\Delta V'_j \tag{8.66}$$

where

$$A = [0 \vdots I]^T \quad .$$

The actual knowledge 2nd moment matrix immediately following the correction is defined by

$$P'^{+}_{K_j} = E\left[\tilde{x}'^{+}_j\ \ \tilde{x}'^{+T}_j\right] \quad . \tag{8.67}$$

Substitution of equation (8.66) into equation (8.67) yields

$$P'^{+}_{K_j} = P'^{-}_{K_j} + A \cdot E\left[\delta\Delta V'_j \; \delta\Delta V'^{T}_j\right] \cdot A^T - A \cdot E\left[\delta\Delta V'_j \; \tilde{x}'^{-T}_j\right]$$

$$- E\left[\tilde{x}'^{-}_j \; \delta\Delta V'^{T}_j\right] \cdot A^T \quad . \tag{8.68}$$

Defining the actual execution error 2nd moment matrix by

$$\tilde{Q}'_j = E\left[\delta\Delta V'_j \; \delta\Delta V'^{T}_j\right] \tag{8.69}$$

and assuming the estimation error immediately prior to the correction and the execution error to be uncorrelated permits us to rewrite equation (8.68) as

$$P'^{+}_{K_j} = P'^{-}_{K_j} + A\tilde{Q}'_j \; A^T - A \cdot E\left[\delta\Delta V'_j\right] \cdot E\left[\tilde{x}'^{-T}_j\right]$$

$$- E\left[\tilde{x}'^{-}_j\right] \cdot E\left[\delta\Delta V'^{T}_j\right] \cdot A^T \quad . \tag{8.70}$$

The mean of the actual estimation error immediately following the correction is obtained simply by applying the expectation operator to equation (8.66) to obtain

$$E\left[\tilde{x}'^{+}_j\right] = E\left[\tilde{x}'^{-}_j\right] - A \cdot E\left[\delta\Delta V'_j\right] \quad . \tag{8.71}$$

The propagation equation for $E\left[\tilde{x}'^{-}_j\right]$ is given by equation (8.23). An expression for $E\left[\delta\Delta V'_j\right]$ is given in section 8.3.3.

The actual estimation error following the correction is defined by

$$\tilde{x}'^{+}_j = \hat{x}'^{+}_j - x'^{+}_j \quad . \tag{8.72}$$

But, since we assume that the nominal state is updated with the most recent estimate at a guidance event,

$$\hat{x}'^{+}_j = 0 \quad . \tag{8.73}$$

Then, substituting equations (8.72) and (8.73) into equation (8.66) yields

$$x_j'^{+} = - \tilde{x}_j'^{-} + A \cdot \delta\Delta V_j' \quad . \tag{8.74}$$

The actual control 2nd moment matrix following the correction is defined by

$$P_{c_j}'^{+} = E\left[x_j'^{+}\ x_j'^{+T}\right] \quad . \tag{8.75}$$

Substitution of equation (8.74) into equation (8.75) and comparing the result with equation (8.68) shows that

$$P_{c_j}'^{+} = P_{K_j}'^{+} \quad . \tag{8.76}$$

Taking the expected value of equation (8.74) and comparing the results with equation (8.71) shows that

$$E\left[x_j'^{+}\right] = - E\left[\tilde{x}_j'^{+}\right] \quad . \tag{8.77}$$

Similarly, under the assumption that we update the nominal solve-for state, we can write

$$E\left[x_{s_j}'^{+}\right] = - E\left[\tilde{x}_{s_j}'^{+}\right] \quad . \tag{8.78}$$

The remaining control 2nd moment matrix partitions are updated in the same manner as the position/velocity partition is updated in equation (8.76).

Equations for the actual target dispersions can now be developed. The actual target state deviation $\delta\tau_j'$ is related to the actual state deviation x_j' at time t_j according to

$$\delta\tau_j' = \eta_j\ x_j' \tag{8.79}$$

where η_j is the variation matrix (see section 7.4.1) for the appropriate midcourse guidance policy. The mean of $\delta\tau_j'$ is given by

$$E\left[\delta\tau_j'\right] = \eta_j \, E\left[x_j'\right] \quad . \tag{8.80}$$

The statistical target dispersions are represented by the actual target condition 2nd moment matrix W_j', which is defined as

$$W_j' = E\left[\delta\tau_j' \, \delta\tau_j'^{T}\right] \quad . \tag{8.81}$$

Substitution of equation (8.79) into equation (8.81) yields

$$W_j' = \eta_j \, P_{c_j}' \, \eta_j^{T} \quad . \tag{8.82}$$

Equations (8.80) and (8.82) are evaluated immediately before and after the guidance correction at time t_j.

8.3.2 Velocity Correction Analysis

The actual commanded velocity correction 2nd moment matrix is defined by

$$S_j' = E\left[\Delta\hat{V}_j' \, \Delta\hat{V}_j'^{T}\right] \tag{8.83}$$

where the actual commanded velocity correction is given by

$$\Delta\hat{V}_j' = \Gamma_j \, \hat{x}_j' = \Gamma_j \, (x_j' + \tilde{x}_j') \quad . \tag{8.84}$$

The guidance matrix Γ_j corresponds to the appropriate linear mid-course guidance policy (see section 7.4.1).

Substitution of equation (8.84) into equation (8.83) yields

$$S_j' = \Gamma_j \left\{ E\left[x_j' \, x_j'^{T}\right] + E\left[\tilde{x}_j' \, \tilde{x}_j'^{T}\right] + E\left[x_j' \, \tilde{x}_j'^{T}\right] + E\left[\tilde{x}_j' \, x_j'^{T}\right] \right\} \Gamma_j^{T} \quad . \tag{8.85}$$

We can write

$$E\left[x_j' \, \tilde{x}_j'^{T}\right] = E\left[\hat{x}_j' \, \tilde{x}_j'^{T}\right] - E\left[\tilde{x}_j' \, \tilde{x}_j'^{T}\right] \quad . \tag{8.86}$$

Then, substituting equation (8.86) into equation (8.85), we obtain

$$S_j' = \Gamma_j \left\{ E\left[x_j' \, x_j'^{T}\right] - E\left[\tilde{x}_j' \, \tilde{x}_j'^{T}\right] + E\left[\hat{x}_j' \, \tilde{x}_j'^{T}\right] + E\left[\tilde{x}_j' \, \hat{x}_j'^{T}\right] \right\} \Gamma_j^{T} \quad . \tag{8.87}$$

If we define

$$C'_{E_j} = E\left[\hat{x}'_j \, \tilde{x}'^{T}_j\right] \tag{8.88}$$

and use the definitions of control and knowledge 2nd moment matrices, we can write equation (8.87) as

$$S'_j = \Gamma_j \left(P'_{C_j} - P'_{K_j} + C'_{E_j} + C'^{T}_{E_j}\right) \Gamma_j^T \quad . \tag{8.89}$$

The corresponding expression for the assumed velocity correction covariance (section 7.4.1) is given by

$$S_j = \Gamma_j \left(P_{C_j} - P_{K_j}\right) \Gamma_j^T \quad . \tag{8.90}$$

The assumed covariance C_{E_j} does not appear in equation (8.90) since the navigation filter assumed the estimate and the estimation error to be orthogonal (if the filter employs an optimal estimation algorithm).

The proper evaluation of equation (8.89) for S'_j requires that C'_{E_j} and associated partitions be propagated between measurements and updated at each measurement. A set of propagation and update equations for C'_{E_j} and associated partitions can be developed in a straightforward fashion. These is some question, however, about the feasibility of carrying along an additional set of 2nd moment partitions merely to obtain a better value for S'_j. The programmed generalized covariance guidance model will assume C_{E_j} can be neglected in equation (8.89).

The mean of the actual commanded velocity correction is obtained by applying the expectation operator to equation (8.84):

$$E\left[\Delta\hat{V}'_j\right] = \Gamma_j \left\{E\left[x'_j\right] + E\left[\tilde{x}'_j\right]\right\} \quad . \tag{8.91}$$

Expressions for $E[x'_j]$ and $E[\tilde{x}'_j]$ are already available.

Equation (8.91) gives no useful information for fuel sizing studies. Instead, we must develop an expression for $E\left[\left|\Delta\hat{V}'_j\right|\right]$. In section 7.4.1 the Hoffman-Young formula is used to evaluate the assumed $E\left[\left|\Delta V_j\right|\right]$. We shall use the same formula to evaluate $E\left[\left|\Delta V'_j\right|\right]$. Thus

$$E\left[\left|\Delta V'_j\right|\right] = \sqrt{\frac{2A}{\pi}}\left(1 + \frac{B\ (\pi-2)}{A^2\ \sqrt{5.4}}\right) \tag{8.92}$$

where now

$$A = \text{trace } S'_j$$

$$B = \lambda'_1\ \lambda'_2 + \lambda'_1\ \lambda'_3 + \lambda'_2\ \lambda'_3 \quad ,$$

and λ'_1, λ'_2, and λ'_3 are the eigenvalues of the 2nd moment matrix S'_j. No rigorous justification for the use of S'_j rather than cov $(\Delta\hat{V}'_j)$ to evaluate $E\left[\left|\Delta V'_j\right|\right]$ is available at present. It should be apparent, however, that the use of cov $(\Delta\hat{V}'_j)$ would not reflect the fact that the actual commanded velocity statistics are distributed symmetrically, not about the origin, but about $E[\Delta\hat{V}'_j]$.

The actual effective or statistical ΔV is defined as

$$\text{"}E\left[\Delta\hat{V}'_j\right]\text{"} = E\left[\left|\Delta\hat{V}'_j\right|\right] \cdot \alpha'_j \tag{8.93}$$

where α'_j denotes a unit vector in the most likely direction of the velocity correction. Although two options are presently available in ERRAN for computing the most likely direction of the assumed statistical ΔV, only one of these options will be defined for the generalized covariance guidance model. The first option assumes α'_j is aligned with $\left[S'_{11},\ S'_{22},\ S'_{33}\right]^T$, where the S'_{ii} are the diagonal elements of S'_j. Because this method presents difficulties in the evaluation of $E\ [\delta\Delta V'_j]$ and $\tilde{Q}'_j$, it will not

be used. The 2nd option, which will be used in the generalized covariance guidance model, assumes α'_j is aligned with the eigenvector associated with the maximum eigenvalue of S'_j.

8.3.3 Execution Error Model

The actual execution error $\delta\Delta V'_j$ will be assumed to have form

$$\delta\Delta V'_j = k'\Delta V'_j + s' \frac{\Delta V'_j}{|\Delta V'_j|} + \delta\Delta V'_{pointing} \tag{8.94}$$

where k' denotes the actual proportionality error; s', the actual resolution error; and $\delta\Delta V'_{pointing}$ the actual pointing error. These actual execution errors are not required to have zero-mean statistics.

Both the mean and 2nd moment of $\delta\Delta V'_j$ are difficult to evaluate because of the complicated functional dependence of $\delta\Delta V'_j$ on $\Delta\hat{V}'_j$. This problem is also encountered in the generation of assumed execution error statistics, and will be resolved by making certain simplifying assumptions.

The components of $\delta\Delta V'_j$ can be found in equation (7.21) and are reproduced as

$$\delta\Delta V'_x = \left(k' + \frac{s'}{\rho'}\right)\Delta\hat{V}'_x + \frac{\rho'\Delta\hat{V}'_y\,\delta\alpha' + \Delta\hat{V}'_x\,\Delta\hat{V}'_z\,\delta\beta'}{\mu'} \tag{8.95}$$

$$\delta\Delta V'_y = \left(k' + \frac{s'}{\rho'}\right)\Delta\hat{V}'_y + \frac{\Delta\hat{V}'_y\,\Delta\hat{V}'_z\,\delta\beta' - \rho'\Delta\hat{V}'_x\,\delta\alpha'}{\mu'} \tag{8.96}$$

$$\delta\Delta V'_z = \left(k' + \frac{s'}{\rho'}\right)\Delta\hat{V}'_z - \mu'\delta\beta' \tag{8.97}$$

where $\rho' = |\Delta\hat{V}'|$, $\mu' = [\Delta\hat{V}'^2_x + \Delta\hat{V}'^2_y]^{\frac{1}{2}}$, and $\delta\alpha'$ and $\delta\beta'$ are the actual pointing angle errors.

Before operating on equations (8.95), (8.96), and (8.97) to obtain expressions for $E\left[\delta\Delta V'_j\right]$ and $\tilde{Q}'_j = E\left[\delta\Delta V'_j\,\delta\Delta V'^T_j\right]$, we shall assume that $\Delta\hat{V}'_x$, $\Delta\hat{V}'_y$, and $\Delta\hat{V}'_z$ can be replaced by the components

of the actual statistical velocity correction "$E[\Delta V_j']$". This means only k', s', $\delta\alpha'$, and $\delta\beta'$ need be treated as random variables when we apply the expectation operator to these equations or to any products of these equations.

Under the previous assumption, we obtain the following expression for the mean of $\delta\Delta V_j'$:

$$E[\delta\Delta V_j'] = E[\delta\Delta V_x']\, e_x + E[\delta\Delta V_y']\, e_y + E[\delta\Delta V_z']\, e_z \tag{8.98}$$

where

$$E[\delta\Delta V_x'] = \left(\overline{k}' + \frac{\overline{s}'}{\rho'}\right)\Delta\hat{V}_x' + \frac{\rho'\Delta\hat{V}_y'\,\overline{\delta\alpha}' + \Delta\hat{V}_x'\,\Delta\hat{V}_z'\,\overline{\delta\beta}'}{\mu'} \tag{8.99}$$

$$E[\delta\Delta V_y'] = \left(\overline{k}' + \frac{\overline{s}'}{\rho'}\right)\Delta\hat{V}_y' + \frac{\Delta\hat{V}_y'\,\Delta\hat{V}_z'\,\overline{\delta\beta}' - \rho'\Delta\hat{V}_x'\,\overline{\delta\alpha}'}{\mu'} \tag{8.100}$$

$$E[\delta\Delta V_z'] = \left(\overline{k}' + \frac{\overline{s}'}{\rho'}\right)\Delta\hat{V}_z' - \mu'\,\overline{\delta\beta}' \tag{8.101}$$

and e_x, e_y, and e_z denote three unit vectors aligned with the inertial ecliptic coordinate axes.

Denoting the elements of $\tilde{Q}_j'$ by $\tilde{Q}_{ik}'$, we will define

$$\left.\begin{aligned}
&\tilde{Q}_{11}' = E[\delta\Delta V_x'^2],\ \tilde{Q}_{22} = E[\delta\Delta V_y'^2],\ \tilde{Q}_{33} = E[\delta\Delta V_z'^2] \\
&\tilde{Q}_{12}' = \tilde{Q}_{21}' = E[\delta\Delta V_x'\,\delta\Delta V_y'] \\
&\tilde{Q}_{13}' = \tilde{Q}_{31}' = E[\delta\Delta V_x'\,\delta\Delta V_z'] \\
&\tilde{Q}_{23}' = \tilde{Q}_{32}' = E[\delta\Delta V_y'\,\delta\Delta V_z']
\end{aligned}\right\} \tag{8.102}$$

Substituting equations (8.95), (8.96), and (8.97) into (8.102) and assuming all execution error sources to be uncorrelated, yields

$$\tilde{Q}'_{11} = \xi' \, \Delta\hat{V}'^2_x + \frac{1}{\mu'^2}\left(\rho'^2 \, \Delta\hat{V}'^2_y \, \overline{\delta\alpha'\delta\alpha'} \quad \Delta\hat{V}'^2_x \, \Delta\hat{V}'^2_z \, \overline{\delta\beta' \, \delta\beta'}\right.$$

$$\left. + 2\rho' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\alpha'} \, \overline{\delta\beta'}\right) + \frac{2\Delta\hat{V}'_x}{\mu'} \zeta'\left(\rho'\Delta\hat{V}'_y \, \overline{\Delta\alpha'} + \Delta\hat{V}'_x \, \Delta\hat{V}'_z \, \overline{\delta\beta'}\right) \tag{8.103}$$

$$\tilde{Q}'_{22} = \xi' \, \Delta\hat{V}'^2_y + \frac{1}{\mu'^2}\left(\Delta\hat{V}'^2_y \, \Delta\hat{V}'^2_z \, \overline{\delta\beta' \, \delta\beta'} + \rho'^2 \, \Delta\hat{V}'^2_x \, \overline{\delta\alpha' \, \delta\alpha'}\right.$$

$$\left. - 2\rho' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\alpha'} \, \overline{\delta\beta'}\right) + \frac{2\Delta\hat{V}'_y}{\mu'} \zeta'\left(\Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta'} - \rho' \, \Delta\hat{V}'_x \, \overline{\delta\alpha'}\right)$$

(8.104)

$$\tilde{Q}'_{33} = \xi' \, \Delta\hat{V}'^2_z + \mu'^2 \, \overline{\delta\beta'\delta\beta'} - 2 \, \Delta\hat{V}'_z \, \mu' \, \zeta' \, \overline{\delta\beta'} \tag{8.105}$$

$$\tilde{Q}'_{12} = \tilde{Q}'_{21} = \xi' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y + \frac{\zeta'}{\mu'}\left[2 \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta'} - \rho'\left(\Delta\hat{V}'^2_x - \Delta\hat{V}'^2_y\right)\overline{\delta\alpha'}\right.$$

$$+ \frac{1}{\mu'^2}\left[-\rho'^2 \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \overline{\delta\alpha' \, \delta\alpha'} + \rho' \, \Delta\hat{V}'_z\left(\Delta\hat{V}'^2_y - \Delta\hat{V}'^2_x\right)\overline{\delta\alpha'} \, \overline{\delta\beta'}\right.$$

$$\left.\left. + \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta\hat{V}'^2_z \, \overline{\delta\beta' \, \delta\beta'}\right]\right] \tag{8.106}$$

$$\tilde{Q}'_{13} = \tilde{Q}'_{31} = \xi' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_z + \zeta'\left[\frac{\Delta\hat{V}'_z}{\mu'}\left(\rho' \, \Delta\hat{V}'_y \, \overline{\delta\alpha'} + \Delta\hat{V}'_x \, \Delta\hat{V}'_z \, \overline{\delta\beta'}\right) - \mu' \, \Delta\hat{V}'_x \, \overline{\delta\beta'}\right]$$

$$- \rho' \, \Delta\hat{V}'_x \, \overline{\delta\alpha'} \, \overline{\delta\beta'} - \Delta\hat{V}'_x \, \Delta\hat{V}'_z \, \overline{\delta\beta' \, \delta\beta'} \tag{8.107}$$

$$\tilde{Q}'_{23} = \tilde{Q}'_{32} = \xi' \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z + \zeta'\left[\frac{\Delta\hat{V}'_z}{\mu'}\left(\Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta'} - \rho' \, \Delta\hat{V}'_x \, \overline{\delta\alpha'}\right) - \mu' \, \Delta\hat{V}'_y \, \overline{\delta\beta'}\right]$$

$$+ \rho' \, \Delta\hat{V}'_x \, \overline{\delta\alpha'} \, \overline{\delta\beta'} - \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta' \, \delta\beta'} \tag{8.108}$$

where

$$\xi' = \overline{k'k'} + \frac{2}{\rho'}\,\overline{k'}\,\overline{s'} + \frac{\overline{s's'}}{\rho'^2} \qquad (8.109)$$

and

$$\zeta' = \overline{k'} + \frac{\overline{s'}}{\rho'} \quad . \qquad (8.110)$$

9. PROBE TARGETING, ERROR ANALYSIS, AND SIMULATION

9.1 Introduction

Main probe targeting proceeds via the same Newton-Raphson procedures as do all the other targeting options. Only the target parameters themselves are different. They are time, latitude, and longitude at impact on the target planet. The last two of these, like inclination and radius of closest approach, are treated by first transforming them into an equivalent pair of impact plane coordinates, B·T and B·R.

Miniprobe targeting, on the other hand, is essentially different from all the other targeting problems solved in NOMNAL. First, its controls are not simply three orthogonal velocity components but rather four release variables. Second, its target parameters outnumber its controls, making it necessary to settle for a minimum-miss rather than a true solution. Miniprobe targeting is therefore treated in Section 9.2 of this chapter.

Probe error analysis and simulation is based directly on the theory presented in Chapters 5, 6, and 7. The only new features are (1) the computation of an execution error covariance for the miniprobe spin release maneuver, and (2) the transformations of states and covariances at entry to the form required by the LTR* program. The details of the error analysis and the simulation of probe release are presented in the analysis sections of subroutines PROBE and PROBS, respectively. The computation of the spin release execution error covariance is treated in the analysis of MINIQ. Finally, subroutine NTRY describes the transformation of the STEAP-generated state and covariance into a form suitable for the LTR program. Thus probe error analysis and simulation require no further treatment in the current chapter.

9.2 Miniprobe Targeting

9.2.1 Introduction

The problem of finding the optimum miniprobe release controls to achieve impacts as near as possible to the respective three desired miniprobe entry sites divides naturally into three parts: (1) model formulation, (2) initial iterate generation, and (3) minimum-miss optimization. NOMNAL's treatment of each of these three areas is surveyed below. For a more detailed discussion, the *STEAP Programmer's Manual* should be consulted under the appropriate subroutine.

---

*Lander Trajectory Reconstruction.

9.2.2 Model

The following bus/miniprobe configuration is assumed. Three miniprobes are suspended on booms from the bus in such a manner that they form the vertices of an equilateral triangle centered about, and perpendicular to, the bus spin axis. These miniprobes are then assumed to be released simultaneously so their three respective velocities relative to the bus form an equilateral triangle.

Only four release control variables are assumed available to the designer in achieving the six target variables of declination and right ascension at impact for each of the three miniprobes. They are (1) the release roll angle,* (2) the tangential velocity at release, (3) the ecliptic declination of the spin axis, and (4) the ecliptic right ascension of the spin axis. Three other potential controls are (a) the release time and (b) and (c) the declination and right ascension of the impact site of the existing bus trajectory at release. Controls (2) and (a) to first order have exactly the same effect on the miniprobe impact site distribution. Hence both cannot be used in a minimum-miss algorithm based on linear sensitivities alone. On the basis of projected Planetary Explorer mission profiles, it was decided to fix the release time rather than the tangential velocity at release. The other two controls could, however, be added to the existing algorithm with only moderate difficulty. The vector of release controls will be denoted in what follows by $\underline{u}$.

To make use of the linear dependence of B·T and B·R on the release controls and to avoid the difficulty of defining a pseudo-impact point when a miniprobe misses the planet, all of the targeting is done in the impact plane. The correspondence between actual impact points on the planet and the fictitious asymptote pierce points in the R·T plane is shown pictorially in Figure 9.1. If the gravitational attraction of the target planet and the sun could be turned off at the planet's sphere of influence, the vehicle would follow a trajectory approximately coincident with the asymptote to its actual trajectory. The shortest vector from the planet to the asymptote or force-free trajectory is called the B vector. The unit vector originating at the planet center and pointing in the direction of the asymptote or hyperbolic excess

---

*The roll angle is measured about the spin axis in the direction of rotation from the vector, which is the cross-product of the spin vector with the ecliptic pole vector.

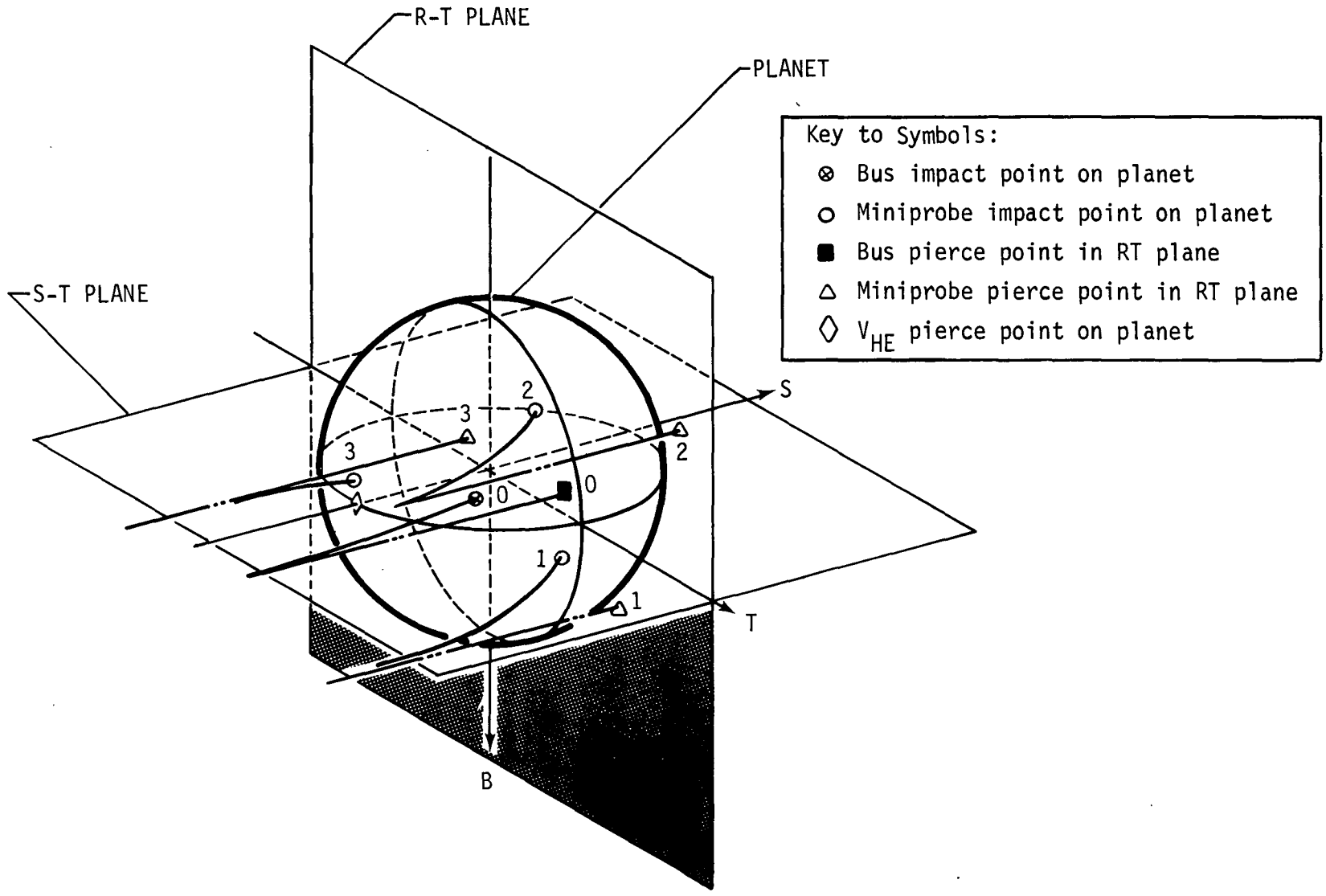


Figure 9.1 Schematic Visualization of RT Plane Miniprobe Targeting

velocity vector is called the $\underline{S}$ vector. The $\underline{T}$ vector is the cross-product of the $\underline{S}$ vector with the ecliptic pole vector $\underline{K}$, while the $\underline{R}$ vector is the cross-produce of the $\underline{S}$ vector with the $\underline{T}$ vector. Naturally the $\underline{B}$ vector lies in the R-T or "impact" plane and is conveniently resolved into the components $B\cdot T$ and $B\cdot R$ along the T and R axes respectively. Each desired miniprobe impact site is converted into a corresponding $B\cdot T_D - B\cdot R_D$ pair using the hyperbolic excess velocity vector for the miniprobe targeted to it. Next the actual B-plane coordinates, $B\cdot T_A$ and $B\cdot R_A$, are calculated. An impact-plane miss vector $\underline{\phi}$ can then be defined as

$$\underline{\phi} = \begin{bmatrix} C_1 (B\cdot T_D - B\cdot T_A)_1 \\ C_2 (B\cdot T_D - B\cdot T_A)_2 \\ C_3 (B\cdot T_D - B\cdot T_A)_3 \\ C_1 (B\cdot R_D - B\cdot R_A)_1 \\ C_2 (B\cdot R_D - B\cdot R_A)_2 \\ C_3 (B\cdot R_D - B\cdot R_A)_3 \end{bmatrix} \tag{9.1}$$

where the numeric subscripts identify the probes. The constants C_1, C_2, and C_3 are simply weighting factors assigned by the user to the respective desired miniprobe impact sites. The magnitude of a given factor should be in approximate proportion to the importance of a nearby impact to the corresponding target site. The miss index, y, can then simply be taken as the norm squared of the miss vector; i.e.,

$$y = \underline{\phi}^T \underline{\phi} \quad . \tag{9.2}$$

Two trajectory propagation models are available in the miniprobe targeting routine. The first is a high-speed conic scheme, while the second is a time-consuming virtual-mass procedure. The n-body integration computer time requirements are aggravated in miniprobe targeting by the fact that each propagation entails the integration of three separate trajectories. Hence one minimum-miss iteration on n release controls demands 3(n+1) trajectory integrations. Here 3n of the integrations are required to generate (by perturbation) the Jacobian sensitivity matrix of the miss vector with respect to the release controls, while the remaining three

are necessary to evaluate the miss vector for the new iteration. Fortunately a given set of release controls produce the same impact time, declination, right ascension, velocity, flightpath angle, and angle of attack for both models to within 1%. However, the set of minimum-miss controls for the two models differ substantially. In particular the optimal spin axis direction can differ by as much as 10° between the two models. Lucklily, when the optimal conic release controls are propagated with the n-body integrator, the miss index is only slightly larger than when the optimal virtual-mass controls are so propagated. It would appear then that the conic propagation model is more than adequate for most preliminary mission analysis work.

The conic model miniprobe propagation mode is somewhat involved. First, the virtual-mass trajectory of the bus/miniprobe combination is propagated to impact. The resulting impact state is then used to generate an osculating planetocentric conic accurately representing the near-planet trajectory. This near-planet conic is then propagated backward to the nominal time of release, and the resulting conic state is taken as the nominal release state. Hence the conic and virtual-mass trajectories are matched at impact rather than release. Velocity perturbations are then applied to the conic nominal release state and are propagated forward conically to generate sensitivity matrices. These matrices are then manipulated by the minimum-miss algorithm to yield a set of release controls that minimize the miss index.

The virtual-mass propagation model is simpler in concept than the conic scheme but more difficult in execution. All trajectories, both nominal and perturbed, are simply propagated by integrating their heliocentric state vectors. The virtual-mass method is objectionable not only because of its prohibitive running time but also because of its limited resolution. Roundoff error limits the accuracy of the sensitivity matrix of the miss vector to the release controls, thereby limiting the precision to which the controls can be made to converge.

Both of these difficulties could probably be eliminated by generating the sensitivities analytically rather than by perturbation. It seems reasonable to suspect that analytical sensitivities based on the two-body model would suffice to produce reasonable convergence in the n-body propagation. The time required to compute the sensitivies analytically would be negligible compared to that required to calculate them numerically. Hence the generation of n virtual-mass optimal release controls could be accelerated n-fold. However, with the current virtual-mass perturbative sensitivity matrix generation, the slight miss index improvement

achieved with optimal virtual-mass controls will seldom justify their increased cost in terms of computer running time.

The miniprobe propagation process is handled by the subroutine TPPROP. Given a set of release controls and a spin axis orientation mode, it generates the miss vector $\underline{\phi}$ by propagating the miniprobes to impact according to either the conic or virtual-mass models as desired.

9.2.3 Initial Control Estimate

To apply an iterative miss index-minimizing algorithm, an initial estimate of the control vector must be devised. Further, since the selected algorithm treats the miss vector $\underline{\phi}$ as a linear function of the control vector $\underline{u}$, the estimate must be in a region about the true minimum-miss point throughout which reasonable linearity prevails. For this reason an elaborate initial control estimate was designed for the conic model iteration. The converged conic model controls then serve as the initial iterate in the virtual-mass iteration.

The initial estimates for two of the controls (the spin axis orientation angles) depend of course on the spin axis orientation mode. In three of the four possible modes the ecliptic decination and right ascension of the spin axis at release are fixed rather than free controls. Hence no initial estimates for them need to be provided. In the remaining mode, however, both of these controls are free, and the initial estimates provided for them are simply those that bring the spin axis into coincidence with the bus velocity vector at release. This orientation was chosen for the initial estimate since it produces the widest distribution of miniprobe entry sites for a given combination of the remaining two controls.

An initial estimate for the other two controls (the release roll angle and the tangential velocity at release) are generated by merely targeting the first miniprobe to the miniprobe target site nearest the bus impact point for the bus trajectory existing at release. Using a single Newton-Raphson step, the release velocity increment perpendicular to the bus spin axis that would carry the first miniprobe to the nearest desired miniprobe site is approximated. From this increment, the corresponding unique pair of controls (1) and (2) is calculated. A more accurate targeting of the first miniprobe is unjustified since this preliminary targeting process ignores the remaining two miniprobes.

The initial control estimate generation is carried out entirely in the subroutine TPRTRG.

9.2.4 Minimum-Miss Algorithm

Minimizing the miss index defined in equation (9.2) is clearly a weighted least-squares optimization problem. Gauss developed an extremely efficient pseudoinverse algorithm for treating such problems when the miss vector $\underline{\phi}$ is approximately a linear function of the control vector $\underline{u}$.* The Gauss procedure, which is merely the exact one-step solution to the problem for the linear case, requires that the control correction

$$\underline{\Delta u} = - (J^T J)^{-1} J^T \underline{\phi} \tag{9.3}$$

be applied at each iteration where J is the Jacobian matrix of the miss vector with respect to the control vector; i.e.,

$$(J)ij = \frac{\partial \phi i}{\partial uj} \qquad \begin{matrix} i = 1,\ldots,m \\ j = 1,\ldots,n \\ m \geq n \end{matrix} \tag{9.4}$$

It should be evident that the Gauss least-squares procedure then degenerates to the Newton-Raphson algorithm when the number of controls, n, is equal to the number of constraints, m.

The linear transformation applied to $\underline{\phi}$ in the control correction formula (9.3) will, of course, be recognized as the left pseudoinverse of J that is known to give the minimum ϕ-norm solution to the system

$$-\underline{\phi} = J\underline{\Delta u} \quad . \tag{9.5}$$

The least-squares property of the Gauss formula is then an immediate consequence of this fact. Alternatively the correction formula can be derived directly by expanding the identity

$$\underline{\nabla y} = 2J^T \underline{\phi} \tag{9.6}$$

---

*See Reference 18.

in a Taylor series about the current control vector iterate $\underline{u}$, neglecting all second and higher order terms, and requiring that

$$\underline{\nabla}y\ (\underline{u} + \Delta\underline{u}) = 0 \quad . \tag{9.7}$$

Heuristically speaking, the Guass or pseudo-inverse scheme operates by constructing hyperplanes tangent at the current iteration point to the manifolds to constant miss for each of the components of the miss vector. The scheme then extrapolates to the "zero-miss" hyperplanes assuming that all of the manifolds of constant miss-vector components are uniformly spaced hyperplanes. Finally it solves for the unique point that is so located as to minimize the sum of the squares of its distances from the respective "zero miss" hyperplanes.

The linearity of the miniprobe targeting makes it a prime candidate for the Gauss algorithm. Nonetheless considerable nonlinearity can arise in the minimprobe release problem because of a poor original control estimate. In such a case the Gauss scheme could easily diverge because of its unjustified linear extrapolation. To cope with this problem, a best-step steepest-descent option was incorporated in the least-squares routine. This option is used when either of two situations arise indicating nonlinearity: (1) the Gauss control correction is larger in norm than some input upper bound, or (2) the Gauss step actually increases the miss index over the previous iterate.

The logic of the steepest descent scheme is straightforward. First the gradient to the miss index is calculated from the Jacobian matrix of the niss vector with respect to the control vector and the miss vector itself as

$$\underline{\nabla}y = 2J^{T}\ \underline{\phi} \quad . \tag{9.8}$$

Next a search is conducted in the negative gradient direction until the miss index is observed to begin increasing. Then a cubic polynomial is fit to the miss index, y, as a function of the step length, λ, in the search direction by exactly matching function values at $\lambda = 0$, $\lambda = a$, and $\lambda = a/2$ and slopes at $\lambda = 0$ where a is that particular step size where y is first found to increase. Then the abscissa value, λ_m, of the minimum of the fitted polynomial is computed. Finally the control correction is taken to be

$$\Delta u = -\lambda_m \underline{\nabla}y/||\underline{\nabla}y|| \quad . \tag{9.9}$$

The convergence of the scheme is only asymptotic, i.e., one can only guarantee that the minimum-miss control vector can be arbitrarily accurately approximated by taking a sufficiently large

number of iterations. Nevertheless, the steepest-descent algorithm seems to be the best available for extremely nonlinear performance indices since it involves no linear extrapolation and since it searches in the only direction in which improvement is assured. Its poor terminal convergence is no handicap in the hybrid weighted least-squares routine used in NOMNAL since once the iteration sequence falls inside a suitably linear region about the miss index minimum, the rapidly convergent Gauss scheme will take over.

The convergence criterion in either mode of the least-squares algorithm is the same. Adequate convergence is assumed when a weighted sum of the length of the change in the control vector and the length of the change in the control vector and the magnitude of the change in this miss index fall below a preassigned value, ε; i.e.,

$$C_1 \, ||\Delta \underline{u}|| + C_2 \, |\Delta y| < \varepsilon \quad . \tag{9.10}$$

With the units of u being radians or decameters/sec* and those of y being km^2, C_1 and C_2 are currently fixed inside the program at 10,000 and 1, respectively. Presently ε is input by the user with a suggested value being 1.

The entire iterative miss minimization process is conducted in the subroutine GAUSLS.

---

*These units were selected so that components of u are all of the same order of magnitude. Numerical problems can arise if this precaution is not taken.

10. INDIVIDUAL SUBROUTINE ANALYSES

This chapter is composed of three major items. In Table 10.1 all the subroutines of STEAP II are listed by category. In Table 10.2 an index of the subroutines is provided with a brief summary of the function of each subroutine, again in categorical order. The remainder and bulk of the chapter supplies the analytical documentation of each technical subroutine in alphabetical order.

Table 10.1 STEAP II Subroutine

I. Virtual-Mass Subroutines

A. Conic	B. Ephemeris	C. Propagation	D. Input/Output
1. CAREL	1. TIME	1. VMP	1. TRAPAR
2. ELCAR	2. BLOCK DATA	2. ESTMT	2. INPUTZ
3. IMPACT	3. ORB	3. VECTOR	3. PRINT
4. SOIPS	4. EPHEM	4. VMASS	4. SPACE
	5. CENTER		5. NEWPGE
	6. PECEQ		
	7. EULMX		
	8. SUBSOL		

II. NOMNAL Subroutines

A. Executive	B. Zero Iterate		C. Targeting
1. EXCUTE	1. BATCON	7. LUNTAR	1. DESENT
2. GIDANS	2. FLITE	8. MULCON	2. KTROL
3. MPPROP	3. HELIO	9. MULTAR	3. TARGET
4. NOMNAL	4. LAUNCH	10. SERIE	4. TARMAX
5. PRELIM	5. LUNA	11. ZERIT	5. TAROPT
6. TRJTRY	6. LUNCON		

D. Insertion	E. Pulsing Arc	F. Miniprobe Targeting
1. COPINS	1. (BATCON)	1. SACOCS
2. INSERS	2. PERHEL	2. TPPROP
3. NONINS	3. PREPUL	3. TRRTRG
	4. PULSEX	

G. Mathematical Functions and Operations		H. Conic		I. Ephemeris	
1. DINCOS	6. SCAD	1. CAREL	6. HYPT	1. EPHEM	
2. DINSIN	7. SCAR	2. CONCAR	7. IMPACT	2. ORB	
3. JACOB	8. THPSOM	3. DIMPCP	8. IMPCT	3. PECEQ	
4. MATIN	9. USCALE	4. ELIPT	9. SPHIMP	4. SUBSOL	
5. MATPY	10. UxV	5. HPOST	10. STIMP		

III. ERRAN and SIMUL Subroutines

A. Executive

1. ERRAN
2. SIMUL

B. Dynamic Model

1. NTM
2. NTMS
3. PSIM
4. NDTM
5. PLND
6. MUND
7. PCTM
8. CONCZ
9. CASCAD

C. Navigation

1. NAVM
2. GNAVM
3. GAIN1
4. GAIN2
5. SCHED
6. TRAKM
7. TRAKS
8. TARPRL
9. STAPRL
10. MEMO
11. MENOS
12. BIAS
13. RNUM
14. DYNO
15. DYNOS
16. GHA
17. JACOBI
18. HYEIS
19. EIGHY
20. MEAN
21. SAVMAT

D. Event

1. SETEVN
2. SETEVS
3. PRED
4. PRESIM
5. BEPS
6. BATCON
7. ZRANS
8. ATANH
9. BPLANE
10. QUASI
11. GUIDM
12. GUISIM
13. GUID
14. GUIS
15. VARADA
16. VARSIM
17. PARTL
18. BIAIM
19. POICOM
20. QCOMP
21. NONLIN
22. PULCOV
23. EXCUT
24. EXCUTS
25. PROBE
26. PROBES
27. MINIQ
28. NTRY
29. GENGID
30. ATCEGV
31. GQCOMP

E. Input/Output

1. DATA
2. DATA1
3. GDATA
4. SKEDM
5. DATAS
6. DATA1S
7. CONURT
8. TRANS
9. CORREL
10. STMPR
11. SUB1
12. TITLE
13. GPRINT
14. MOMENT
15. PRINT3
16. PRNTS3
17. PRINT4
18. PRNTS4

Table 10.2 STEAP II Subroutine Summaries

Subroutine		Function
I. Virtual-Mass Subroutines		
A. Conic		
	1. CAREL	Convert a Cartesian state to conic elements
	2. ELCAR	Convert conic elements to a Cartesian state
	3. IMPACT	Compute the impact-plane parameters
	4. SOIPS	Conically extrapolate from the nearest integration state to obtain impact data at the SOI and at the planet surface
B. Ephemeris		
	1. BLOCK DATA	Set the emphemeris constants of the gravitational bodies
	2. CENTER	Convert the states of bodies to barycentric coordinates
	3. EPHEM	Compute the inertial state of a gravitational body at a given time
	4. EULMX	Compute the rotational transformation matrix from the Euler angles
	5. ORB	Compute the orbital elements of a gravitational body at a given time
	6. PECEQ	Compute the transformation matrix from ecliptic to equatorial coordinates
	7. SUBSOL	Compute the transformation matrix from ecliptic to subsolar coordinates
	8. TIME	Convert Julian dates epoch 1900 to calendar dates or vice versa

Subroutine	Function
C. Propagation	
1. ESTMT	Determine final position and magnitude of the virtual mass on the current step
2. VECTOR	Compute the final position of the spacecraft on the current step
3. VMASS	Determine the virtual-mass data for the current step
4. VMP	Direct the virtual-mass trajectory propagation
D. Input/Output	
1. INPUTZ	Convert the input data into a form on which VMP can operate
2. NEWPGE	Print headings for each new page in VMP printout
3. PRINT	PRINT periodic trajectory-status data
4. SPACE	Space paper keeping tracking of paging
5. TRAPAR	Compute and record navigation parameter data
II. NOMNAL Subroutines	
A. Executive	
1. EXCUTE	Control the execution of a velocity-increment trajectory correction
2. GIDANS	Control the computation of a velocity-increment trajectory correction
3. MPPROP	Generate a time history of the main-probe trajectory

4.	NOMNAL	Control the generation of the nominal trajectory (main program)
5.	PRELIM	Perform preliminary data processing for NOMNAL
6.	TRJTRY	Propagate the nominal trajectory to the next guidance event

B. Zero Iterate

1.	BATCON	Propagate a conic trajectory by means of the universal conic functions
2.	FLITE	Obtain the solution to Lambert's time-of-flight equation
3.	HELIO	Compute the heliocentric phase of the interplanetary zero iterate
4.	LAUNCH	Compute the launch phase of the interplanetary zero iterate
5.	LUNA	Control lunar zero-iterate generation
6.	LUNCON	Generate a patched conic lunar trajectory
7.	LUNTAR	Control the patched conic targeting
8.	MULCON	Generate the lunar multiconic trajectory
9.	MULTAR	Control the lunar multiconic targeting
10.	SERIE	Compute the universal conic functions used in FLITE
11.	ZERIT	Control the computation of the zero iterate

C. Targeting

1.	DESENT	Compute the interplanetary velocity targeting corrections using the descent scheme

2.	KTROL	Compute the heliocentric ecliptic velocity corrections given the launch-planetocentric velocity controls
3.	TARGET	Control the n-body targeting
4.	TARMAX	Compute the Newton-Raphson targeting matrix
5.	TAROPT	Set up the actual and auxiliary target parameter arrays

D. Insertion

1.	COPINS	Compute the coplanar orbit insertion maneuver
2.	INSERS	Control the orbit insertion computation
3.	NONINS	Compute the nonplanar orbit insertion

E. Pulsing Arc

1.	(BATCON)	Propagate a conic trajectory by means of the universal conic functions
2.	PERHEL	Propagate a perturbed heliocentric conic
3.	PREPUL	Perform the preliminary data processing for a multiple-pulse trajectory correction
4.	PULSEX	Execute pulsing arc

F. Miniprobe Targeting

1.	SAOCS	Compute the sines and cosines of the spin-axis right ascension and declination given the spin-axis orientation mode
2.	TPPROP	Propagate the three miniprobe trajectories according to either a conic or virtual-mass model

3.	TPRTRG	Control the miniprobe targeting procedure

G. Mathematical Functions and Operations

1.	DINCOS	Calculate in degrees the inverse cosine of a real number
2.	DINSIN	Calculate in degrees the inverse sine of a real number
3.	JACOB	Approximate by divided differences the Jacobian sensitivity matrix of a vector-valued function with respect to a vector variable
4.	MATIN	Invert a matrix of real-valued elements
5.	MATPY	Multiply two matrices of real-valued elements
6.	SCAD	Calculate both the sine and cosine of an angle given in degrees
7.	SCAR	Calculate both the sine and cosine of an angle given in radians
8.	THPOSM	Find the minimum of a function on a given interval by cubic interpolation
9.	USCALE	Scale the length of a three-vector to a specified real number
10.	UXV	Calculate the vector product of two three-vectors

H. Conic

1.	CAREL	Convert a Cartesian state to conic elements
2.	CONCAR	Convert a conic state in terms of r, θ, e, $\underline{P}$, $\underline{Q}$, and μ into a Cartesian state

3.	DIMPCP	Calculate the desired B-plane asymptote pierce-point coordinates given the right ascension and declination of a probe target site
4.	ELIPT	Calculate the time from periapsis on an elipse given the true anomaly
5.	HPOST	Calculate the radius and true anomaly on a hyperbola given the time from periapsis
6.	HYPT	Calculate the time from periapsis on a hyperbola given the true anomaly
7.	IMPACT	Compute the impact-plane parameters
8.	IMPCT	For auxiliary targeting compute actual and desired B-plane asymptote pierce points as well as actual target values
9.	SPHIMP	Calculate the true anomaly and time from periapsis at which a conic approach trajectory pierces a planetocentric sphere of a given radius
10.	STIMP	Calculate the B-plane asymptote pierce-point coordinates of a conic trajectory given a state upon it

I. Ephemeris

1.	EPHEM	Compute the inertial state of a gravitational body at a given time
2.	ORB	Compute the orbital elements of a gravitational body at a given time
3.	PECEQ	Compute the transformation matrix from ecliptic to equatorial coordinates
4.	SUBSOL	Compute the transformation matrix from ecliptic to subsolar coordinates

III. ERRAN and SIMUL Subroutines

A. Executive

1.	ERRAN	Control error analysis program (main program)
2.	SIMUL	Control simulation program (main program)

B. Dynamic Model

1.	NTM	Control generation of trajectory data for ERRAN
2.	NTMS	Control generation of trajectory data for SIMUL
3.	PSIM	Control computation of state transition matrix (STM)
4.	NDTM	Compute unaugmented partition of STM by numerical differencing
5.	PLND	Compute STM partition associated with ephemeris biases
6.	MUND	Compute STM partition associated with gravitational constants
7.	PCTM	Compute unaugmented partition of STM by patched conic technique
8.	CONC2	Compute unaugmented partition of STM by virtual-mass technique
9.	CASCAD	Compute unaugmented partition of STM by cascaded Darby matrizants

C. Navigation

1.	NAVM	Propagate covariance matrices between measurements and between events in SIMUL
2.	GNAVM	Propagate assumed and actual covariance matrices between measurements and between events in ERRAN

3.	GAIN1	Compute the Kalman GAIN matrices
4.	GAIN2	Compute the GAIN matrices for the equivalent recursive consider weighted-least-squares filter
5.	SCHED	Select next measurement time from measurement schedule
6.	TRAKM	Compute observation matrices
7.	TRAKS	Compute observation matrices and actual measurements
8.	TARPRL	Compute target planet position partials
9.	STAPRL	Compute station location position and velocity partials
10.	MENO	Compute assumed measurement noise covariance matrix
11.	MENOS	Compute assumed and actual measurement noise covariance matrices
12.	BIAS	Compute actual measurement bias
13.	RNUM	Generate random numbers
14.	DYNO	Compute dynamic noise covariance matrix
15.	DYNOS	Compute dynamic noise covariance matrix and actual dynamic noise
16.	GHA	Compute Greenwich hour angle
17.	JACOBI	Compute eigenvalues and eigenvectors of a matrix
18.	HYELS	Compute hyperellipsoids
19.	EIGHY	Control computation of eigenvalues, eigenvectors, and hyperellipsoids

20.	MEAN	Propagate and update means of actual state or parameter deviations and actual state of parameter estimation errors
21.	SAVMAT	Stores one vector in a second vector

D. Event

1.	SETEVN	Perform computations common to most events in ERRAN
2.	SETEVS	Perform computations common to most events in SIMUL
3.	PRED	Perform prediction event in ERRAN
4.	PRESIM	Perform prediction event in SIMUL
5.	BEPS	Compute B-Plane-Related covariances and state transition matrices
6.	BATCON	Compute trajectory data at time T given position and velocity at time 0
7.	ZRANS	Calculate transcendental functions used in the universal form of Kepler's equation
8.	ATANH	Find the angle Y whose TANH is X
9.	BPLANE	Compute B-plane parameters
10.	QUASI	Perform quasi-linear filtering event in SIMUL
11.	GUIDM	Perform guidance event in ERRAN
12.	GUISIM	Perform guidance event in SIMUL
13.	GUID	Compute guidance and variation matrices in ERRAN
14.	GUIS	Compute guidance and variation matrices in SIMUL
15.	VARADA	Compute 3VBP variation matrix in ERRAN

16.	VARSIM	Compute 3VBP variation matrix in SIMUL
17.	PARTL	Compute partials of B•T, B•R, wrt state
18.	BIAIM	Perform biased aimpoint guidance
19.	POICOM	Compute probability of impact
20.	QCOMP	Compute execution error covariance matrix
21.	NONLIN	Control execution of nonlinear guidance events
22.	PULCOV	Propagate covariance matrix across a series of pulses
23.	EXCUT	Control execution of pulsing arc in ERRAN
24.	EXCUTS	Control execution of pulsing arc in SIMUL
25.	PROBE	Control execution of probe release events in ERRAN
26.	PROBES	Control execution of probe release events in SIMUL
27.	MINIQ	Compute execution error covariance matrix for miniprobe release
28.	NTRY	Compute entry parameters, covariance, and communication angle
29.	GENGID	Generalized covariance technique applied to guidance processes
30.	ATCEGV	Compute eigenvalues and eigenvectors of actual target condition 2nd moment matrices
31.	GQCOMP	Compute actual execution error statistics

E. Input/Output

1.	DATA	Perform preliminary computations and read data in ERRAN
2.	DATA1	Continuation of DATA
3.	GDATA	Initialized generalized covariance quantities
4.	SKEDM	Set up bus, main probe, and miniprobe measurement schedules
5.	DATAS	Perform preliminary computations and read data in SIMUL
6.	DATAS1	Continuation of DATAS
7.	CONVRT	Convert JPL injection conditions to Cartesian components
8.	TRANS	Compute coordinate transformations
9.	CORREL	Compute and print correlation matrix partitions and standard deviations
10.	STMPR	Print STM partitions
11.	SUB1	Compute position and velocity magnitudes
12.	TITLE	Print titles
13.	GPRINT	Print actual estimation error statistics
14.	MOMENT	Convert 2nd moment matrices to correlation matrices and print them
15.	PRINT3	Print basic cycle data in ERRAN
16.	PRNTS3	Print ERRAN summary
17.	PRINT4	Print basic cycle data in SIMUL
18.	PRNTS4	Print SIMUL summary

INDIVIDUAL SUBROUTINE ANALYSES

BATCON Analysis

BATCON is a conic propagator using the Battin universal variable formulation. A total derivation is too involved to be given here; rather the results of Battin's work will be given here.

Let the initial state of a point mass moving under the influence of a gravitational force μ be given by $\vec{r}_o$, $\vec{v}_o$. It is required to determine the state $\vec{r}$, $\vec{v}$ at a time T units later. It is useful to introduce the parameters

$$\sigma_o = \frac{\vec{r}_o \cdot \vec{v}_o}{\sqrt{\mu}}$$
$$\alpha = \frac{2}{r_o} - \frac{v_o^2}{\mu} \qquad (1)$$

Battin's approach is to introduce a new independent variable x(t) in place of time by the relation

$$\frac{dx}{dt} = \frac{\sqrt{\mu}}{r(t)} \qquad x(0) = 0 \qquad (2)$$

This parametrization greatly simplifies the conic propagation problem. For suppose that the value of x corresponding to t = T is given by X, i.e. x(T) = X. Then the final state is given by

$$\vec{r} = R_1(X)\,\vec{r}_o + R_2(X)\,\vec{v}_o$$
$$\vec{v} = V_1(X)\,\vec{r}_o + V_2(X)\,\vec{v}_o \qquad (3)$$

where

$$R_1(X) = 1 - \frac{1}{r_o} U_2(X) \qquad R_2(X) = \frac{1}{\sqrt{\mu}} \left[r_o U_1(X) + \sigma_o U_2(X) \right] \qquad (4)$$
$$V_1(X) = -\frac{\sqrt{\mu}}{r_o \dot{r}_o} U_1(X) \qquad V_2(X) = 1 - \frac{1}{r_o} U_2(X)$$

and where

$$U_o(X) = \cos \alpha X \quad \alpha > 0 \qquad = \cosh \sqrt{-\alpha}\, X \quad \alpha < 0$$

$$U_1(X) = \frac{\sin\sqrt{\alpha}\, x}{\sqrt{\alpha}} \quad \alpha > 0 \qquad \frac{\sinh\sqrt{-\alpha}\, X}{\sqrt{-\alpha}} \quad \alpha < 0 \tag{5}$$

$$U_2(X) = \frac{1-U_o(X)}{\alpha} \qquad U_3(X) = \frac{X-U_1(X)}{\alpha}$$

The problem is thus reduced to the determination of X. X is generated iteratively by the recursive formulae

$$x_{n+1} = x_n - \frac{\sqrt{\mu}\, t_n - \sqrt{\mu}\, t}{r_n} = x_n - \Delta x \tag{6}$$

where

$$\sqrt{\mu}\ t_n = r_o U_1(x_n) + \sigma U_2(x_n) + U_3(x_n)$$

$$r_n = r_o U_o(x_n) + \sigma U_1(x_n) + U_2(x_n) \tag{7}$$

To start the process the initial guess is set to

$$x_o = \frac{\sqrt{\mu}\, T}{r_o} \left\{ 1 - \frac{\sigma_o}{2r_o^2}\sqrt{\mu}\, T + \frac{1}{6r_o^4}\left[3\sigma_o^2 - r_o(1-\alpha r_o) \right] \mu T^2 \right\} \tag{8}$$

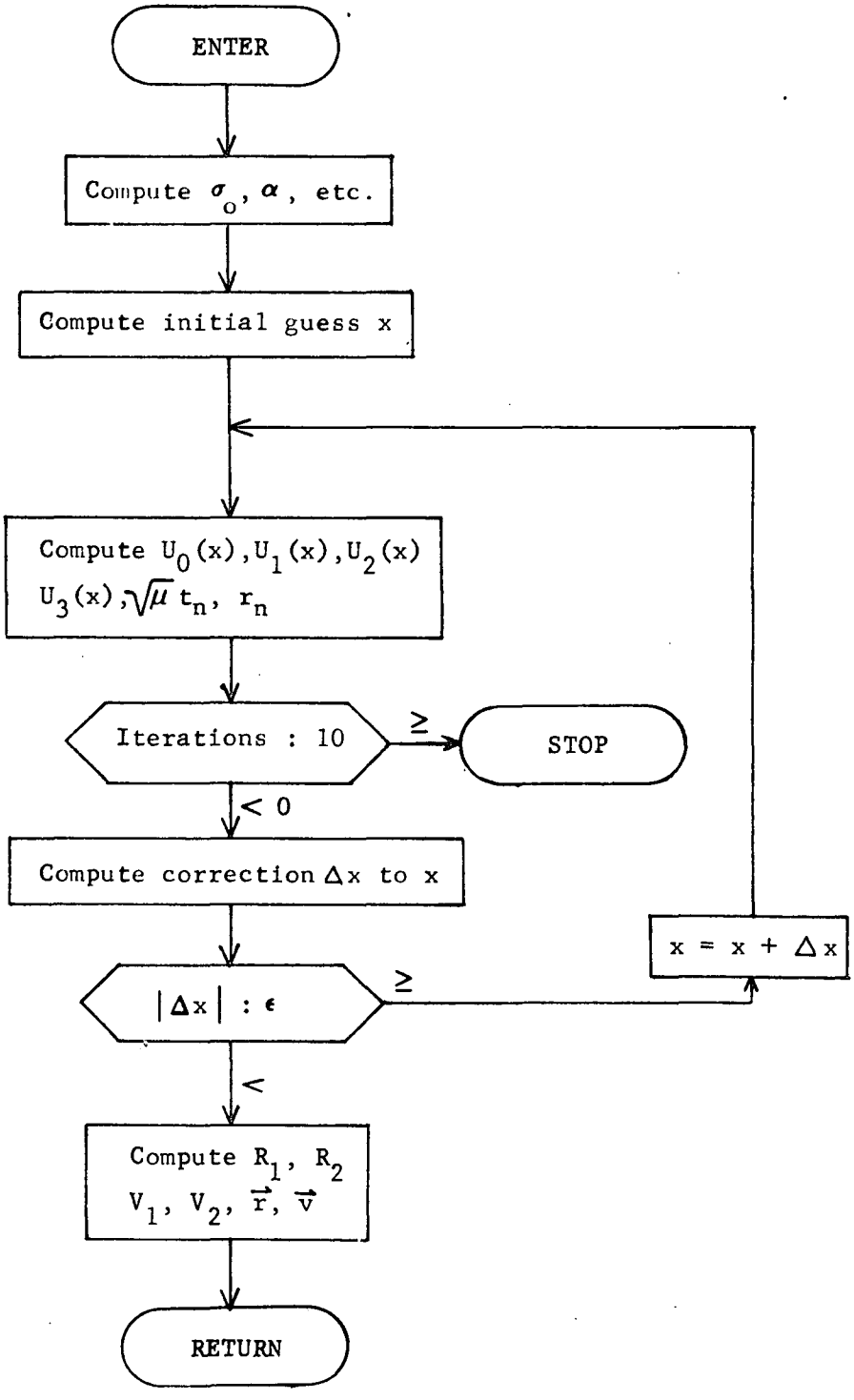
The program sets $X = x_n$ when the correction Δx is less than 10^{-8}. It terminates if the number of iterations exceeds 10.

References:

Battin, R.H., Astronautical Guidance, McGraw-Hill Book Co., New York, 1964.

Battin, R. H. and Fraser, D.C., Space Guidance and Navigation, AIAA Professional Study Series, 1970.

BATCON Flowchart



BIAIM Analysis

Subroutine BIAIM performs biased aimpoint guidance computations. If planetary quarantine constraints are in effect at injection or at a midcourse correction, and if the nominal aimpoint does not satisfy these constraints, subroutine BIAIM will compute a biased aimpoint and the required bias velocity correction such that the constraints are satisfied and some performance functional is minimized.

Aimpoint biasing is performed in the impact plane and as such permits only two degrees of freedom in the selection of the biased aimpoint. The general aimpoint in the impact plane will be denoted by the 2-dimensional vector $\vec{\mu}_j$, where the j-subscript indicates that the biased aimpoint guidance event is occurring at time t_j. Three midcourse guidance policies are available in STEAP, and it will be necessary to relate $\vec{\mu}_j$ to the specific aimpoint for each of these three policies. These relationships are summarized below:

(a) Two-variable B-plane (2VBP):

$$\vec{\mu}_j = \begin{bmatrix} B \cdot T \\ B \cdot R \end{bmatrix} \tag{1}$$

(b) Three-variable B-plane (3VBP):

$$\vec{\mu}_j = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} B \cdot T \\ B \cdot R \\ t_{SI} \end{bmatrix} \tag{2}$$

(c) Fixed-time-of-arrival (FTA):

$$\vec{\mu}_j = A\,\vec{r}_{CA} \tag{3}$$

where $\vec{r}_{CA}$ is the nominal closest approach position of the spacecraft relative to the target planet. Coordinate transformation A projects the 3-dimensional vector $\vec{r}_{CA}$ (referred to ecliptic coordinates) into an equivalent FTA impact plane which is defined to be the plane containing $\vec{r}_{CA}$ and perpendicular to the spacecraft closest approach velocity $\vec{v}_{CA}$ relative to the target planet. If the ecliptic coordinates of $\vec{r}_{CA}$ and $\vec{v}_{CA}$ are denoted by r_x, r_y, r_z and v_x, v_y, v_z, respectively, then the transformation A is given by

$$A = \begin{bmatrix} \frac{r_x}{r_{CA}} & \frac{r_y}{r_{CA}} & \frac{r_z}{r_{CA}} \\ \frac{r_y v_z - r_z v_y}{r_{CA} v_{CA}} & \frac{r_z v_x - r_x v_z}{r_{CA} v_{CA}} & \frac{r_x v_y - r_y v_x}{r_{CA} v_{CA}} \end{bmatrix} \tag{4}$$

Spacecraft state variations at t_j are related to aimpoint variations (target condition variations) by the variation matrix η_j, which is always available prior to calling BIAIM. Thus, the statistical state dispersions about the nominal following the guidance correction at t_j and represented by the control covariance $P_{c_j}^+$, can be related to the dispersions about the nominal aimpoint represented by W_j^+ according to the equation

$$W_j^+ = \eta_j P_{c_j}^+ \eta_j^T \tag{5}$$

The control covariance $P_{c_j}^+$ is computed from

$$P_{c_j}^+ = P_{k_j}^- + \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \tilde{Q}_j \end{array}\right] \tag{6}$$

where $P_{k_j}^-$ is the knowledge covariance prior to the guidance event and $\tilde{Q}_j$ is the execution error covariance.

Transformations employed in equations (1) through (3) can also be employed to project W_j^+ into the impact plane. The resulting projection is denoted by the covariance Λ_j, and is obtained from W_j^+ according to the following equations:

(a) 2VBP : $$\Lambda_j = W_j^+ \tag{7}$$

(b) 3VBP : $$\Lambda_j = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} W_j^+ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{8}$$

(c) FTA : $$\Lambda_j = A W_j^+ A^T \tag{9}$$

With covariance Λ_j available, it is now possible to compute the probability of impact PØI. Assuming the probability density function associated with Λ_j is gaussian and nearly constant over the target planet capture area permits us to compute PØI using the equation

$$P\emptyset I = \pi R_c^2 \, p \tag{10}$$

where R_c is the target planet capture radius and p represents the gaussian density function evaluated at the target planet center and is given by

$$p = \frac{1}{2\pi \left|\Lambda_j\right|^{\frac{1}{2}}} \exp\left[-\frac{1}{2} \vec{\mu}^{*T} \Lambda_j^{-1} \vec{\mu}^{*}\right] \tag{11}$$

The nominal impact plane aimpoint is denoted by $\vec{\mu}^{*}$. Subroutine BIAIM calls subroutine PØICØM to perform the computations involved in equations (7) through (11).

Capture radius R_c is simply the physical radius R_p of the target planet if the FTA guidance policy is employed, while for the two B-plane policies the capture radius is given by

$$R_c = R_p \sqrt{1 + \frac{2\mu_p}{V_\infty^2 R_p}} \tag{12}$$

where μ_p is the target planet gravitational constant and V_∞ is the hyperbolic excess velocity.

If the probability of impact PØI does not exceed the permissible impact probability P_I, and if the nominal aimpoint has not been previously biased, we simply return to subroutine GUIDM (or GUISIM). If the nominal aimpoint has been previously biased, a velocity correction $\Delta\vec{V}_{RB_j}$ required to remove that bias is computed prior to returning. But if PØI exceeds P_I, an aimpoint bias $\delta\vec{\mu}_j$ and the associated bias velocity correction $\Delta\vec{V}_{B_j}$ must be computed. Before describing the details of the biasing technique it is necessary to define the relationship between $\Delta\vec{V}_j$ and $\delta\vec{\mu}_j$ for linear midcourse guidance policies.

Linear impulsive guidance policies have form

$$\Delta\vec{V}_j = \Gamma_j \, \delta\vec{X}_j \tag{13}$$

where Γ_j is the guidance matrix and $\delta\vec{X}_j$ is the spacecraft state deviation from the targeted nominal trajectory. (These guidance policies are discussed in more detail in the subroutine GUIS analysis section.) Such guidance policies can be readily generalized to account for changes in the target conditions from their nominal values. This generalized version of equation (13) has form

$$\Delta\vec{V}_j = \Gamma_j \, \delta\vec{X}_j + \Psi_j \, \delta\vec{\mu}_j \tag{14}$$

where Ψ_j can also be referred to as a guidance matrix. For the purposes of the BIAIM analysis, we shall assume that $\delta\vec{\mu}_j$ in equation (14) is always an aimpoint change in the impact plane. Thus, Ψ_j will be a 3x2 guidance matrix. The derivation of the Ψ_j matrix is quite similar to the derivation of the Γ_j matrix and will not be presented here. If we partition the previously discussed variation matrix η_j as follows:

$$\eta_j = \left[\eta_1 \,\vdots\, \eta_2\right] \tag{15}$$

then the Ψ_j matrices for the three midcourse guidance policies are given by the following equations:

(a) 2VBP :
$$\Psi_j = \eta_2^T \left(\eta_2 \, \eta_2^T\right)^{-1} \tag{16}$$

(b) 3VBP :
$$\Psi_j = \eta_2^{-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{17}$$

(c) FTA :
$$\Psi_j = \eta_2^{-1} \, A^T \tag{18}$$

If an aimpoint bias were to be removed at time t_j, the required velocity correction would be given by

$$\Delta\vec{V}_{RB_j} = -\Psi_j \, \delta\vec{\mu}_j \tag{19}$$

If an aimpoint bias were to be imparted at time t_j, the bias velocity correction would be given by

$$\Delta\vec{V}_{B_j} = \Psi_j \, \delta\vec{\mu}_j \tag{20}$$

If an aimpoint bias $\delta\vec{\mu}_j^{(1)}$ had been previously imparted, and if a new aimpoint bias $\delta\vec{\mu}_j^{(2)}$ is to be imparted, then the total bias velocity correction would be given by

$$\Delta\vec{V}_{B_j} = \Psi_j \left[\delta\vec{\mu}_j^{(2)} - \delta\vec{\mu}_j^{(1)} \right] \tag{21}$$

The general statement of the biased aimpoint guidance problem is as follows: Find an aimpoint $\vec{\mu}_j$ in the impact plane which satisfies the impact probability constraint

$$P\emptyset I \leq P_I \tag{22}$$

and minimizes a performance functional having form

$$J = (\vec{\mu}_j - \vec{\mu}^*)^T \tilde{A} (\vec{\mu}_j - \vec{\mu}^*) \tag{23}$$

where $\vec{\mu}^*$ is the nominal aimpoint and $\tilde{A}$ is a constant symmetric matrix that will be defined subsequently.

The solution of this problem is detailed in the section on biased aimpoint guidance in the analytical manual. Only the results will be presented here. The assumption of constant probability density over the target planet capture area permits us to rewrite constraint equation (22) as

$$\lambda_1 \mu_1^2 + 2\lambda_3 \mu_1 \mu_2 + \lambda_2 \mu_2^2 = C^2 \tag{24}$$

where

$$C^2 = 2 \ln \left[\frac{R_c^2}{2 |\Lambda|^{\frac{1}{2}} P_I} \right] \tag{25}$$

and $\vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$ and $\Lambda^{-1} = \begin{bmatrix} \lambda_1 & \lambda_3 \\ \lambda_3 & \lambda_2 \end{bmatrix}$.

The inequality has been replaced by an equality since the solution can be shown to lie on the constraint boundary, which, from inspection of equation (24) is an ellipse centered at the target planet.

If t_j is the time of the final midcourse correction, matrix $\tilde{A}$ will be chosen as a 2x2 identity matrix. The minimization of J is then equivalent to minimization of the miss distance $|\vec{\mu}_j - \vec{\mu}^*|$. If t_j is not the final midcourse correction time, $\tilde{A}$ will be defined as follows:

$$\tilde{A} = \Psi_{j+1}^{T} \Psi_{j+1} \tag{26}$$

Here Ψ_{j+1} denotes the aimpoint guidance matrix for the next midcourse correction occurring at time t_{j+1}. In this case the minimization of J is equivalent to the minimization of $\left| \Delta\vec{V}_{RB_{j+1}} \right|$, i.e., the velocity required to remove bias $\delta\vec{\mu}_j$ at time t_{j+1} will by minimized. The computation of Ψ_{j+1} is based on the variation matrix η_{j+1}, just as Ψ_j was based on η_j. However, η_{j+1} can be computed more efficiently by using the relationship

$$\eta_{j+1} = \eta_j \Phi_{j+1,\,j}^{-1} \tag{27}$$

where $\Phi_{j+1,j}$ is the state transition matrix over $\left[t_j,\ t_{j+1} \right]$.

If we define

$$\tilde{A} = \begin{bmatrix} a_1 & a_3 \\ a_3 & a_2 \end{bmatrix}$$

then the necessary condition for a minimum is given by

$$(a_1\lambda_3 - a_3\lambda_1)\mu_1^2 + (a_3\lambda_2 - a_2\lambda_3)\mu_2^2 + (a_1\lambda_2 - a_2\lambda_1)\mu_1\mu_2$$

$$+ (-a_1\lambda_3\mu_1^* - a_3\lambda_3\mu_2^* + a_3\lambda_1\mu_1^* + a_2\lambda_1\mu_2^*)\mu_1 +$$

$$(-a_1\lambda_2\mu_1^* - a_3\lambda_2\mu_2^* + a_3\lambda_3\mu_1^* + a_2\lambda_3\mu_2^*)\mu_2 = 0 \tag{28}$$

Thus, our problem is reduced to finding μ_1 and μ_2 which satisfy equations (24) and (28). Since the analytical solution of these equations proved intractable, a standard Newton iteration technique is employed in BIAIM which quickly converges to solutions for μ_1 and μ_2. The iteration process is started with an initial guess defined as the intersection of the extended $\vec{\mu}^*$ vector and the constraint boundary defined by equation (24). This initial guess is given by

$$\mu_1^{o} = \left(\frac{\mu_1^{*}}{\mu_2^{*}} \right) \mu_2^{o}$$

$$\mu_2^{o} = \operatorname{sgn}(\mu_2^{*}) \frac{c}{\sqrt{\lambda_1 \left(\frac{\mu_1^{*}}{\mu_2^{*}} \right)^2 + 2\lambda_3 \left(\frac{\mu_1^{*}}{\mu_2^{*}} \right) + \lambda_2}} \tag{29}$$

where c is defined by equation (25).

In addition to the previously described iteration process, subroutine BIAIM also employs an outer iteration loop which accounts for the dependence of $\tilde{Q}_j$ (equation (6)) on $\delta\vec{\mu}_j$. The execution error covariance $\tilde{Q}_j$ is a function of the total velocity correction at t_j, but the total velocity correction, in particular $\Delta\vec{V}_{B_j}$, depends on $\delta\vec{\mu}_j$. This coupling is resolved by recomputing $\tilde{Q}_j$ at the end of the previously described biasing technique and repeating the biasing cycle until the error function

$$\left| P\emptyset I - P_I \right| \leq P_I \times 10^{-3}$$

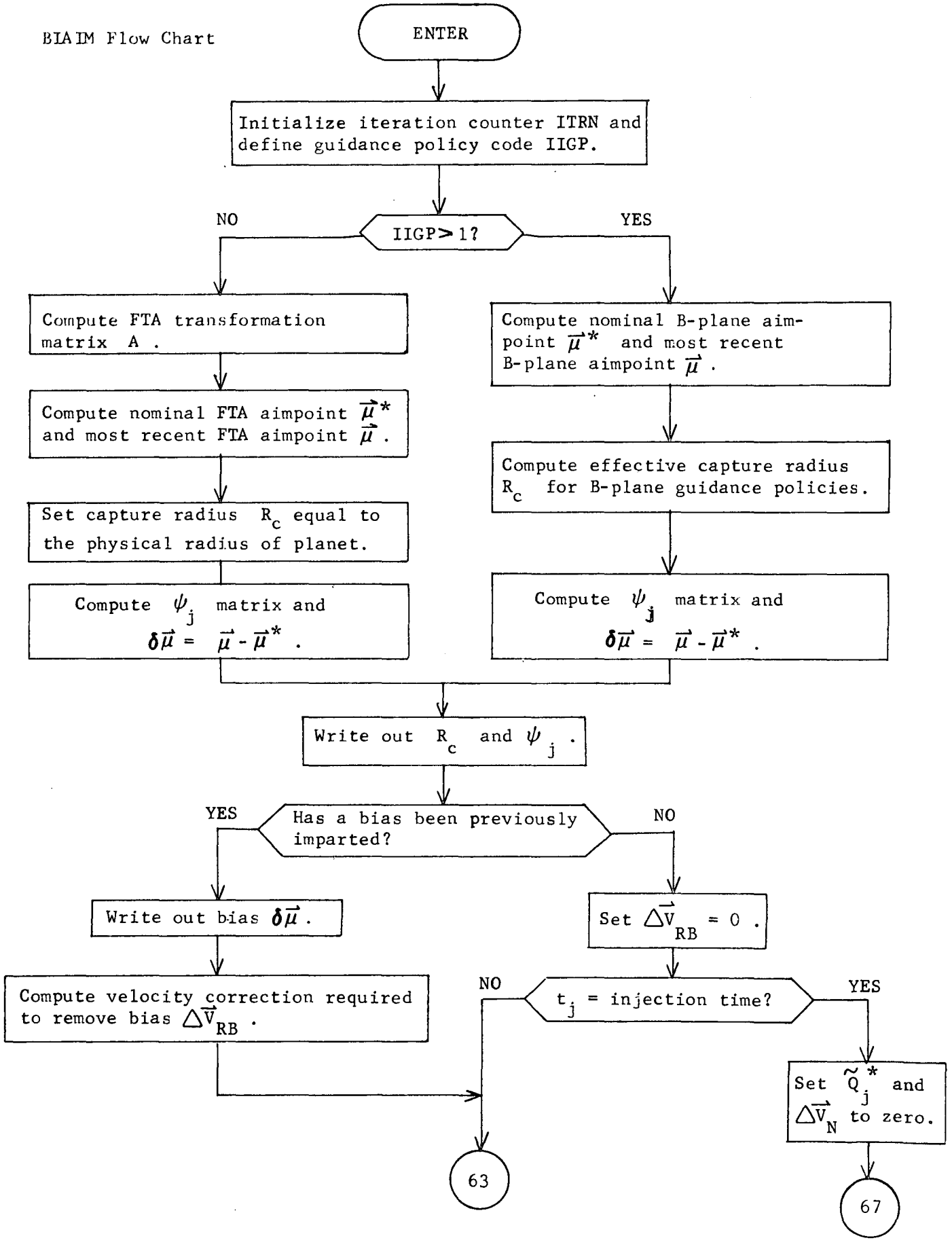
is satisfied. This outer iteration process is not performed, however, if t_j = injection time since at injection equation (6) is replaced by the equation

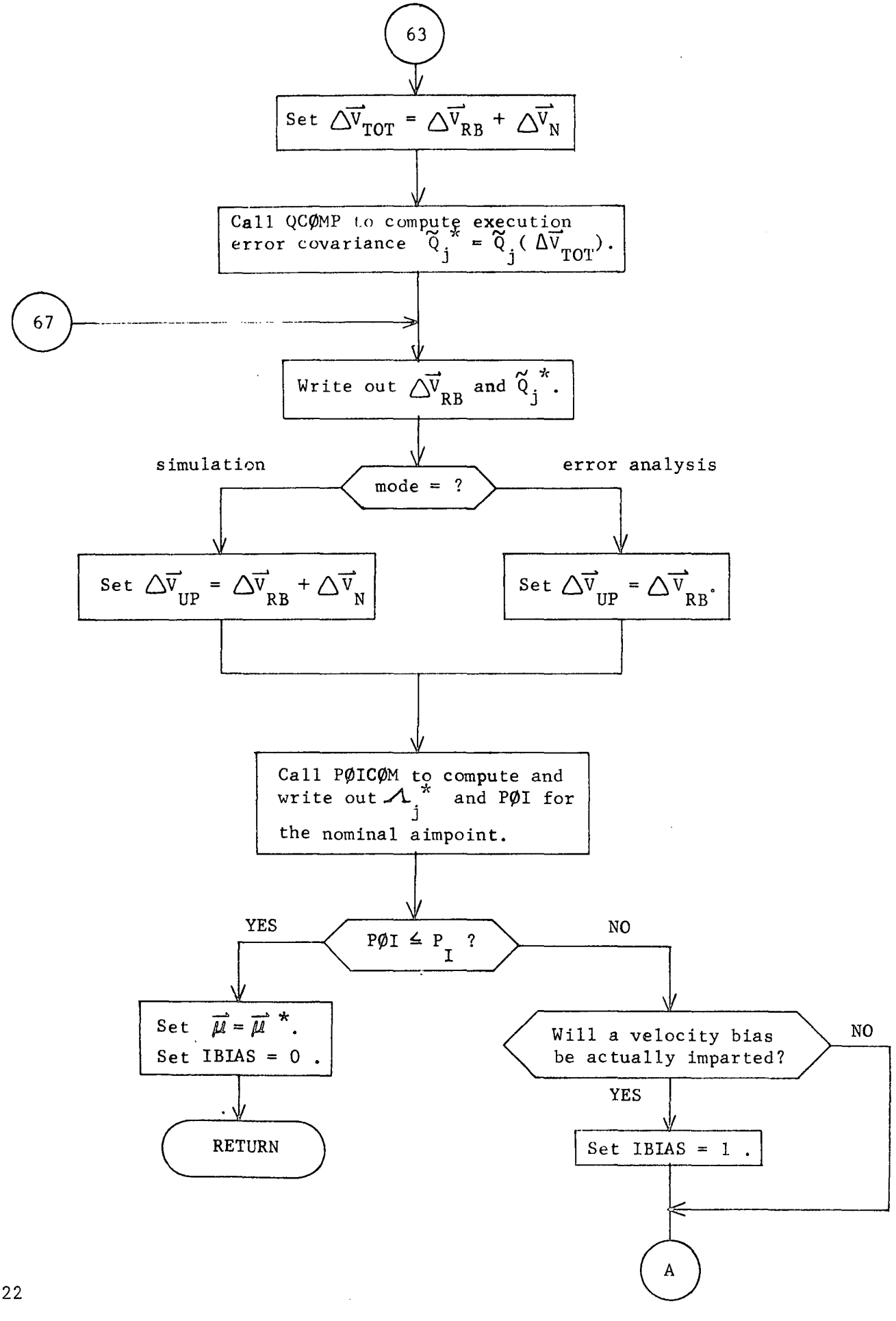
$$P_{c_j} = P_{k_j}$$

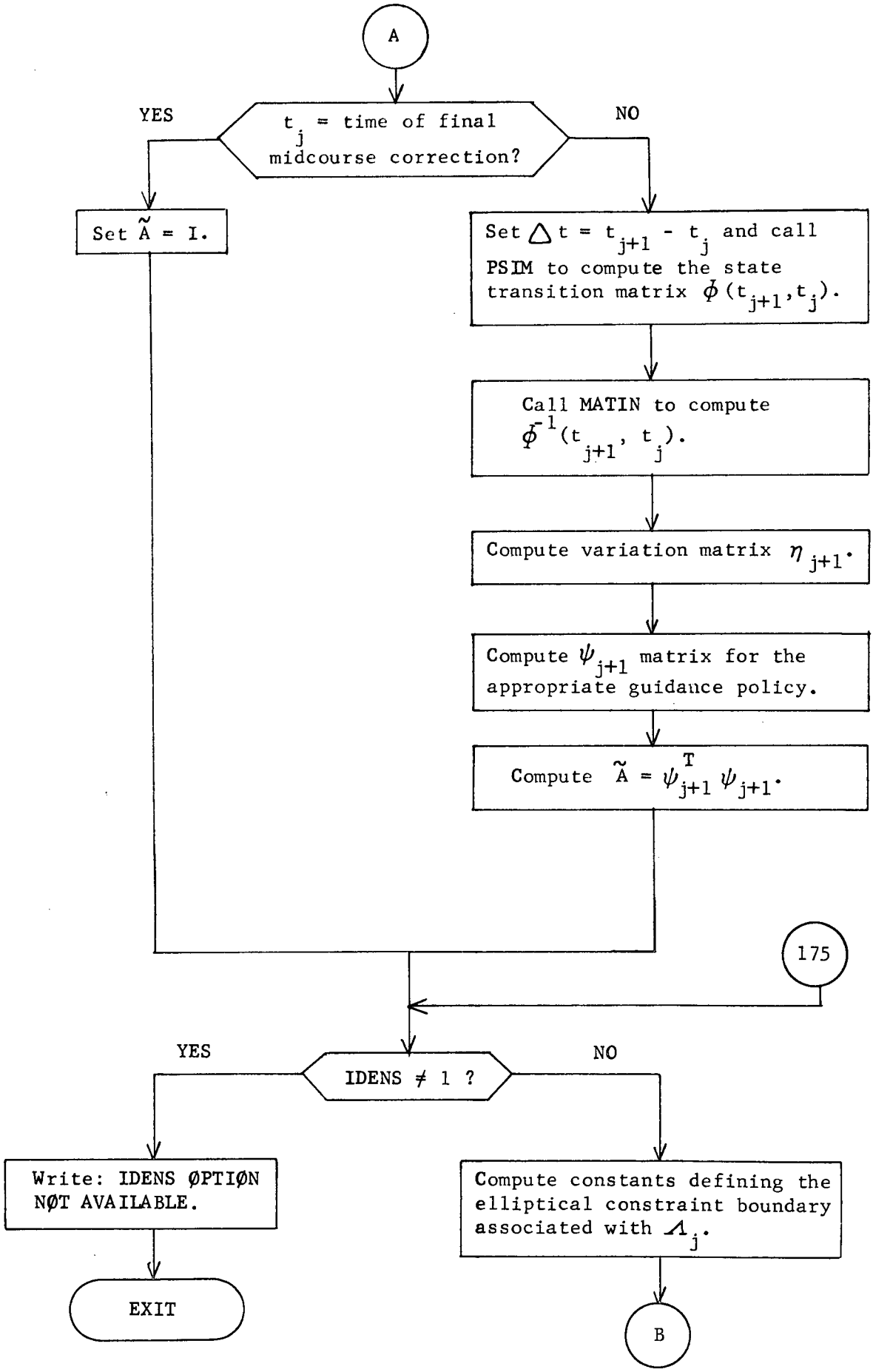
and $\tilde{Q}_j$ is always zero.

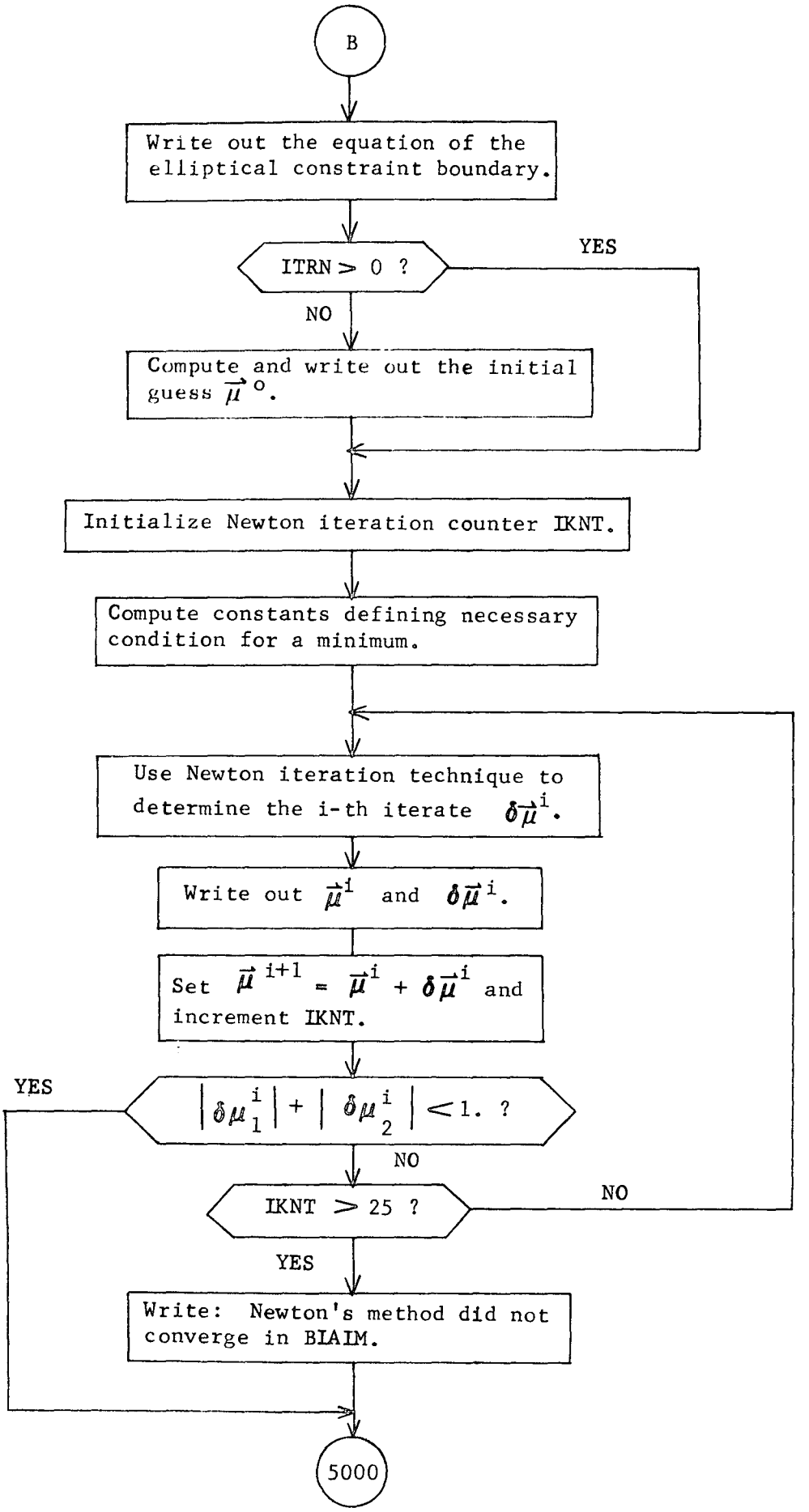
Reference: Mitchell, R. T., and Wong, S. K.: Preliminary Flight Path Analysis Orbit Determination and Maneuver Strategy Mariner Mars 1969. Project Document 138, Jet Propulsion Laboratory, 1968.

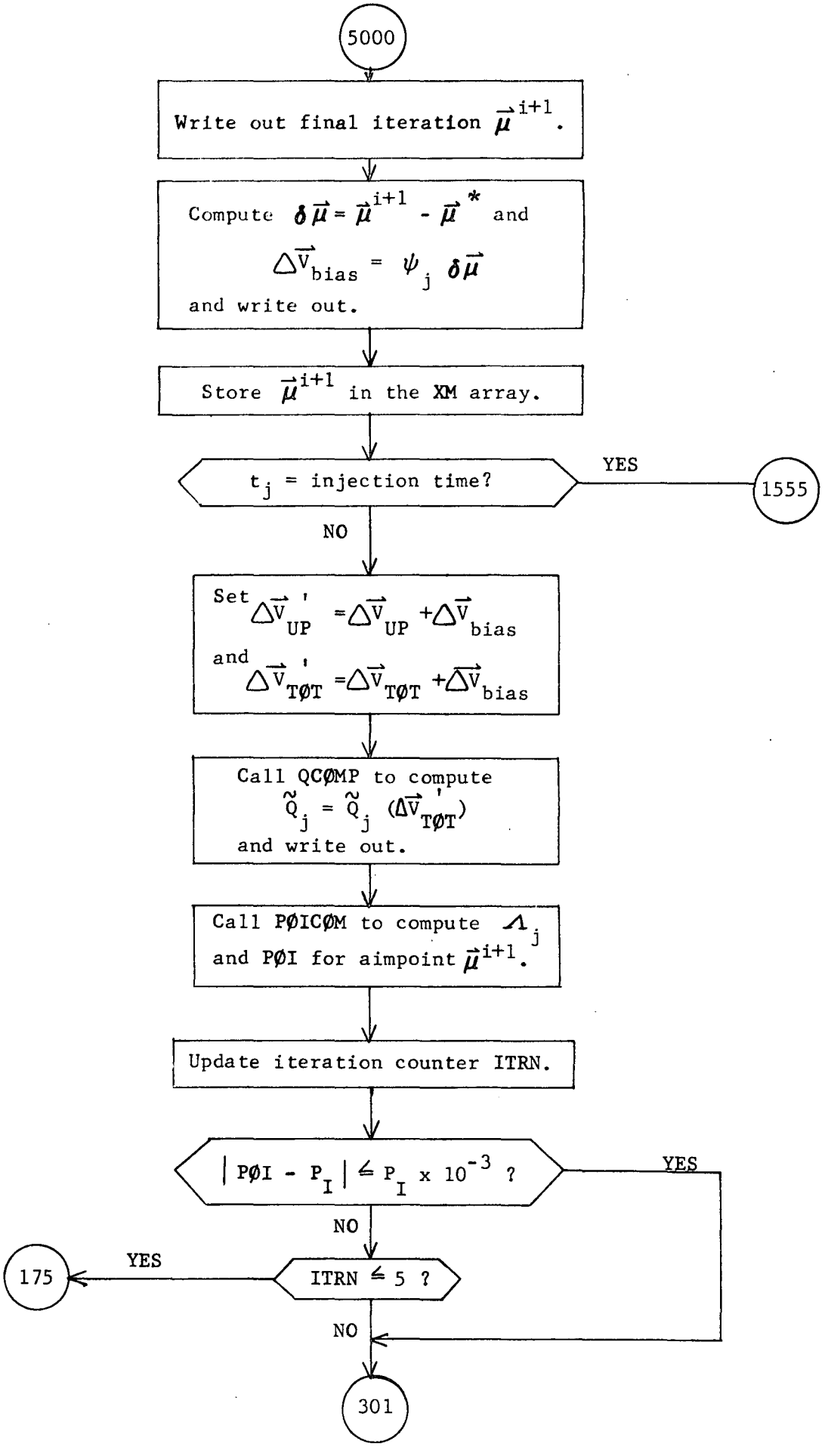
BIAIM Flow Chart

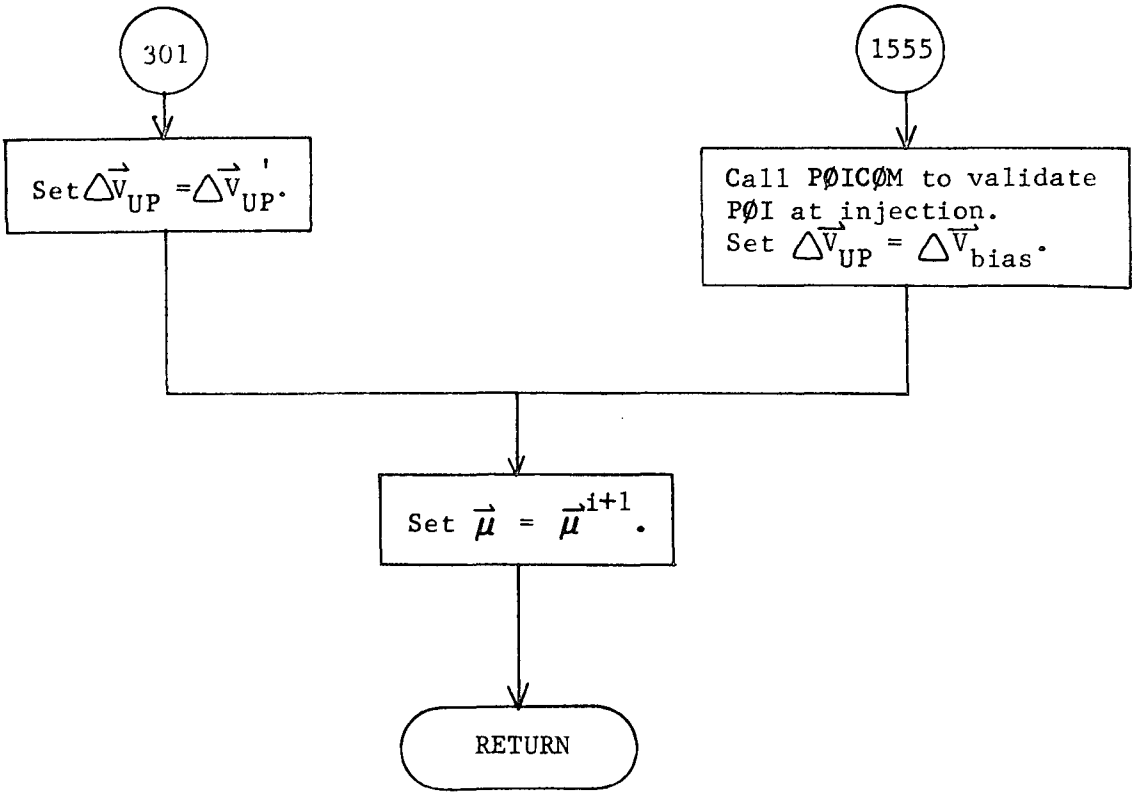



63
Set $\Delta\vec{V}_{TOT} = \Delta\vec{V}_{RB} + \Delta\vec{V}_N$
Call QCØMP to compute execution error covariance $\tilde{Q}_j^* = \tilde{Q}_j(\Delta\vec{V}_{TOT})$.
67
Write out $\Delta\vec{V}_{RB}$ and $\tilde{Q}_j^*$.
simulation
mode = ?
error analysis
Set $\Delta\vec{V}_{UP} = \Delta\vec{V}_{RB} + \Delta\vec{V}_N$
Set $\Delta\vec{V}_{UP} = \Delta\vec{V}_{RB}$.
Call PØICØM to compute and write out λ_j^* and PØI for the nominal aimpoint.
YES
PØI ≤ P_I ?
NO
Set $\vec{\mu} = \vec{\mu}^*$.
Set IBIAS = 0 .
RETURN
Will a velocity bias be actually imparted?
NO
YES
Set IBIAS = 1 .
A


A
YES
t_j = time of final midcourse correction?
NO
Set $\tilde{A}$ = I.
Set $\Delta t = t_{j+1} - t_j$ and call PSIM to compute the state transition matrix $\phi(t_{j+1}, t_j)$.
Call MATIN to compute $\phi^{-1}(t_{j+1}, t_j)$.
Compute variation matrix η_{j+1}.
Compute ψ_{j+1} matrix for the appropriate guidance policy.
Compute $\tilde{A} = \psi_{j+1}^T \psi_{j+1}$.
175
YES
IDENS ≠ 1 ?
NO
Write: IDENS ØPTIØN NØT AVAILABLE.
EXIT
Compute constants defining the elliptical constraint boundary associated with Λ_j.
B


B
Write out the equation of the elliptical constraint boundary.
ITRN > 0 ?
YES
NO
Compute and write out the initial guess $\vec{\mu}^{\,0}$.
Initialize Newton iteration counter IKNT.
Compute constants defining necessary condition for a minimum.
Use Newton iteration technique to determine the i-th iterate $\delta\vec{\mu}^{\,i}$.
Write out $\vec{\mu}^{\,i}$ and $\delta\vec{\mu}^{\,i}$.
Set $\vec{\mu}^{\,i+1} = \vec{\mu}^{\,i} + \delta\vec{\mu}^{\,i}$ and increment IKNT.
YES
$|\delta\mu_1^i| + |\delta\mu_2^i| < 1.$?
NO
IKNT > 25 ?
NO
YES
Write: Newton's method did not converge in BIAIM.
5000


5000
Write out final iteration $\vec{\mu}^{i+1}$.
Compute $\delta\vec{\mu} = \vec{\mu}^{i+1} - \vec{\mu}^{*}$ and
$\Delta\vec{V}_{bias} = \psi_j \delta\vec{\mu}$
and write out.
Store $\vec{\mu}^{i+1}$ in the XM array.
t_j = injection time?
YES
1555
NO
Set $\Delta\vec{V}'_{UP} = \Delta\vec{V}_{UP} + \Delta\vec{V}_{bias}$
and $\Delta\vec{V}'_{TØT} = \Delta\vec{V}_{TØT} + \Delta\vec{V}_{bias}$
Call QCØMP to compute
$\tilde{Q}_j = \tilde{Q}_j (\Delta\vec{V}'_{TØT})$
and write out.
Call PØICØM to compute Λ_j
and PØI for aimpoint $\vec{\mu}^{i+1}$.
Update iteration counter ITRN.
$|PØI - P_I| \leq P_I \times 10^{-3}$?
YES
NO
175
YES
ITRN $\leq$ 5 ?
NO
301


301
1555
Set $\vec{\Delta V}_{UP} = \vec{\Delta V}_{UP}'$.
Call PØICØM to validate PØI at injection.
Set $\vec{\Delta V}_{UP} = \vec{\Delta V}_{bias}$.
Set $\vec{\mu} = \vec{\mu}^{i+1}$.
RETURN

BIAS Analysis

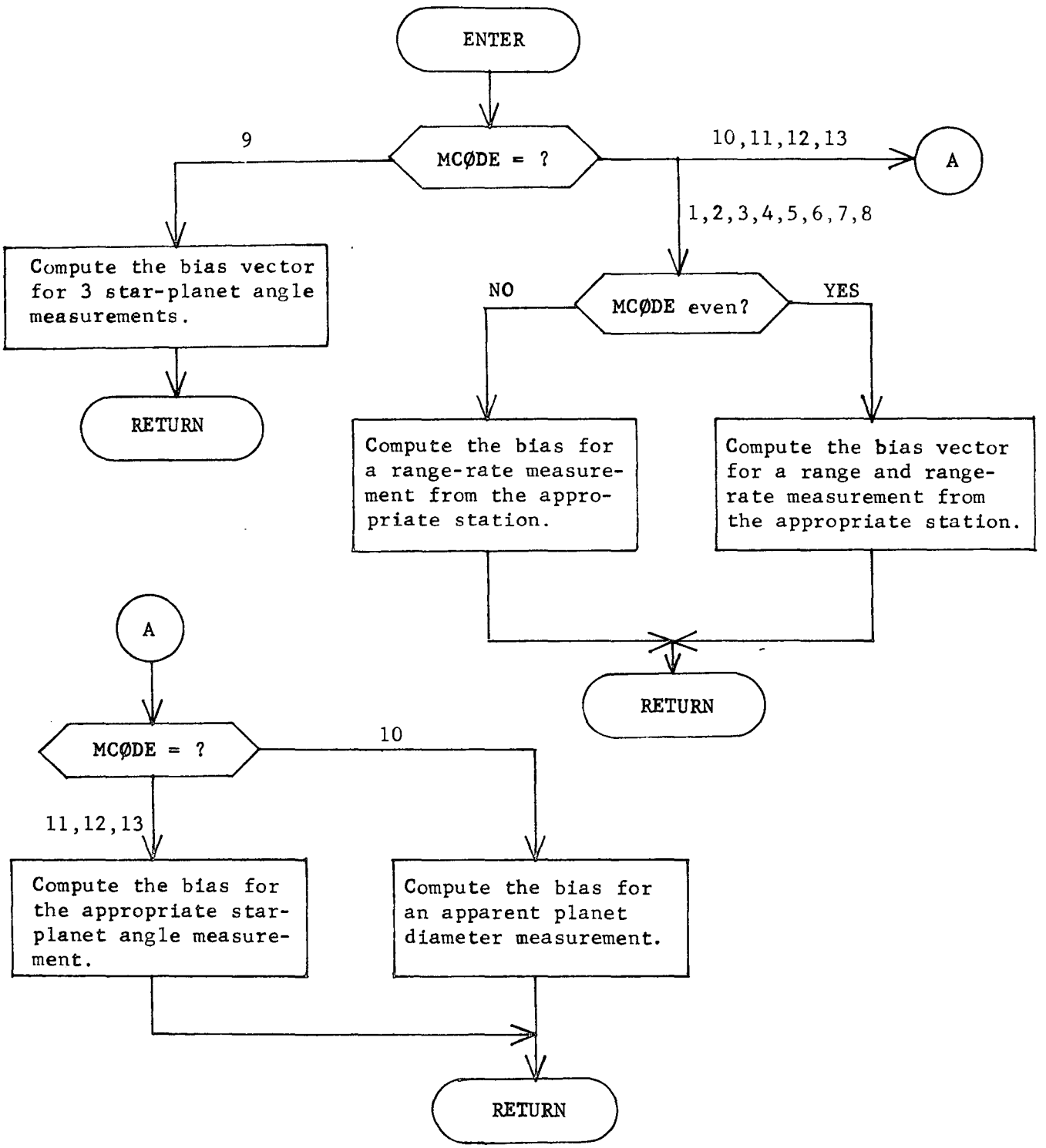
The actual measurement Y_k^a at time t_k is given by

$$Y_k^a = \underline{Y}_k + b_k + \nu_k$$

where $\underline{Y}_k$ is the ideal measurement, which would be made in the absence of instrumentation errors, b_k is the actual measurement bias and ν_k represents the actual measurement noise.

The function of subroutine BIAS is to compute the measurement bias b_k for the appropriate measurement type. The constant biases for all measurement devices are stored in the vector BIA. Subroutine BIAS selects the appropriate elements from this vector to construct the actual measurement bias.

BIAS Flow Chart



BLKDAT Analysis

Subroutine BLKDAT is responsible for setting up constants used in computing ephemeris data for the gravitating bodies.

The arrays set up by BLKDAT and their definitions are as follows:

Array	Definition
CN(80)	Constants defining mean elements for inner planets
ST(50)	Constants defining mean elements for outer planets
SMJR(18)	Constants defining semi-major axes for planets and moon
EMN(15)	Constants defining lunar elements
PMASS(11)	Gravitational constants of sun, planets, and moon
RMASS(11)	Mass of bodies relative to sun
RADIUS(11)	Surface radii of sun, planets, and moon
SPHERE(11)	Sphere of influence radii of sun, planets, and moon
MONTH(12)	Names of months for output purposes
PLANET(11)	Names of planets for output purposes

The definitions of the CN, ST, SMJR, and EMN arrays are provided in Tables 2 through 5 on the following page. The actual constants stored in those arrays are the ephemeris data listed on the next pages following.

The constants stored in the other arrays are given below.

Body	PMASS (AU^3/day^2)	RMASS*	RADIUS (AU)	SPHERE (AU)
Sun	2.959122083(-4)	1.0	4.66582(-3)	NA
Mercury	4.850(-11)	1.639(-7)	1.617(-5)	7.46(-4)
Venus	7.243(-10)	2.448(-6)	4.044(-5)	4.12(-3)
Earth	8.88757(-10)	3.003(-6)	4.263(-5)	6.18(-3)
Mars	9.5497905(-11)	3.236(-7)	2.279(-5)	3.78(-3)
Jupiter	2.8252(-7)	9.547(-4)	4.7727(-4)	.3216
Saturn	8.454(-8)	2.857(-4)	4.0374(-4)	.3246
Uranus	1.290(-8)	4.359(-5)	1.5761(-4)	.346
Neptune	1.5(-8)	5.069(-5)	1.4906(-4)	.5805
Pluto	7.4(-10)	2.501(-6)	4.679(-5)	.2366
Moon	1.0921748(-11)	3.696(-8)	1.161(-5)	3.71394(-4)

* Truncated from program values

Array Definitions

Constant	i	Ω	$\tilde{\omega}$	e	M	a	ω	E
Mercury	1	2	3	4	5	6	7	8
Venus	9	10	11	12	13	14	15	16
Earth	17	18	19	20	21	22	23	24
Mars	25	26	27	28	29	30	31	32
Jupiter	33	34	35	36	37	38	39	40
Saturn	41	42	43	44	45	46	47	48
Uranus	49	50	51	52	53	54	55	56
Neptune	57	58	59	60	61	62	63	64
Pluto	65	66	67	68	69	70	71	72
Moon	73	74	75	76	77	78	79	80

Table 1. ELMNT Array -- Conic Elements

Constant	a_0	a_1
Mercury	1	2
Venus	3	4
Earth	5	6
Mars	7	8
Jupiter	9	10
Saturn	11	12
Uranus	13	14
Neptune	15	16
Pluto	17	18

Table 2. SMJR Array

Constant	i_0	i_1	i_2	i_3	Ω_0	Ω_1	Ω_2	Ω_3	$\tilde{\omega}_0$	$\tilde{\omega}_1$	$\tilde{\omega}_2$	$\tilde{\omega}_3$	e_0	e_1	e_2	e_3	M_0	M_1	M_2	M_3
Mercury	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Venus	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Earth	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
Mars	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

Table 3. CN Array -- Inner Planet Constants

Constant	i_0	i_1	Ω_0	Ω_1	$\tilde{\omega}_0$	$\tilde{\omega}_1$	e_0	e_1	M_0	M_1
Jupiter	1	2	3	4	5	6	7	8	9	10
Saturn	11	12	13	14	15	16	17	18	19	20
Uranus	21	22	23	24	25	26	27	28	29	30
Neptune	31	32	33	34	35	36	37	38	39	40
Pluto	41	42	43	44	45	46	47	48	49	50

Table 4. ST Array -- Outer Planet Constants

Constant	Ω_0	Ω_1	Ω_2	Ω_3	$\tilde{\omega}_0$	$\tilde{\omega}_1$	$\tilde{\omega}_2$	$\tilde{\omega}_3$	L_0	L_1	L_2	L_3	i	e	a
Moon	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table 5. EMN Array -- Lunar Constants

Planetary and Lunar Ephemerides

Mean Elements of Mercury

$$i = 0.1222233228 + 3.24776685 \times 10^{-5}\ T - 3.199770295 \times 10^{-7}\ T^2$$

$$\Omega = 0.8228518595 + 2.068578774 \times 10^{-2}\ T + 3.034933644 \times 10^{-6}\ T^2$$

$$\tilde{\omega} = 1.3246996178 + 2.714840259 \times 10^{-2}\ T + 5.143873156 \times 10^{-6}\ T^2$$

$$e = 0.20561421 + 0.00002046\ T - 0.000000030\ T^2$$

$$M = 1.785111955 + 7.142471000 \times 10^{-2}\ d + 8.72664626 \times 10^{-9}\ D^2$$

$$a = 0.3870986 \text{ A.U.} = 57{,}909{,}370 \text{ km}$$

Mean Elements of Venus

$$i = 0.0592300268 + 1.7555510339 \times 10^{-5}\ T - 1.696847884 \times 10^{-8}\ T^2$$

$$\Omega = 1.3226043500 + 1.570534527 \times 10^{-2}\ T + 7.155849933 \times 10^{-6}\ T^2$$

$$\tilde{\omega} = 2.2717874591 + 2.457486613 \times 10^{-2}\ T + 1.704120089 \times 10^{-5}\ T^2$$

$$e = 0.00682069 - 0.00004774\ T + 0.000000091\ T^2$$

$$M = 3.710626172 + 2.796244623 \times 10^{-2}\ d + 1.682497399 \times 10^{-6}\ D^2$$

$$a = 0.7233316 \text{ A.U.} = 108{,}209{,}322 \text{ km}$$

Mean Elements of Earth

$$i = 0$$

$$\Omega = 0$$

$$\tilde{\omega} = 1.7666368138 + 3.000526417 \times 10^{-2}\ T + 7.902463002 \times 10^{-6}\ T^2 + 5.817764173 \times 10^{-8}\ T^3$$

$$e = 0.01675104 - 0.00004180\ T - 0.000000126\ T^2$$

$$M = 6.256583781 + 1.720196977 \times 10^{-2}\ d - 1.954768762 \times 10^{-7}\ D^2 - 1.22173047 \times 10^{-9}\ D^3$$

$$a = 1.0000003 \text{ A.U.} = 149{,}598{,}530 \text{ km}$$

Mean Elements of Mars

$$i = 0.0322944089 - 1.178097245 \times 10^{-5}\ T + 2.201054112 \times 10^{-7}\ T^2$$

$\Omega = 0.8514840375 + 1.345634309 \times 10^{-2}\ T - 2.424068406 \times 10^{-8}\ T^2$

$- 9.308422677 \times 10^{-8}\ T^3$

$\tilde{\omega} = 5.8332085089 + 3.212729365 \times 10^{-2}\ T + 2.266503959 \times 10^{-6}\ T^2$

$-2.084698829 \times 10^{-8}\ T^3$

$e = 0.09331290 + 0.000092064\ T - 0.000000077\ T^2$

$M = 5.576840523 + 9.145887726 \times 10^{-3}\ d + 2.365444735 \times 10^{-7}\ D^2$

$+ 4.363323130 \times 10^{-10}\ D^3$

$a = 1.5236915$ A.U. = 227,941,963 km

Mean Elements of Jupiter

$i = 0.0228410270 - 9.696273622 \times 10^{-5}\ T$

$\Omega = 1.7355180770 + 1.764479392 \times 10^{-2}\ T$

$\tilde{\omega} = 0.2218561704 + 2.812302353 \times 10^{-2}\ T$

$e = 0.0483376 + 0.00016302\ T$

$M = 3.93135411 + 1.450191928 \times 10^{-3}\ d$

$a = 5.202803$ A.U. = 778,331,525 km

Mean Element of Saturn

$i = 0.0435037861 - 7.757018898 \times 10^{-8}\ T$

$\Omega = 1.9684445802 + 1.523977870 \times 10^{-2}\ T$

$\tilde{\omega} = 1.5897996653 + 3.419861162 \times 10^{-2}\ T$

$e = 0.0558900 - 0.00034705\ T$

$M = 3.0426210430 + 5.837120844 \times 10^{-4}\ d$

$a = 9.538843$ A.U. = 1,426,996,160 km

Mean Elements of Uranus

$i = 0.0134865470 + 0.696273622 \times 10^{-6}\ T$

$\Omega = 1.2826407705 + 8.912087493 \times 10^{-3}\ T$

$\tilde{\omega} = 2.9502426085 + 2.834608631 \times 10^{-2}\ T$

$e = 0.0470463 + 0.00027204\ T$

$M = 1.2843599198 + 2.046548840 \times 10^{-4}\ d$

$a = (19.182281 - 0.00057008\ T)$ A.U. $= (2{,}869{,}640{,}310 - 85271\ T)$ km

Mean Elements of Neptune

$i = 0.0310537707 - 1.599885148 \times 10^{-4}\ T$

$\Omega = 2.2810642235 + 1.923032859 \times 10^{-2}\ T$

$\tilde{\omega} = 0.7638202701 + 1.532704516 \times 10^{-2}\ T$

$e = 0.00852849 + 0.00007701\ T$

$M = 0.7204851506 + 1.033089473 \times 10^{-4}\ d$

$a = (30.057053 + 0.001210166\ T)$ A.U. $= (4{,}496{,}490{,}000 + 181039\ T)$ km

Mean Elements of Pluto

$i = 0.2996706970859694$

$\Omega = 1.1914337550102258$

$\tilde{\omega} = 3.909919302791948$

$e = 0.2488033053623924$

$M = 3.993890007 + 0.6962635708298997 \times 10^{-4}$

$a = 39.37364135300176$ A.U. $= 5{,}890{,}213{,}786{,}146{,}730$ km

Mean Elements of Moon

$i = 5.1453964^{\circ}$

$\Omega = 259.183275^{\circ} - 0.0529539222d + 0.002078\ T^2 + 0.000002\ T^3$

$\tilde{\omega} = 334.329556^{\circ} + 0.1114040803d - 0.010325\ T^2 - 0.000012\ T^3$

$L = 270.434164^{\circ} + 13.1763965268d - 0.001133\ T^2 + 0.0000019\ T^3$

$a = .00256954448$ A.U.

$e = 0.054900489$

Note 1: The above elements are referred to the mean equinox and ecliptic of date except for Pluto.

Note 2: The elements for pluto are oscillating values for epoch 1960 September 23.0 E.T. = J.D. 2437200.5

Note 3: The time interval from the epoch is denoted by T when measured in Julian centuries of 36,525 ephemeris days, by D = 3.6525 T when measured in units of 10,000 ephemeris days, and by d = 10,000D = 36,525 T when measured in ephemeris days. Times are measured with respect to the epoch 1900 January 0.5 E.T. = J.D. 2415020.0.

Note 4: Angular relations are expressed in radians for planets and degrees for moon.

References: (1) Space Research Conic Program, Phase III, J.P.L., May 1969 (Planetary constants)
(2) The American Ephemeris and Nautical Almanac - 1965, U.S. Government Printing Office, Washington, p. 493 (Lunar constants)

CAREL Analysis

CAREL converts the cartesian state (position and velocity) of a massless point referenced to a gravitational body to the equivalent conic elements about that body.

Let the cartesian state be denoted $\vec{r}, \vec{v}$ and let the gravitational constant of the central body be μ .

The angular momentum constant c is

$$c = |\vec{r} \times \vec{v}| \tag{1}$$

The unit normal $\hat{W}$ to the orbital plane is

$$\hat{W} = \frac{\vec{r} \times \vec{v}}{c} \tag{2}$$

The semilatus rectum p is

$$p = \frac{c^2}{\mu} \tag{3}$$

The semi-major axis a is

$$a = \frac{r}{2 - \frac{rv^2}{\mu}} \tag{4}$$

Thus $a > 0$ for elliptical motion, $a < 0$ for hyperbolic motion. The eccentricity e is

$$e = \sqrt{1 - \frac{p}{a}} \tag{5}$$

Thus $e < 1$ for elliptical motion, $e > 1$ for hyperbolic motion. The inclination of the orbit i is computed from

$$\cos i = \hat{W}_z \tag{6}$$

The longitude of the ascending node Ω is defined by

$$\tan \Omega = \frac{\hat{W}_x}{-\hat{W}_y} \tag{7}$$

The true anomaly f at the given state is computed from

$$\cos f = \frac{p - r}{e\, r} \qquad \sin f = \frac{c\dot{r}}{\mu e} \tag{8}$$

Now define an auxiliary vector $\hat{z}$ by

$$\hat{z} = \frac{r}{c}\,\vec{v} - \frac{\dot{r}}{c}\,\vec{r} \tag{9}$$

Then $\hat{P}$, the unit vector to periapsis, and $\hat{Q}$, the in-plane normal to $\hat{P}$, are defined by

$$\hat{P} = \hat{r}\cos f - \hat{z}\sin f \tag{10}$$

$$\hat{Q} = \hat{r}\sin f + \hat{z}\cos f \tag{11}$$

where $\hat{r} = \frac{\vec{r}}{r}$. The argument of periapsis ω is then computed from

$$\tan \omega = \frac{\hat{P}_z}{\hat{Q}_z} \tag{12}$$

The conic time from periapsis t_p is computed from different formulae depending upon the sign of the semi-major axis. For $a > 0$ (elliptical motion)

$$t_p = \sqrt{\frac{a^3}{\mu}}\,(E - e \sin E)$$

$$\cos E = \frac{e + \cos f}{1 + e \cos f} \qquad \sin E = \frac{\sqrt{1 - e^2}\,\sin f}{1 + e \cos f} \tag{13}$$

For $a < 0$ (hyperbolic motion) the time from periapsis is

$$t_p = \sqrt{\frac{a^3}{\mu}}\,(e \sinh H - H)$$

$$\tanh \frac{H}{2} = \sqrt{\frac{e - 1}{e + 1}}\,\tan \frac{f}{2} \tag{14}$$

Reference: Battin, R. H., Astronautical Guidance, McGraw-Hill Book Co., New York, 1964.

CASCAD Analysis

CASCAD approximates the state transition matrix $\Phi_{f,o}$ defining state perturbations over an arbitrary interval $\left[t_o, t_f\right]$ by recursively computing state transition matrices over intervals $\left[t_o, t_1\right]$, $\left[t_o, t_2\right]$, ..., $\left[t_o, t_f\right]$.

The recursive formula for the k+1 iteration based on the k-th iteration is given by

$$\Phi_{k+1,o} = \Psi_{k+1,k}\ \Phi_{k,o} \qquad (1)$$

where $\Psi_{k+1,k}$ is the state transition matrix for the k+1-st interval $\left[t_k, t_{k+1}\right]$.

The time interval $\Delta t_{k+1} = t_{k+1} - t_k$ is determined by the position vector $\vec{r}_k$ of the spacecraft relative to the target planet along the nominal n-body trajectory at the time t_k. Then if R_{SOI} denotes the radius of the sphere of influence of the target planet the time interval is defined by

$\Delta t_{k+1} = \Delta t_{planet}$ if $r_k \leq R_{SOI}$

$= \Delta t_{sun}$ if $r_k > R_{SOI}$ and the n-body nominal trajectory propagated over Δt_{sun} does not intersect the SOI.

$= \Delta t_{SOI}$ if $r_k > R_{SOI}$ and the n-body nominal trajectory intersects the SOI after the time interval Δt_{SOI} where $\Delta t_{SOI} \quad \Delta t_{sun}$.

where Δt_{planet} and Δt_{sun} are input parameters. For the last interval a partial step may be required so that $\Delta t_n = t_f - t_{n-1}$.

The $\Psi_{k+1,k}$ matrix may be computed by either of two models. In the patch conic model the position and velocity vectors $\vec{R}_k$, $\vec{V}_k$ of the spacecraft relative to the dominant body (the sun if $\Delta t_{k+1} = \Delta t_{sun}$ or Δt_{SOI}, the target planet if $\Delta t_{k+1} = \Delta t_{planet}$) at the time t_k is used to define a

conic with respect to the dominant body and the Danby matrizant over the given interval defines $\Psi_{k+1,k}$ (CØNC2) .

In the virtual mass model the position and velocity vectors $\vec{R}_k$, $\vec{V}_k$ are computed relative to the virtual mass and the gravitational constant used is that of the virtual mass magnitude at the time t_k. The Danby matrizant corresponding to this conic then is used to compute $\Psi_{k+1,k}$ (CØNC2).

The recursive process continues until the state transition matrix over the entire interval $\left[t_o, t_f\right]$ is determined.

Reference: Danby, J.M.A., "The Matrizant of Keplerian Motion," AIAA Journal, vol 2, no 1, January, 1964.

CENTER Analysis

Let the state vector of position and velocity of the gravitating bodies (excluding the moon) in heliocentric ecliptic coordinates be denoted ρ_i, υ_i at some reference time. Let the index of the earth be i_E. Then the coordinates of all bodies (excluding the moon) relative to the earth is

$$\begin{aligned} \vec{r}_i &= \vec{\rho}_i - \vec{\rho}_{i_E} \qquad i=1,n\ ,\ i\neq i_M \\ \vec{v}_i &= \vec{\omega}_i - \vec{\omega}_{i_E} \qquad i=1,n\ ,\ i\neq i_M \end{aligned} \tag{1}$$

Let the position and velocity of the moon relative to the earth be denoted $\vec{r}_{i_M}$, $\vec{v}_{i_M}$.

Define the radius vector to the center of mass (in earth ecliptic coordinates) by

$$R_{CM} = \frac{1}{M} \sum_{i=1}^{n} \mu_i \vec{r}_i \qquad M = \sum_{i=1}^{n} \mu_i \tag{2}$$

Its velocity relative to the earth may then be found by differentiation.

$$\vec{V}_{CM} = \frac{1}{M} \sum_{i=1}^{n} \mu_i \vec{v}_i \tag{3}$$

The coordinates of all gravitating bodies relative to the center of mass may then be computed

$$\begin{aligned} \vec{R}_i &= \vec{r}_i - \vec{R}_{CM} \\ \vec{V}_i &= \vec{v}_i - \vec{V}_{CM} \end{aligned} \tag{4}$$

CONC2 Analysis

CONC2 is responsible for the computation of a state transition matrix about a conic trajectory using the Danby matrizant analytic formula.

Danby has shown (see Reference 2) that the state transition matrix (or matrizant) has a particularly simple form if written in the orbital plane coordinate system. The state transition matrix Φ defined by

$$\delta x_f = \Phi (t_f, t_o) \delta X_o \tag{1}$$

where δx_f, δx_o refer to perturbations about a conic trajectory at time t_f, t_o respectively may be written in the orbital plane system

$$\bar{\Phi}(t_f, t_o) = M(t_f) M^{-1} (t_o) \tag{2}$$

where $M(t)$, $M^{-1}(t)$ may be computed from the following formulae

$$M = \begin{bmatrix} \dot{X} & Y\dot{X}-h & 0 & 2X-3\tau\dot{X} & Y\dot{Y} & 0 \\ \dot{Y} & -X\dot{X} & 0 & 2Y-3\tau\dot{Y} & -Y\dot{X}-2h & 0 \\ 0 & 0 & Y & 0 & 0 & -X \\ \ddot{X} & Y\dot{X}+Y\ddot{X} & 0 & -\dot{X}-3\tau\ddot{X} & \dot{Y}^2 + Y\ddot{Y} & 0 \\ \ddot{Y} & -\dot{X}^2-X\ddot{X} & 0 & -\dot{Y}-3\tau\ddot{Y} & -\dot{X}\dot{Y}-Y\ddot{X} & 0 \\ 0 & 0 & \dot{Y} & 0 & 0 & -\dot{X} \end{bmatrix} \tag{3}$$

$$M^{-1} = AJM^TJ^T \tag{4}$$

where $X, Y, \dot{X}, \dot{Y}, \ddot{X}, \ddot{Y}$ are evaluated at the time t
h is the angular momentum constant
τ is the time interval from t to some epoch (periapsis)

and $$A = \text{diag}\ (a/\mu,\ a/\mu h,\ 1/h,\ a/\mu,\ a/\mu h,\ 1/h) \tag{5}$$

$$J = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix} \tag{6}$$

Thus to use the Danby formulation one must determine the transformation from the reference frame to the orbital plane coordinates, compute the values of the quantities $X, Y, \dot{X}, \dot{Y}, \ddot{X}, \ddot{Y}$ and h and τ at the times t_o, t_f and then use the above equations.

Let the initial state of the conic be denoted $\vec{r}$, $\vec{v}$, the gravitational force μ, and the time interval $\triangle t$. Then the unit vectors $\hat{P}$ in the direction of periapsis, $\hat{W}$ in the direction of the angular momentum vector, and $\hat{Q} = \hat{W} \times \hat{P}$ defining the orbital plane coordinate system may be computed by the following conic equations

$$h = |\vec{r} \times \vec{v}| \tag{7}$$

$$\hat{W} = \frac{\vec{r} \times \vec{v}}{h} \tag{8}$$

$$\dot{r} = \frac{\vec{r} \cdot \vec{v}}{v} \tag{9}$$

$$p = \frac{h^2}{\mu} \tag{10}$$

$$a = \frac{r}{2 - rv^2/\mu} \tag{11}$$

$$e = \sqrt{1 - p/a} \tag{12}$$

$$\cos f = \frac{p - r}{er} \qquad \sin f = \frac{\dot{r} h}{\mu e} \tag{13}$$

$$\vec{z} = \frac{r}{h} \vec{v} - \frac{\dot{r}}{h} \vec{r} \tag{14}$$

$$\hat{P} = \cos f \frac{\vec{r}}{r} - \sin f \, \vec{z} \tag{15}$$

$$\hat{Q} = \sin f \frac{\vec{r}}{r} + \cos f \, \vec{z} \tag{16}$$

$$\dot{f} = \frac{c}{r^2} \tag{17}$$

The transformation matrix from the original $\vec{r}$, $\vec{v}$ system to the orbital plane system may then be written

$$T = \left[\hat{P} \mid \hat{Q} \mid \hat{W} \right] \tag{18}$$

Let the true anomaly at the pertinent time (t_o or t_f) be denoted f. Then the quantities required in (3) are written

$$\begin{aligned} X &= r \cos f & Y &= r \sin f \\ \dot{X} &= \dot{r} \cos f - r\dot{f} \sin f & \dot{Y} &= \dot{r} \sin f + r\dot{f} \cos f \\ \ddot{X} &= -\frac{\mu X}{r^3} & \ddot{Y} &= -\frac{\mu Y}{r^3} \end{aligned} \tag{19}$$

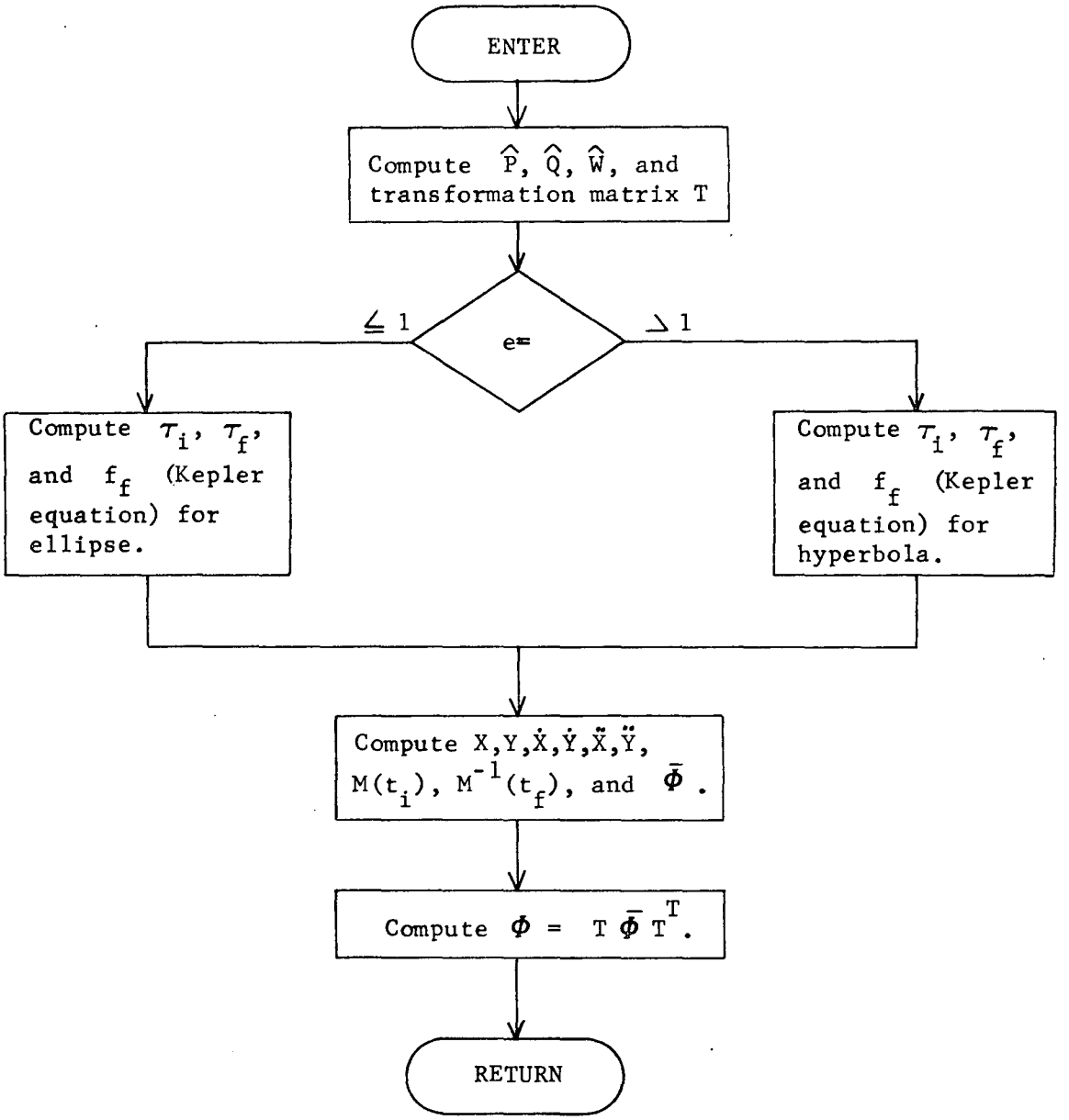
Having computed the state transition matrix $\bar{\Phi}$ corresponding to the orbital plane system by equations (2), (3), (4), it is an easy task to convert it to the normal reference system

$$\Phi = T \bar{\Phi} T^T \tag{20}$$

References: Battin, R. H., Astronautical Guidance, McGraw-Hill Book Co., New York, 1964.

Danby, J.M.A., Matrizant of Keplerian Motion, AIAA J., vol. 3, no. 4, April, 1965.

CONC2 Flow Chart



CONVRT Analysis

Geocentric equatorial position and velocity components are related to geocentric radius, declination, right ascension, velocity magnitude, flight path angle, and azimuth through the following equations:

$$x = r \cos \phi \cos \theta$$

$$y = r \cos \phi \sin \theta$$

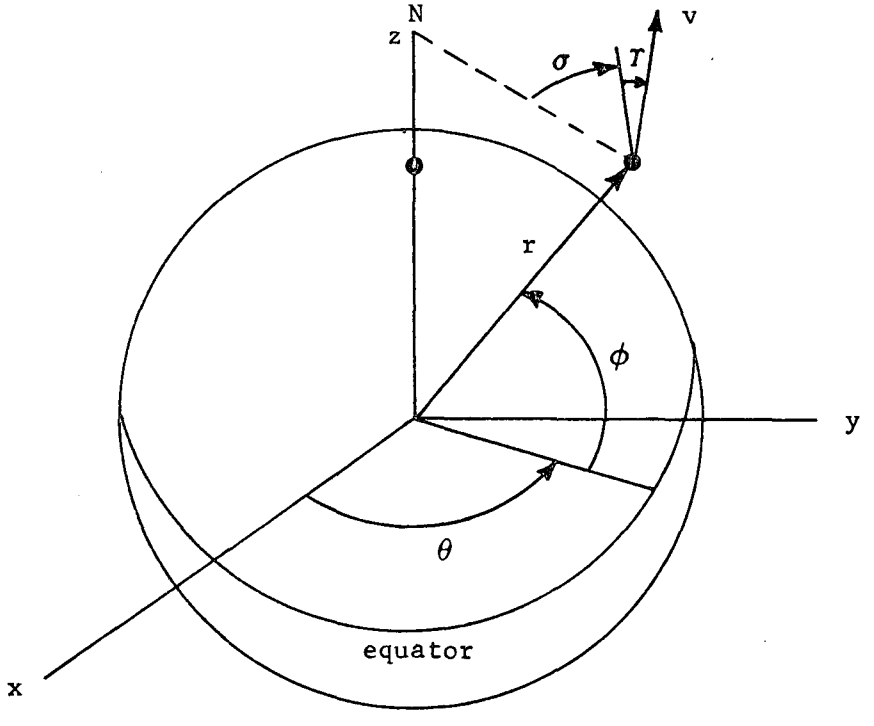
$$z = r \sin \phi$$

$$\dot{x} = v (\sin \gamma \cos \phi \cos \theta - \cos \gamma \sin \sigma \sin \theta - \cos \gamma \cos \sigma \sin \phi \cos \theta)$$

$$\dot{y} = v (\sin \gamma \cos \phi \sin \theta + \cos \gamma \sin \sigma \cos \theta - \cos \gamma \cos \sigma \sin \phi \sin \theta)$$

$$\dot{z} = v (\sin \gamma \sin \phi + \cos \gamma \cos \sigma \cos \phi)$$

The definitions of pertinent quantities are apparent in the following figure.



COPINS Analysis:

COPINS determines the impulsive correction and time required to insert from an approach hyperbola into a coplanar elliptical orbit. The approach hyperbola is specified by a planetocentric state $\vec{r}, \vec{v}$ at a decision time t_d. The desired elliptical orbit is prescribed by input parameters a, e, $\Delta\omega$ where a and e are the semi-major axis and eccentricity of the desired ellipse and $\Delta\omega$ is the angle (measured counter clockwise) from the hyperbolic periapsis to the periapsis of the desired orbit. The situation is illustrated in Figure 1.

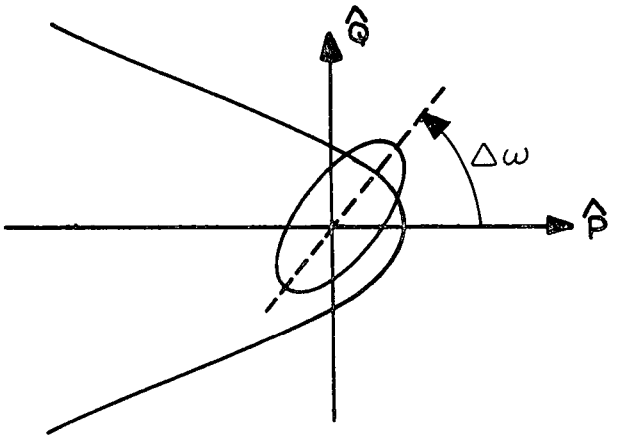


Figure 1. Approach Hyperbola and Desired Orbit

The planetocentric ecliptic state $\vec{r}, \vec{v}$ at the time of decision t_d is first converted to Keplerian elements $(a_H, e_H, i_H, \ _H, \Omega_H, t_{Hd})$ via subroutine CAREL where t_{Hd} is the time from periapsis (negative on the approach ray). The angle f_∞ between the hyperbolic periapsis and the approach asymptote $\hat{S}$ is computed from

$$\cos f_\infty = \frac{1}{e} \qquad 0 < f_\infty < 90^\circ \tag{1}$$

Thus the angle ω between the hyperbolic periapsis and the desired elliptical periapsis is given by

$$\omega = \Delta\omega \tag{2}$$

The hyperbola and ellipse may therefore be described in the PQ plane by standard conic formula, specifically,

$$r_H = \frac{p_H}{1 + e_H \cos\theta}$$

$$r_E = \frac{p_E}{1 + e_E \cos(\theta - \omega)} \quad . \tag{3}$$

where θ is measured counter-clockwise from $\hat{P}$ and p_H, p_E are the semi-latus rectum of the hyperbola and ellipse respectively. Obviously if an angle of intersection θ^* is known, the states on both conics $(\vec{r}^*, \vec{v}_H^*)$ and $(\vec{r}^*, \vec{v}_E^*)$ may be computed from conic formulae and the desired impulsive correction is given by

$$\vec{\Delta v} = \vec{v}_E^* - \vec{v}_H^* \tag{4}$$

Likewise the time from periapsis to the intersection point t^* may be computed using hyperbolic formula and therefore the time from decision to execution is given by

$$\Delta t = t^* - t_d \tag{5}$$

Thus the coplanar insertion problem reduces to the determination of the optimal angle θ^* for the impulsive maneuver.

From (3) the values of θ for which $r_H = r_E$ are given by

$$\cos\theta = \frac{-xy \pm z\sqrt{D}}{y^2 + z^2} \tag{6}$$

where

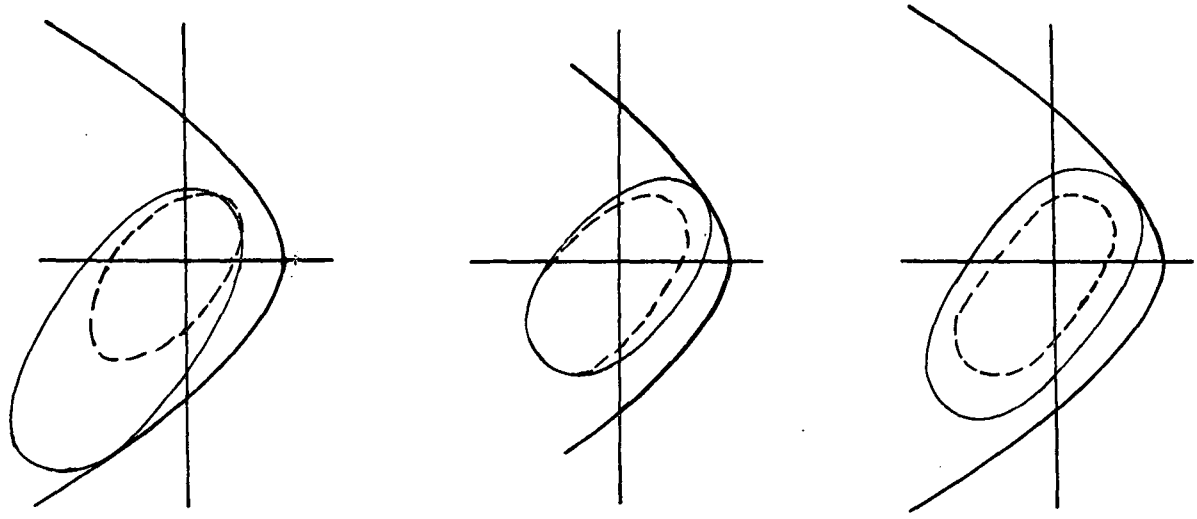
$$\begin{aligned} x &= p_H - p_E \\ y &= p_H e_E \cos\omega - p_E e_H \\ z &= p_H e_E \sin\omega \\ D &= y^2 + z^2 - x^2 \end{aligned} \tag{7}$$

If the discriminant $D \geq 0$ there are at most two real non-extraneous solutions θ_1, θ_2 such that $r_E(\theta) = r_H(\theta)$. Note that the angle θ may not lie in the region inside the approach and departure asymptotes. If there are two solutions, both Δv 's are computed by (4) and the minimum Δv transfer is selected.

If $D<0$, the applied hyperbola and the desired orbit do not intersect and there is no impulsive transfer between the two conics. In such a case the desired elements a_E and e_E are modified to determine the "best" tangential solution possible. Three different modifications are tested:

(1) Vary r_a while holding r_p at the desired value.
(2) Vary r_p while holding r_a at the desired value.
(3) Vary a_E while holding e_E at the desired value.

The three modification schemes are illustrated in Figure 2 where the original nonintersecting orbit is shown by the broken lines.



a. Modify r_a b. Modify r_p c. Modify a

Figure 2. Candidate Orbit Modifications

It is desired to modify the "a" and the "e" of the desired orbit to achieve the tangential configurations. From (6) it is obvious that a necessary condition for a tangential solution is given by D=0. Using (7) D may be written

$$D = p_H^2 (e_E^2 - 1) + p_E^2 b + 2p_H p_E - cp_E e_E$$

where $b = e_H^2 - 1$

$$c = 2p_H e_H \cos\omega \qquad (8)$$

where it is observed the approach hyperbola is fixed and it is desired not to vary the ω of the desired ellipse so that subsequent apsidal rotations are avoided.

Modification Option 1: Rewriting (8a) in terms of a and r_p leads to

$$a^2D = (4\, r_p^2\, b + 4\, r_p\, p_H - 2\, r_p\, c)\, a^2 + (-2\, p_H^2\, r_p - 4\, r_p^3\, b - 2\, p_H\, r_p^2 + 3\, r_p^2\, b)a + (p_H^2\, r_p^2 + r_p^4\, b - c\, r_p^3) \qquad (9)$$

Now if D is set equal to 0, r_p held at its desired value, and the resulting quadratic solved for "a", the solution will correspond to the tangential solution which holds r_p constant. If $a \leq 0$ or imaginary, the solution is disregarded. The modified eccentricity is of course defined by

$$e = 1 - \frac{r_p}{a} \qquad (10)$$

Modification Option 2: Rewriting (8a) in terms of a and r_a leads to

$$a^2D = (4\, r_a^2\, b + 4\, r_a\, p_H + 2\, r_a\, c)a^2 + (-2\, p_H^2\, r - 4\, r_a^3\, b - 2\, p_H\, r_a^2 - 3\, r_a^2\, c)a + (p_H^2\, r_a^2 + r_a^4\, b + c\, r_a^3) \qquad (11)$$

For computational purposes the similarity between (9) and (11) may be exploited. Again setting $D = 0$ and holding r_a at its desired value, the value of "a" may be determined which specifies the tangential solution holding r_a constant. Having determined a realistic value of "a", the corresponding eccentricity is given by

$$e = \frac{r_a}{a} - 1 \qquad (12)$$

Modification Option 3: Rewriting (8a) in terms of a and e_E leads to

$$D = (d^2b)a^2 + (2p_Hd - cd\, e_E)a - dp_H^2$$
$$d = (1 - e_E^2) \qquad (13)$$

Setting $D = 0$ and solving for "a" while holding e_E at its desired value then defines the option 3 solution.

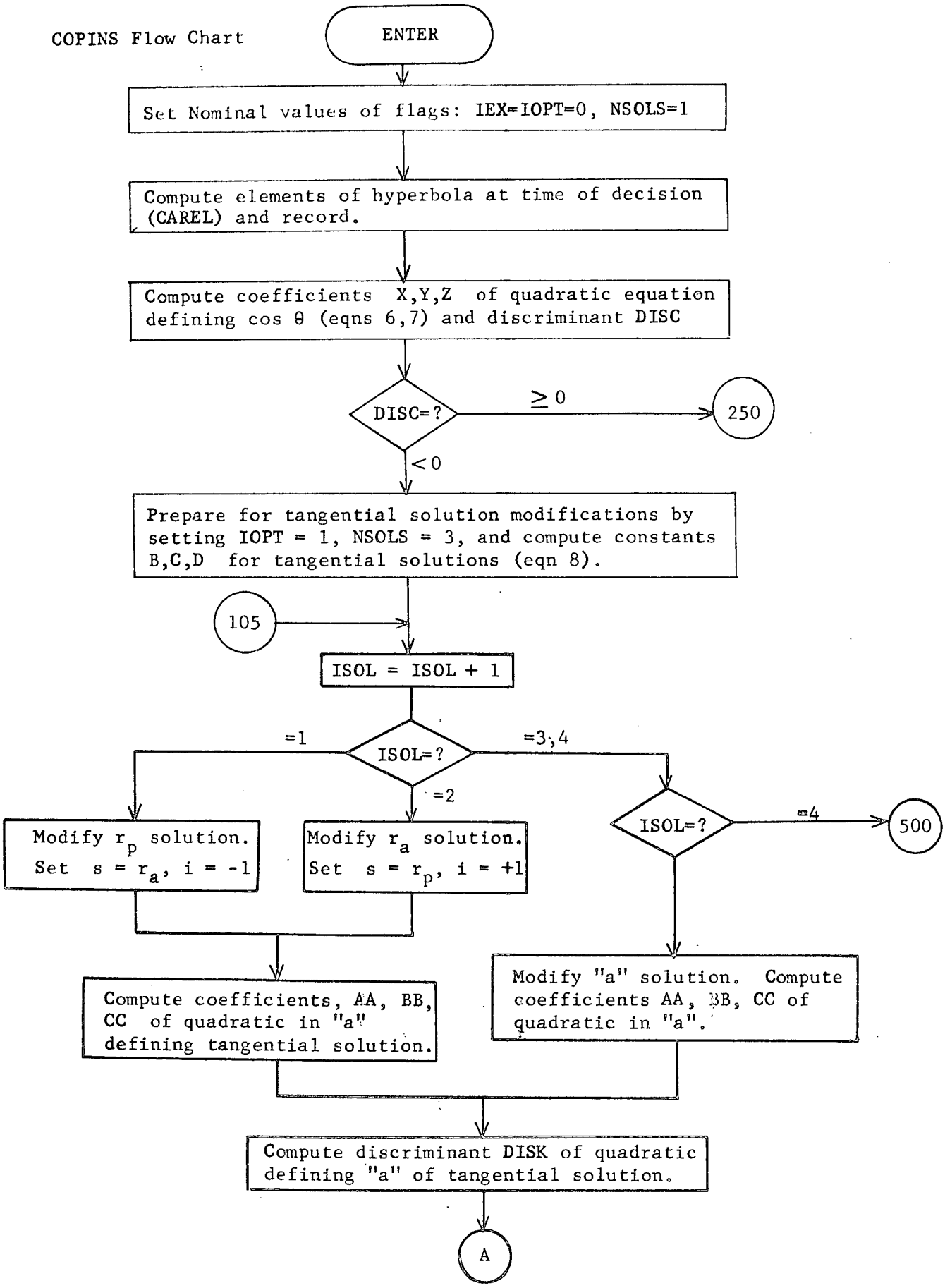
To determine the "best" modified orbit from the three candidate options a rather arbitrary scheme is used. A scalar error is assigned to each option according to a weighting factor and the difference between the desired and achieved values of the periapsis and apoapsis radii:

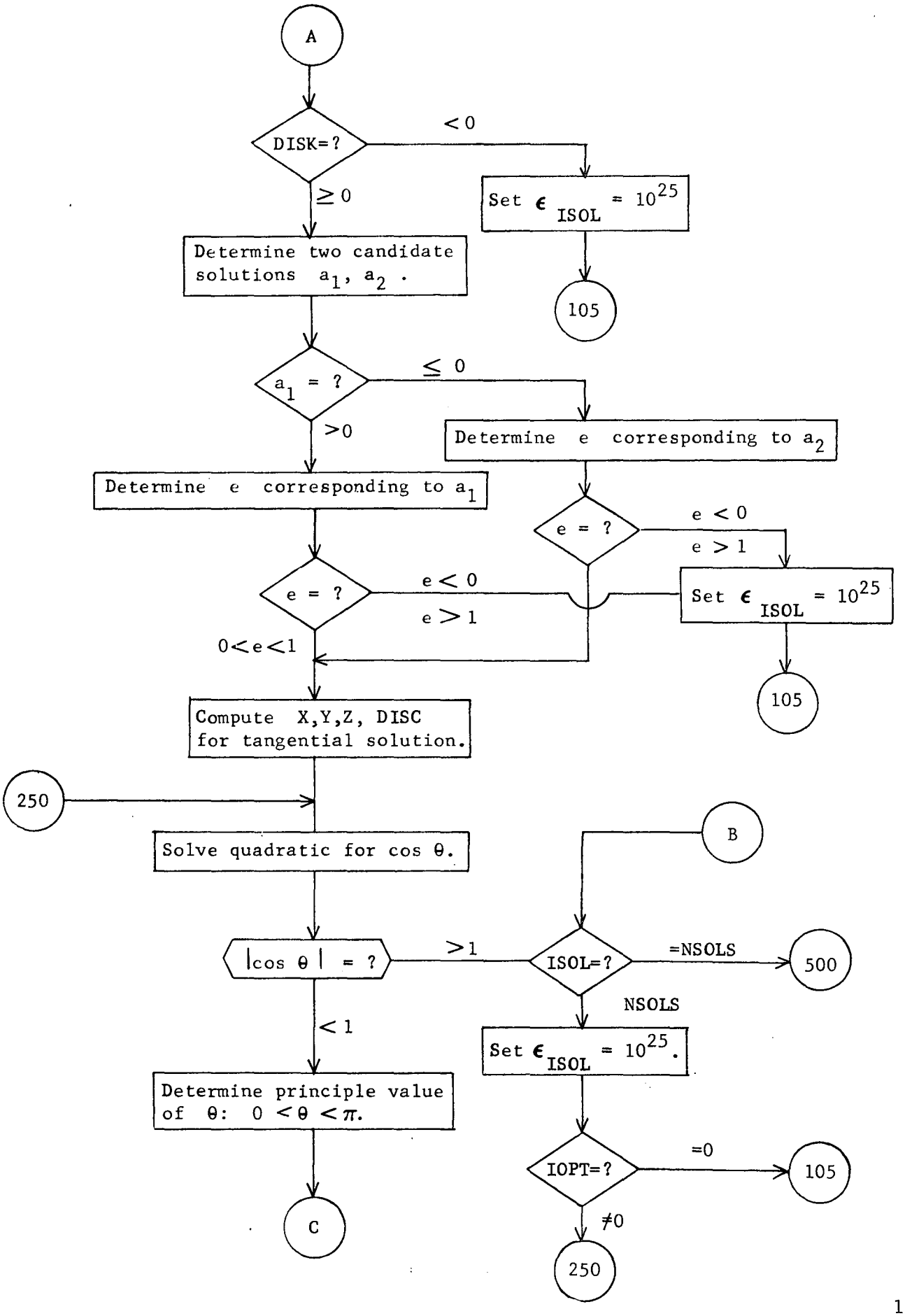
$$E_i = W_i \left(|\Delta r_a| + |\Delta r_p| \right)$$

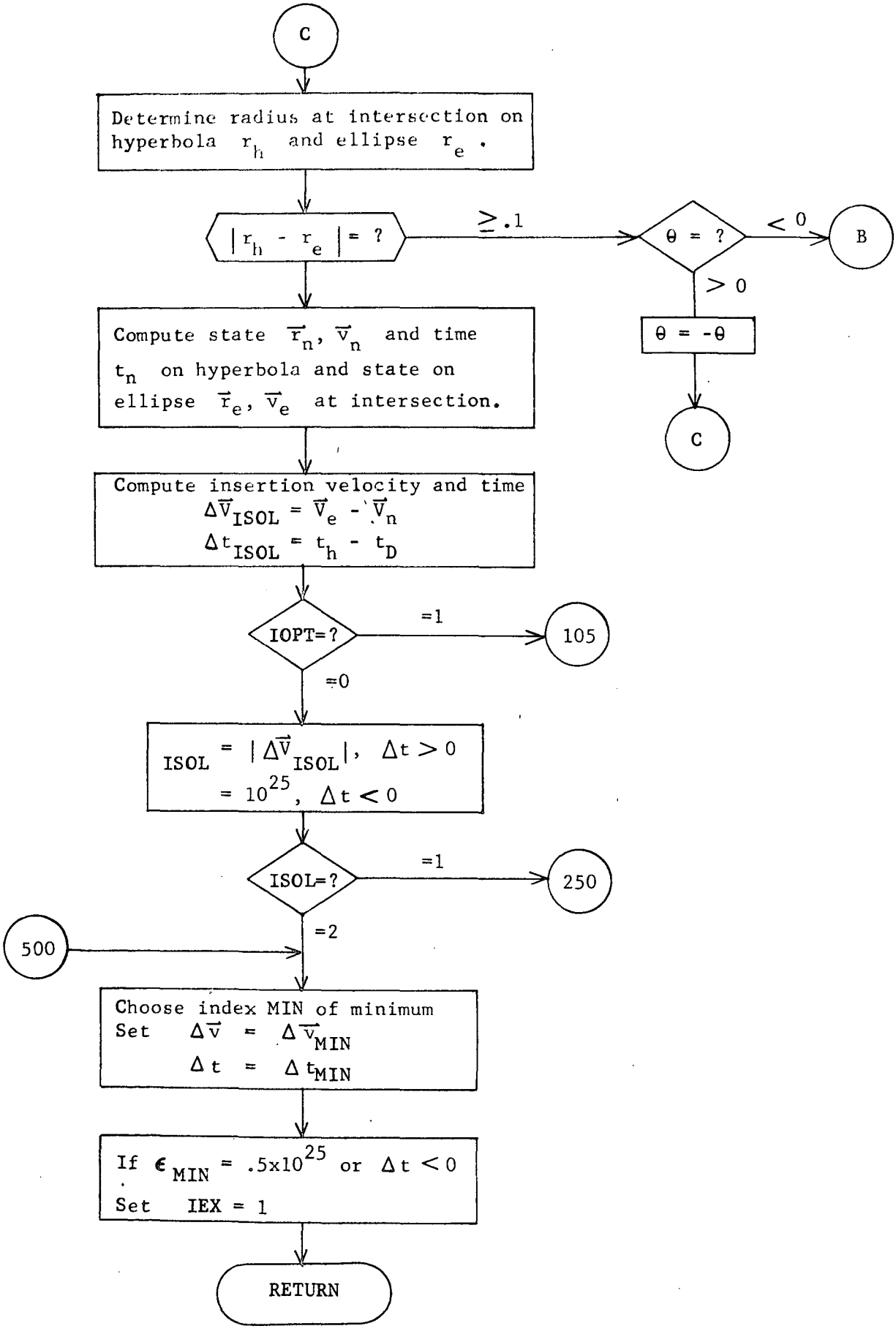
where the scalar factor W_i is set to 1,2,3 respectively for the three options. Thus the preferred strategy is the one which requires a correction only at apoapsis while the least desired scheme requires subsequent corrections both at periapsis and apoapsis.

Having determined orbital elements that necessarily lead to a tangential solution, (6) may now be used to compute the angle of intersection θ .

COPINS Flow Chart




A
DISK=?
< 0
≥ 0
Set $\epsilon_{ISOL} = 10^{25}$
105
Determine two candidate solutions a_1, a_2 .
a_1 = ?
≤ 0
>0
Determine e corresponding to a_2
Determine e corresponding to a_1
e = ?
e < 0
e > 1
e = ?
e < 0
e > 1
Set $\epsilon_{ISOL} = 10^{25}$
105
0<e<1
Compute X,Y,Z, DISC for tangential solution.
250
Solve quadratic for cos θ.
B
|cos θ| = ?
>1
ISOL=?
=NSOLS
500
NSOLS
< 1
Set $\epsilon_{ISOL} = 10^{25}$.
Determine principle value of θ: 0 <θ < π.
IOPT=?
=0
105
C
≠0
250


C
Determine radius at intersection on hyperbola r_h and ellipse r_e.
$|r_h - r_e| = ?$
$\geq .1$
$\theta = ?$
< 0
B
> 0
$\theta = -\theta$
C
Compute state $\vec{r}_n$, $\vec{v}_n$ and time t_n on hyperbola and state on ellipse $\vec{r}_e$, $\vec{v}_e$ at intersection.
Compute insertion velocity and time
$\Delta\vec{V}_{ISOL} = \vec{V}_e - \vec{V}_n$
$\Delta t_{ISOL} = t_h - t_D$
IOPT=?
=1
105
=0
ISOL $= |\Delta\vec{V}_{ISOL}|$, $\Delta t > 0$
$= 10^{25}$, $\Delta t < 0$
ISOL=?
=1
250
500
=2
Choose index MIN of minimum
Set $\Delta\vec{v} = \Delta\vec{v}_{MIN}$
$\Delta t = \Delta t_{MIN}$
If $\epsilon_{MIN} = .5x10^{25}$ or $\Delta t < 0$
Set IEX = 1
RETURN

DESENT Analysis

DESENT computes a correction to an initial velocity by the steepest descent or conjugate gradient techniques for use by TARGET.

The technique used is determined by the value of METHOD. DESENT takes n steps in the conjugate gradient directions before rectifying by making a steepest descent step where n = METHOD - 1 . Thus if METHOD = 1, all steps are taken in the steepest descent direction.

Let the current iterate initial state be denoted $\vec{r}$, $\vec{v}$. Let the scalar error of the auxiliary parameters corresponding to this state be denoted ϵ . Let the perturbation size for the sensitivities be dv .

The current gradient $\vec{g}_c$ is computed by numerical differencing. For the k-th component of $\vec{g}_c$ the corresponding component of velocity is perturbed by dv

$$\vec{v}_p = \vec{v} + dv \left[\delta_{1K}, \delta_{2K}, \delta_{3K}\right]^T \tag{1}$$

The initial state ($\vec{r}$, $\vec{v}_p$) is then propagated to the final stopping conditions. Let the auxiliary parameters of that trajectory be denoted $\vec{\alpha}_p$. The error associated with the perturbed state is then

$$\epsilon_p = \vec{W} \cdot (\vec{\alpha}_p - \vec{\alpha}^*) \tag{2}$$

where $\vec{W}$ represents the weighting factors and $\vec{\alpha}^*$ are the desired target conditions. The k-th component of the current gradient is then

$$g_{c_K} = \frac{\epsilon_p - \epsilon}{dv} \tag{3}$$

The corrected gradient is given by

$$\begin{aligned} \vec{p}_c &= \vec{g}_c && \text{steepest descent step} \\ &= \frac{|g_c|^2}{|g_p|^2} \vec{p}_p + \vec{g}_c && \text{conjugate gradient step} \end{aligned} \tag{4}$$

where the subscript c refers to a current parameter, p refers to a previous-step parameter.

The unit vector in the direction of the next step is then given by

$$\vec{q}_c = -\frac{\vec{p}_c}{p_c} \tag{5}$$

The directional derivative of the scalar error in the the direction $\vec{q}_c$ is

$$d = \vec{g}_c \cdot \vec{q}_c \tag{6}$$

The nominal step size $\bar{h}$ is computed from a linear approximation to null the error

$$\bar{h} = \frac{\epsilon}{-d} \tag{7}$$

The initial state corrected by this nominal correction is then propagated to the final stopping conditions and the resulting error $\bar{\epsilon}$ computed. The three conditions

$$\begin{aligned} y(o) &= \epsilon \\ y(\bar{h}) &= \bar{\epsilon} \\ y'(o) &= d \end{aligned} \tag{8}$$

may now be applied to the formula of a parabola $y - \epsilon^* = a(x - h^*)^2$ to predict the optimal step size h^* yielding the minimum error ϵ^*

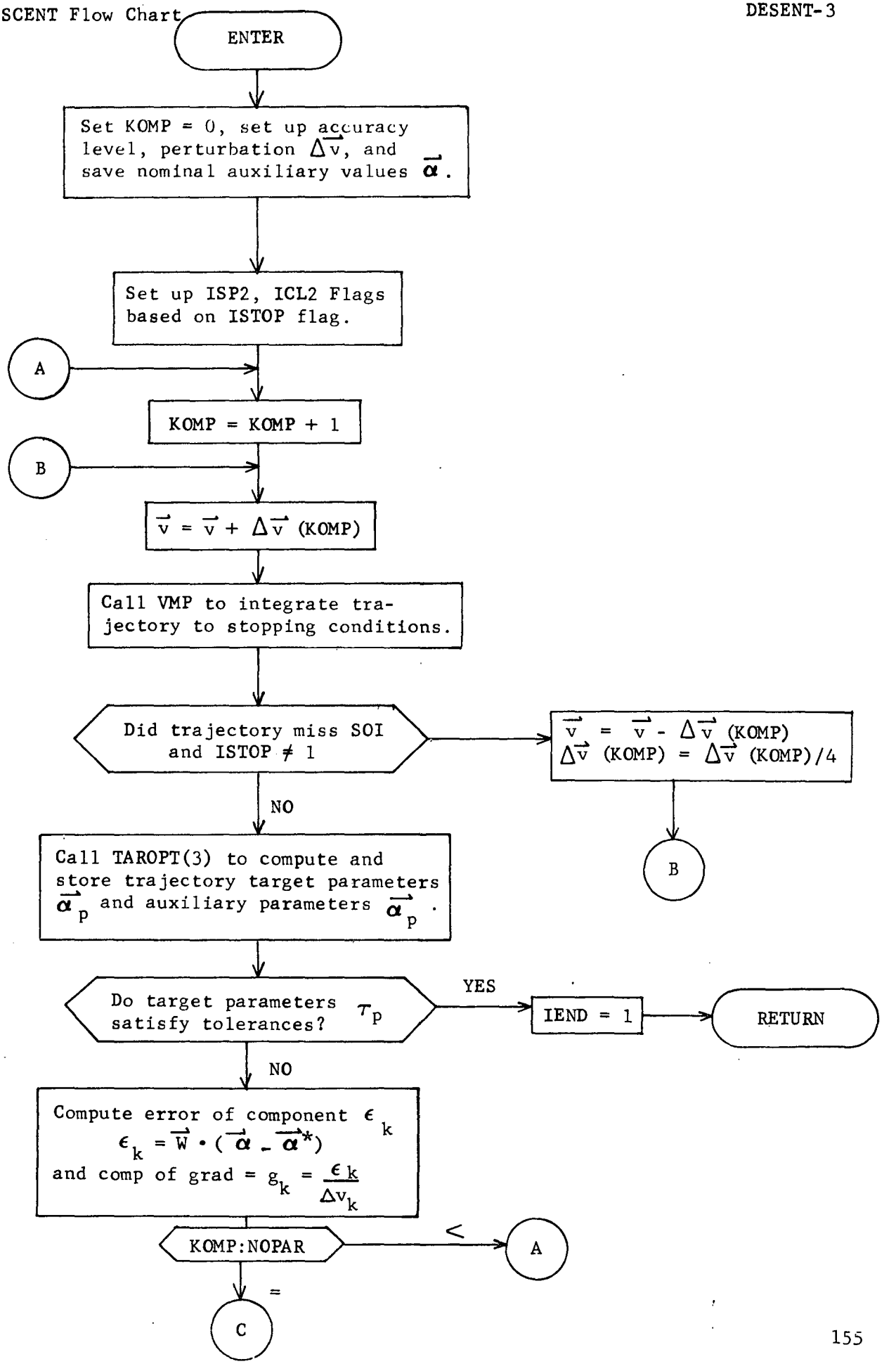
$$h^* = \frac{d\bar{h}^2}{2(d\bar{h} + \epsilon - \bar{\epsilon})} \tag{9}$$

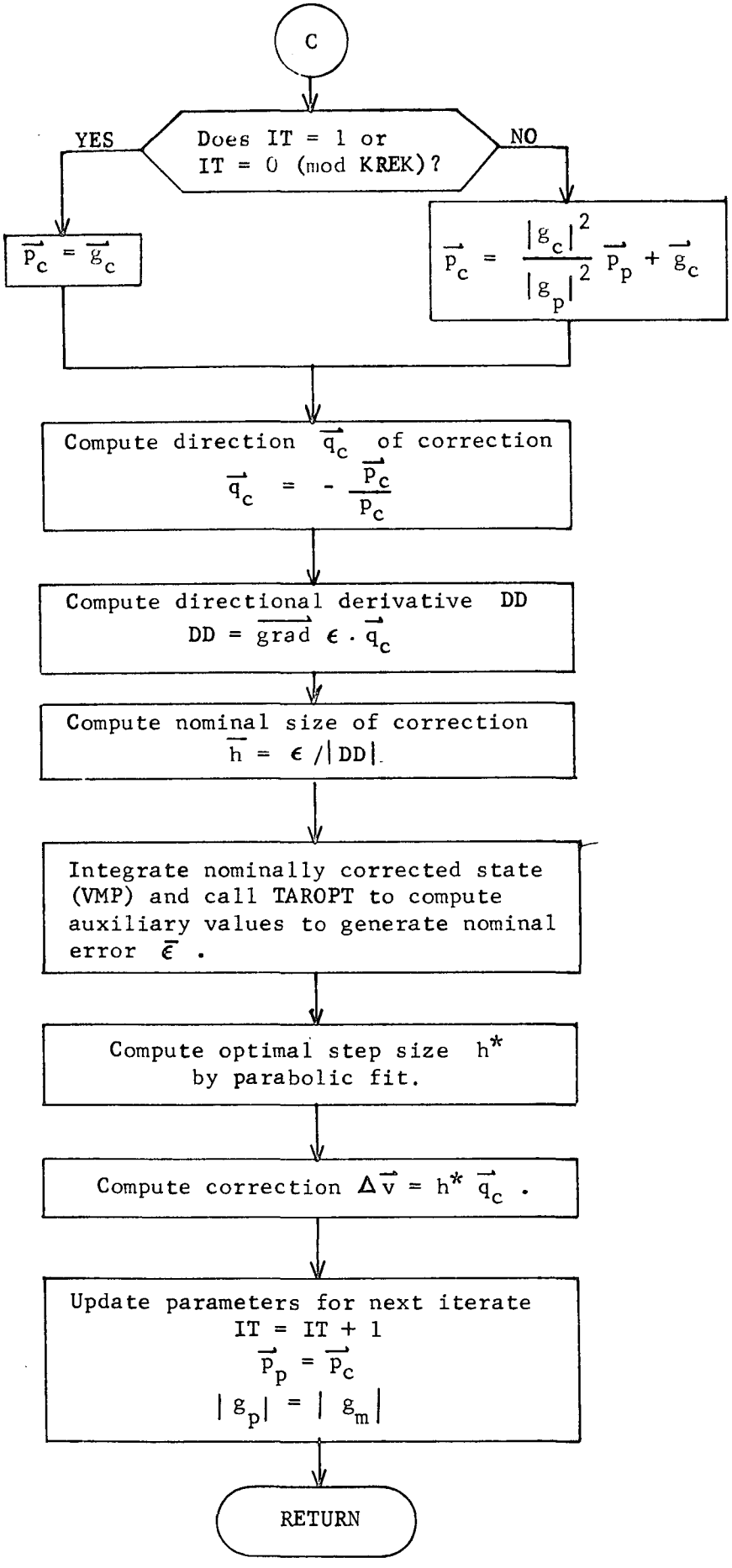
The correction for the current is then given by

$$\Delta v = h^* \vec{q}_g \tag{10}$$

Reference: Myers, G. E., "Properties of the Conjugate Gradient and Davidon Methods", AAS Paper 68-081. Presented at 1968 AAS/AIAA Astrodynamics Specialist Conference, Jackson, Wyoming.

DESCENT Flow Chart




C
Does IT = 1 or IT = 0 (mod KREK)?
YES
NO
$\vec{p}_c = \vec{g}_c$
$\vec{p}_c = \frac{|g_c|^2}{|g_p|^2}\vec{p}_p + \vec{g}_c$
Compute direction $\vec{q}_c$ of correction
$\vec{q}_c = -\frac{\vec{p}_c}{p_c}$
Compute directional derivative DD
$DD = \overrightarrow{grad}\ \epsilon \cdot \vec{q}_c$
Compute nominal size of correction
$\bar{h} = \epsilon / |DD|$
Integrate nominally corrected state (VMP) and call TAROPT to compute auxiliary values to generate nominal error $\bar{\epsilon}$.
Compute optimal step size h^* by parabolic fit.
Compute correction $\Delta\vec{v} = h^* \vec{q}_c$.
Update parameters for next iterate
IT = IT + 1
$\vec{p}_p = \vec{p}_c$
$|g_p| = |g_m|$
RETURN

DIMPCP Analysis

Subroutine DIMPCP converts the actual probe target parameters of declination δ and right ascension α of the trajectory impact point on the planet into the auxiliary target parameters of equivalent B•T and B•R. To do so it assumes the direction of the hyperbolic excess velocity and the energy of the trajectory are known so the $\underline{S}$, $\underline{T}$, and $\underline{R}$* in the ecliptic frame and the semimajor axis, a, of the approach hyperbola are available as inputs. To complete the specification of the probe impact point, the subroutine also requires the radius, r, of the planet at impact as well as the transformation, L, from the inertial ecliptic frame to the coordinate system to which the right ascension and delcination are referenced.

Derivation of the necessary equations is relatively straightforward once the appropriate variables are defined. Let $\underline{\rho}$ be a planet-centered unit vector in the direction of the impact point. Then, in the inertial ecliptic system,

$$\underline{\rho} = L^T \begin{pmatrix} \cos\alpha \cos\delta \\ \sin\alpha \cos\delta \\ \sin\delta \end{pmatrix} \quad . \tag{1}$$

Define ϕ to be the unique angle on the closed interval from 0 to π between $\underline{\rho}$ and $\underline{S}$ (see Fig. 1). Finally denote the true anomalies of $\underline{\rho}$ and $\underline{S}$ by θ and θ_S, respectively.

First DIMPCP determines whether the desired impact point is indeed targetable. It is apparent from Figure 1 that

$$|\theta| = \phi - \theta_S \tag{2}$$

It is further obvious from the figure that the approach hyperbola will intersect the planet surface at true anomalies of both $+\theta$ and $-\theta$. Obviously only the negative true anomaly impact points are physically realizable since the trajectory stops at the first intersection with the planet. Hence DIMPCP requires the

$$\theta_S \leq \phi \quad . \tag{3}$$

---

*For definitions of these vectors see the analysis section of the subroutine STIMP.

2-5

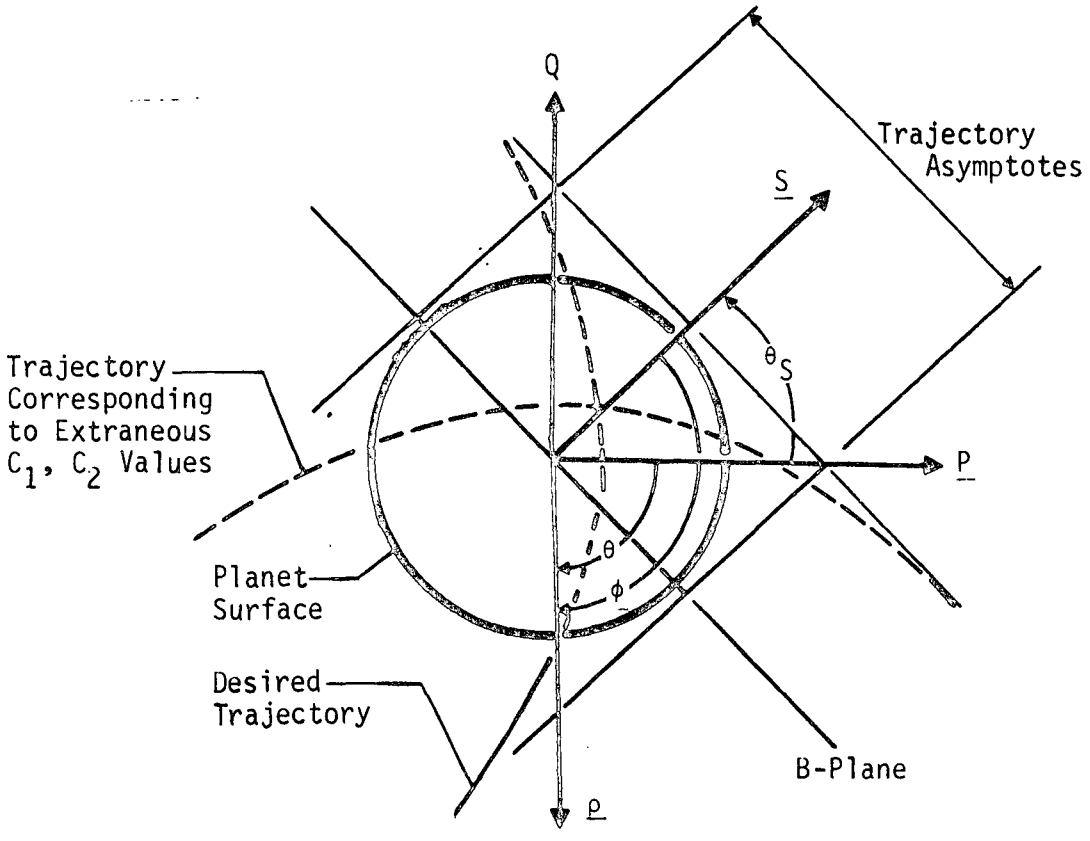


Figure 1 Geometry of Probe Impact

In other words, there is a circular region on the planet surface of radius θ_S about the outgoing pierce point of the $\underline{S}$ vector inside of which no probes can be targeted. If ρ falls in this untargetable region, DIMPCP repositions the desired impact point direction to $\underline{\rho}'$, the nearest acceptable direction in the plane determined by $\underline{S}$ and $\underline{\rho}$. Analytically this is done by expressing $\underline{\rho}'$ as a linear combination of $\underline{\rho}$ and $\underline{S}$; that is

$$\underline{\rho}' = d_1 \underline{\rho} + d_2 \underline{S} \quad . \tag{4}$$

Then the constraints

$$||\underline{\rho}'|| = 1 \tag{5}$$

and

$$\underline{\rho} \cdot \underline{\rho}' = \cos(\theta_S - \phi) \tag{6}$$

are applied. These result in the following pair of simultaneous equations for d_1 and d_2:

$$1 = d_1^2 + 2\, d_1\, d_2 \cos\,\phi + d_2^2 \qquad (7)$$

$$\cos\,(\theta_S - \phi) = d_1 + d_2 \cos\,\phi \quad . \qquad (8)$$

Solving (8) for d_1 in terms to d_2 and substituting into (7) produces the quadratic

$$1 - \cos^2\,(\theta_S - \phi) = d_2^2\,(1 - \cos^2\,\phi) \quad . \qquad (9)$$

Assuming $|\cos\,\phi| \neq 1$ leads then to the conclusion that

$$d_2 = \pm\sqrt{\frac{1 - \cos^2\,(\theta_S - \phi)}{1 - \cos^2\,\phi}} \quad . \qquad (10)$$

Figure 2 geometrically interprets the two roots of equation (9) given by (10).

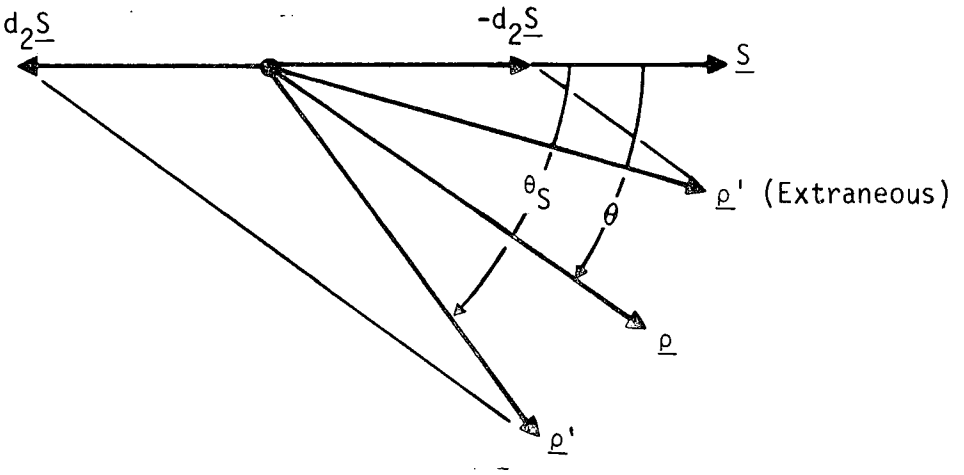


Figure 2 Geometrical Interpretation of the Two d_2-Roots

Clearly the d_2 value corresponding to the positive radical is extraneous since it produces a $\underline{\rho}'$ nearer $\underline{S}$ than $\underline{\rho}$. Hence

$$d_2 = -\sqrt{\frac{1 - \cos^2 (\theta_S - \phi)}{1 - \cos^2 \phi}} \tag{11}$$

$$d_1 = \cos (\theta_S - \phi) + \cos \phi \sqrt{\frac{1 - \cos^2 (\theta_S - \phi)}{1 - \cos^2 \phi}} \quad . \tag{12}$$

The exceptional case that $\cos \phi = -1$ cannot occur since then $\phi = \pi$ and hence $\theta_S < \phi$. However, $\cos \phi$ can equal 1. In this case $\underline{\rho}$ and $\underline{S}$ are coincident so $\underline{\rho}'$ cannot be taken as a linear combination of the two. Further, no particular point on the boundary of the circular untargetable region recommends itself. Hence DIMPCP arbitrarily puts $\underline{\rho}'$ in the S-T plane as

$$\underline{\rho}' = \underline{S} \cos \theta_S + \underline{T} \sin \theta_S \quad . \tag{13}$$

On repositioning $\underline{\rho}$, DIMPCP prints out the right ascention α' and declination δ' of $\underline{\rho}'$ making use of the formulae

$$\alpha' = \tan^{-1} (\rho_2'/\rho_1') \tag{14}$$

$$\delta' = \sin^{-1} (\rho_3') \quad . \tag{15}$$

Having repositioned $\underline{\rho}$, if necessary, DIMPCP calculates the magnitude of the desired $\underline{B}$. It can readily be shown that

$$\cos \theta_S = \frac{1}{e} \tag{16}$$

$$\sin \theta_S = \frac{\sqrt{e^2 - 1}}{e} \tag{17}$$

$$B = |a| \sqrt{e^2 - 1} \quad . \tag{18}$$

Recall the polar equation of the near-planet conic trajectory, namely

$$r = \frac{a(1 - e^2)}{1 + e \cos \theta} \tag{19}$$

Substituting equation (2) into (19) and rearranging gives

$$a(1 - e^2) = r \left[1 + e \left(\cos \phi \cos \theta_S + \sin \phi \sin \theta_S\right)\right] . \tag{20}$$

Using equations (16) and (17) in (2) yields

$$a(1 - e^2) = r \left(1 + \cos \phi + \sqrt{e^2 - 1} \sin \phi\right) . \tag{21}$$

Eliminating the eccentricity from equation (19) by means of (18) produces a quadratic in B, that is,

$$-B^2/a = r (1 + \cos \phi) - rB/a . \tag{22}$$

Applying the quadratic formula to (20) gives

$$B = \frac{\left[r \sin \phi \pm \sqrt{r^2 \sin^2 \phi - 4 ar (1 + \cos \phi)}\right]}{2} \tag{23}$$

Since $1 + \cos \phi \geq 0$ for all ϕ and $a < 0$ for hyperbolic approach trajectories, the root corresponding to the negative radical in (21) produces a negative magnitude of $\underline{B}$ and hence must be extraneous. Thus

$$B = \frac{\left[r \sin \phi + \sqrt{r^2 \sin^2 \phi - 4 ar (1 + \cos \phi)}\right]}{2} \tag{24}$$

One can further conclude from the radicand of (23) that a solution for B will exist if, and only if,

$$4 ar (1 + \cos \theta) \leq r^2 (1 - \cos^2 \phi) , \tag{25}$$

or equivalently if $\cos \phi \neq -1$

$$\cos \phi \leq 1 - \frac{4a}{r} . \tag{26}$$

Since a is negative for a hyperbolic approach, this last inequality is always true. Further if $\cos \phi = -1$, $B = 0$. Hence equation (22) always has the unique nonnegative solution given by (24).

Next DIMPCP computes the direction of $\underline{B}$. Since the desired $\underline{B}$ must lie in the plane determined by $\underline{S}$ and $\underline{\rho}$, there must exist real numbers C_1 and C_2 so that

$$\underline{B}/B = C_1 \underline{S} + C_2 \underline{\rho} \quad . \tag{27}$$

Applying the constraints that $||\underline{B}/B|| = 1$ and $\underline{B} \cdot \underline{S} = 0$ respectively

$$C_1^2 + 2 C_1 C_2 \cos \phi + C_2^2 = 1 \tag{28}$$

$$C_1 + C_2 \cos \phi = 0 \quad . \tag{29}$$

Solving these two equations simultaneously for a and b gives

$$C_2 = \pm 1/\sin \phi \tag{30}$$

$$C_1 = \mp \cot \phi \quad . \tag{31}$$

The negative C_2 root and the corresponding positive C_1 root are extraneous since they place $\underline{B}$ on the side of $\underline{S}$ opposite to $\underline{\rho}$ as shown in Figure 1. Substituting the correct pair of roots from (30) and (31) into (27) gives the direction of the desired $\underline{B}$ as

$$\underline{B}/B = (\underline{\rho} - \underline{S} \cos \phi)/\sin \phi \quad . \tag{32}$$

Clearly in the exceptional case that $\sin \phi = 0$, the trajectory passes through the center of the planet coinciding with its asymptote so that $\underline{B} = \underline{0}$.

Finally DIMPCP calculates the desired B·T and B·R coordinates now that $\underline{B}$ is known:

$$B \cdot T = B_1 T_1 + B_2 T_2 \tag{33}$$

$$B \cdot R = B_1 R_1 + B_2 R_2 + B_3 R_3 \quad . \tag{34}$$

DYNØ Analysis

Subroutine DYNØ evaluates the assumed dynamic covariance matrix Q over the time interval $t = t_{k+1} - t_k$ if ICØDE = 0. If ICØDE = 1 the actual dynamic noise covariance matrix Q' is evaluated over the same interval. In either case the dynamic noise covariance matrix is assumed to have the form

$$Q = \text{diag}\left(\tfrac{1}{4} K_1 \Delta t^4, \tfrac{1}{4} K_2 \Delta t^4, \tfrac{1}{4} K_3 \Delta t^4, K_1 \Delta t^2, K_2 \Delta t^2, K_3 \Delta t^2\right)$$

where dynamic noise constants K_1, K_2, and K_3 have units of km^2/s^4. To compute the actual dynamic noise covariance matrix Q', we simply replace K_1, K_2, and K_3 with the actual dynamic noise constants K_1', K_2', and K_3', respectively.

DYNØS Analysis

Subroutine DYNØS performs two functions. It's first function is identical to that of subroutine DYNØ, namely, to evaluate the dynamic noise covariance matrix Q over the time interval $\Delta t = t_{k+1} - t_k$.

The second function of subroutine DYNØS is to compute the actual dynamic noise $\vec{\omega}_{k+1}$, which represents the integrated effect of unmodelled accelerations acting on the spacecraft over the time interval Δt. Actual dynamic noise $\vec{\omega}_{k+1}$ is used elsewhere in the program to compute the actual state deviations of the spacecraft from the most recent nominal trajectory.

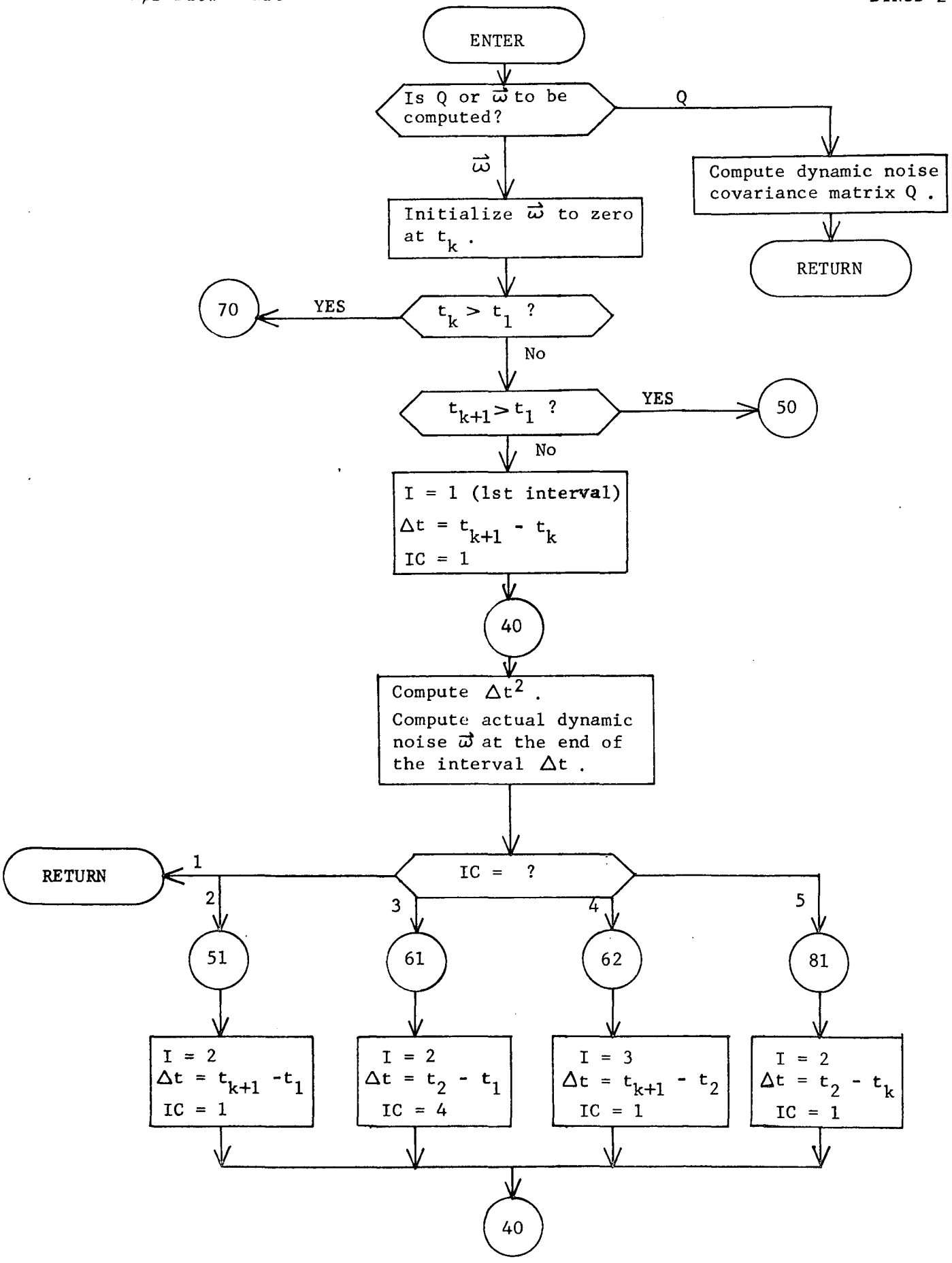
If we define $\vec{\omega}_{k+1} = \left[\vec{\omega}_{r_{k+1}}, \quad \vec{\omega}_{v_{k+1}} \right]^T$, where

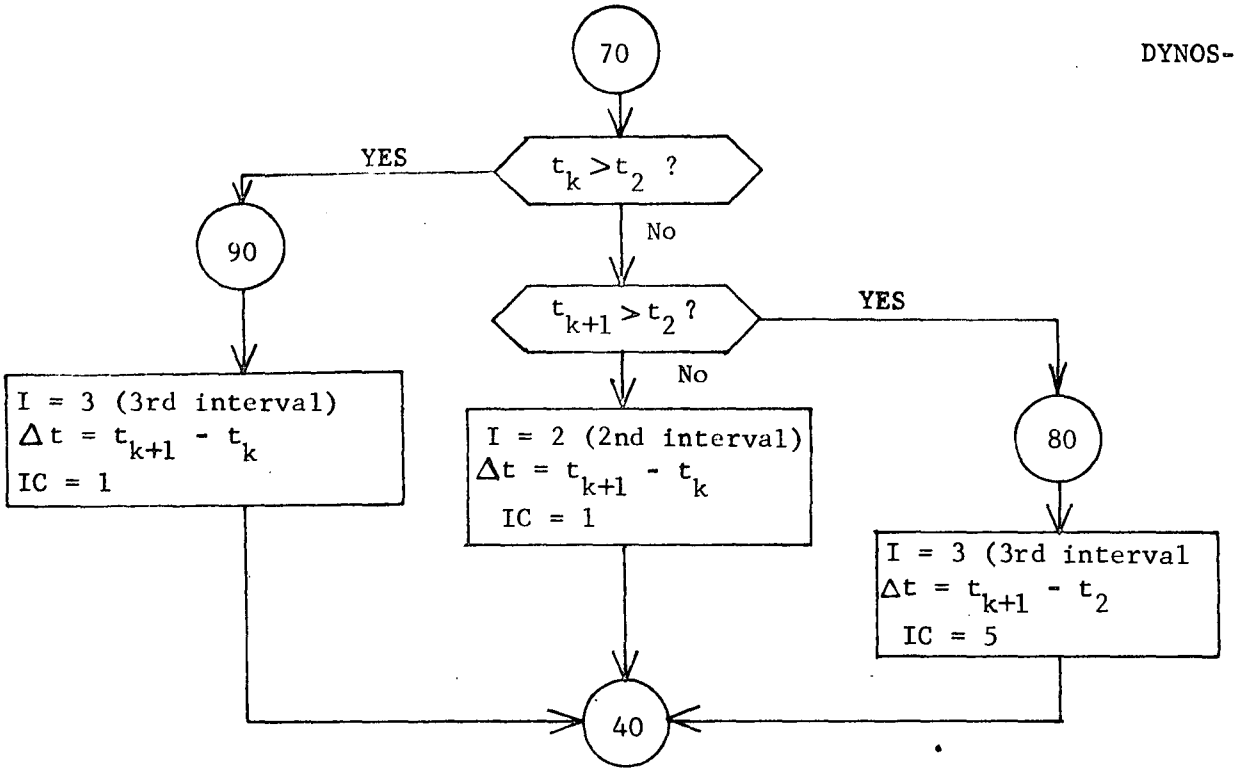
$\vec{\omega}_{r_{k+1}}$ and $\vec{\omega}_{v_{k+1}}$ denote the contributions of unmodelled accelerations to spacecraft position and velocity, respectively, and if we assume constant unmodelled acceleration $\vec{a}$, then

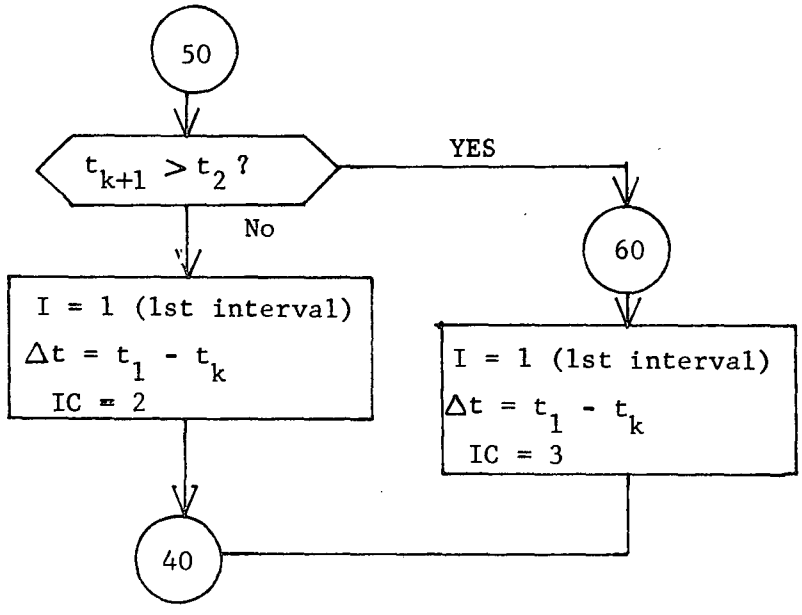
$$\vec{\omega}_{r_{k+1}} = \frac{\vec{a}}{2} (t_{k+1} - t_k)^2 + \vec{\omega}_{v_k} (t_{k+1} - t_k) + \vec{\omega}_{r_k}$$

$$\vec{\omega}_{v_{k+1}} = \vec{a} (t_{k+1} - t_k) + \vec{\omega}_{v_k}$$

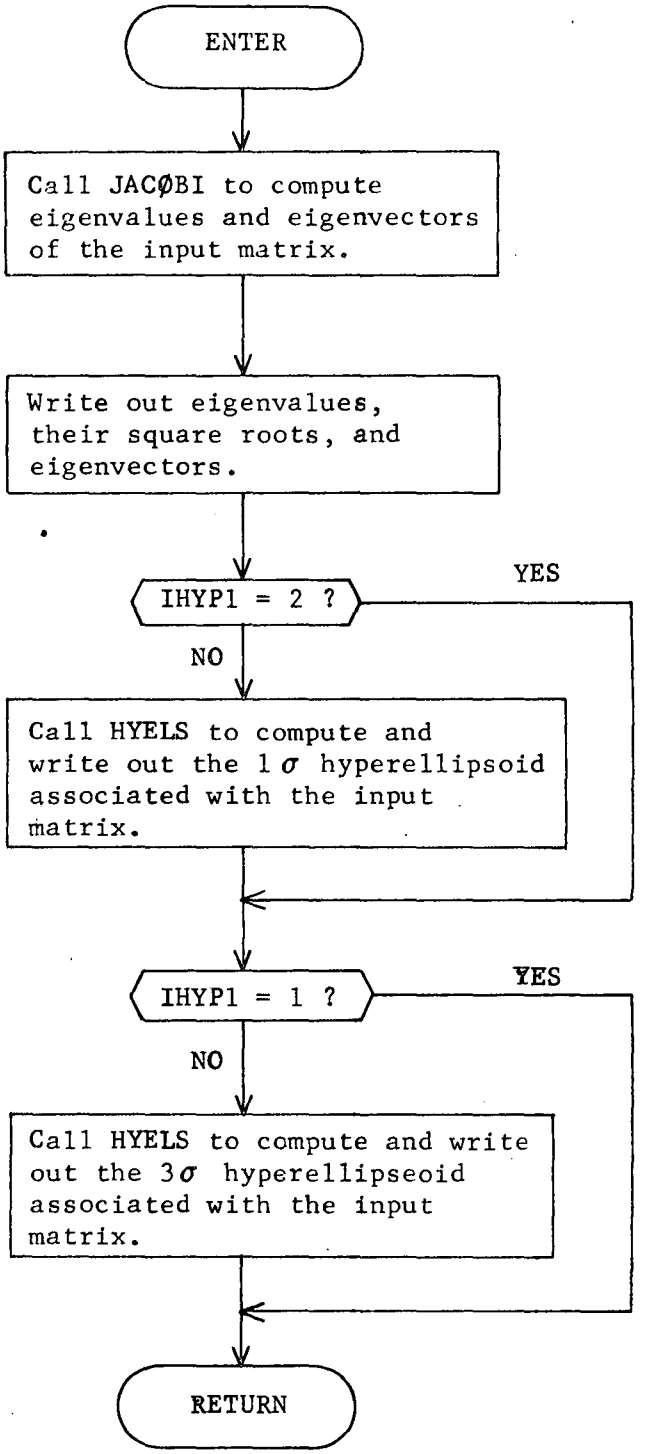
The program permits the entire trajectory to be divided into three arbitrary consecutive intervals, over each of which a different constant unmodelled acceleration $\vec{a}$ can be specified. These intervals are represented by (t_o, t_1), (t_1, t_2), and (t_2, t_f), where t_o is the initial trajectory time and t_f is the final trajectory time. If t_k and t_{k+1} occur in different intervals, then the above equations must be evaluated piece-wise over (t_k, t_{k+1}).


ENTER
Is Q or $\vec{\omega}$ to be computed?
Q
$\vec{\omega}$
Compute dynamic noise covariance matrix Q .
RETURN
Initialize $\vec{\omega}$ to zero at t_k .
70
YES
$t_k > t_1$?
No
$t_{k+1} > t_1$?
YES
50
No
I = 1 (1st interval)
$\Delta t = t_{k+1} - t_k$
IC = 1
40
Compute Δt^2 .
Compute actual dynamic noise $\vec{\omega}$ at the end of the interval Δt .
RETURN
1
IC = ?
2
3
4
5
51
61
62
81
I = 2
$\Delta t = t_{k+1} - t_1$
IC = 1
I = 2
$\Delta t = t_2 - t_1$
IC = 4
I = 3
$\Delta t = t_{k+1} - t_2$
IC = 1
I = 2
$\Delta t = t_2 - t_k$
IC = 1
40


70
$t_k > t_2$?
YES
No
90
$t_{k+1} > t_2$?
YES
No
I = 3 (3rd interval)
$\Delta t = t_{k+1} - t_k$
IC = 1
I = 2 (2nd interval)
$\Delta t = t_{k+1} - t_k$
IC = 1
80
I = 3 (3rd interval
$\Delta t = t_{k+1} - t_2$
IC = 5
40


50
$t_{k+1} > t_2$?
YES
No
60
I = 1 (1st interval)
$\Delta t = t_1 - t_k$
IC = 2
I = 1 (1st interval)
$\Delta t = t_1 - t_k$
IC = 3
40

EIGHY Flow Chart



ELCAR Analysis

ELCAR transforms the standard conic elements of a massless point referenced to a gravitational body to cartesian position and velocity components with respect to that body.

Let the gravitational constant of the body be denoted μ and the given conic elements $(a, e, i, \omega, \Omega, f)$. The semilatus rectum p is

$$p = a(1 - e^2) \tag{1}$$

Then the magnitude of the radius vector is given by

$$r = \frac{p}{1 + e \cos f} \tag{2}$$

The unit vector in the direction of the position vector is

$$\begin{aligned} u_x &= \cos(\omega + f)\cos\Omega - \cos i \sin(\omega + f)\sin\Omega \\ u_y &= \cos(\omega + f)\sin\Omega + \cos i \sin(\omega + f)\cos\Omega \\ u_z &= \sin(\omega + f)\sin i \end{aligned} \tag{3}$$

The position vector $\vec{r}$ is therefore

$$\vec{r} = r\hat{u} \tag{4}$$

The velocity vector $\vec{v}$ is given by

$$\begin{aligned} v_x &= \sqrt{\frac{\mu}{p}}\left[(e + \cos f)(-\sin\omega\cos\Omega - \cos i \sin\Omega\cos\omega)\right. \\ &\qquad \left. -\sin f(\cos\omega\cos\Omega - \cos i \sin\Omega\sin\omega)\right] \\ v_y &= \sqrt{\frac{\mu}{p}}\left[(e + \cos f)(-\sin\omega\sin\Omega + \cos i \cos\Omega\cos\omega)\right. \\ &\qquad \left. -\sin f(\cos\omega\sin\Omega + \cos i \cos\Omega\sin\omega)\right] \\ v_z &= \sqrt{\frac{\mu}{p}}\left[(e + \cos f)\sin i\cos\omega - \sin f \sin i \sin\omega\right] \end{aligned} \tag{5}$$

The conic time from periapsis t_p is computed from different formulae depending upon the sign of the semi-major axis. For $a > 0$ (elliptical motion)

$$t_p = \sqrt{\frac{a^3}{\mu}}(E - e \sin E)$$

$$\cos E = \frac{e + \cos f}{1 + e \cos f} \qquad \sin E = \frac{\sqrt{1 - e^2}\ \sin f}{1 + e \cos f} \tag{6}$$

For $a < 0$ (hyperbolic motion) the time from periapsis is

$$t_p = \sqrt{\frac{a^3}{\mu}}\ (e \sinh H - H)$$

$$\tanh \frac{H}{2} = \sqrt{\frac{e - 1}{e + 1}}\ \tan \frac{f}{2} \tag{7}$$

EPHEM Analysis

EPHEM first determines the current value for the mean anomaly of the pertinent body. The mean anomaly M is computed from

$$M = M_0 + M_1 t + M_2 t^2 + M_2 t^3 \qquad \text{for inner planets}$$

$$M = M_0 + M_1 t \qquad \text{for outer planets}$$

$$M = L_0 + L_1 t + L_2 t^2 + L_3 t^3 - \tilde{\omega}(t) \qquad \text{for the moon}$$

Kepler's equation $M = E - e \sin E$ is then solved iteratively to determine the eccentric anomaly E. The subsequent computations are basic conic manipulations:

$$p = a(1 - e^2)$$

$$r = a(1 - e \cos E)$$

$$v = \sqrt{\mu\left(\frac{2}{r} - \frac{1}{a}\right)}$$

$$\cos f = \frac{p - r}{er} \qquad \sin f = \sqrt{1 - \cos^2 f}\ \ \text{sgn}(\sin E)$$

$$\cos \gamma = \frac{\sqrt{\mu p}}{rv} \qquad \sin \gamma = \sqrt{1 - \cos^2} \qquad \text{sgn}(\sin E)$$

$$\omega = \tilde{\omega} - \Omega$$

The cartesian position and velocity relative to the reference body are then

$$\bar{r} = r_x \hat{i} + r_y \hat{j} + r_z \hat{k}$$

$$r_x = r \cos(\omega + f) \cos\Omega - r \sin(\omega + f) \sin\Omega \cos i$$
$$r_y = r \cos(\omega + f) \sin\Omega + r \sin(\omega + f) \cos\Omega \cos i$$
$$r_z = r \sin(\omega + f) \sin i$$

$$\vec{v} = \frac{v}{r}\left[(\hat{w} \times \vec{r}) \cos\gamma + \vec{r} \sin\gamma\right]$$

where $\hat{w} = (\sin i \sin\Omega)\,\hat{i} - (\sin i \cos\Omega)\,\hat{j} + (\cos i)\,\hat{k}$

When option 1 is used, the reference body for all the planets is the sun while the reference body for the moon is the earth.

When option 2 is used with heliocentric inertial coordinates, the cartesian state of the earth is added to the cartesian state of the moon to convert the state of the moon to heliocentric coordinates before storing that state in the F-array.

When option 2 is used with barycentric inertial coordinates, subroutine CENTER is called to convert all elements to barycentric coordinates before storing in the F-array.

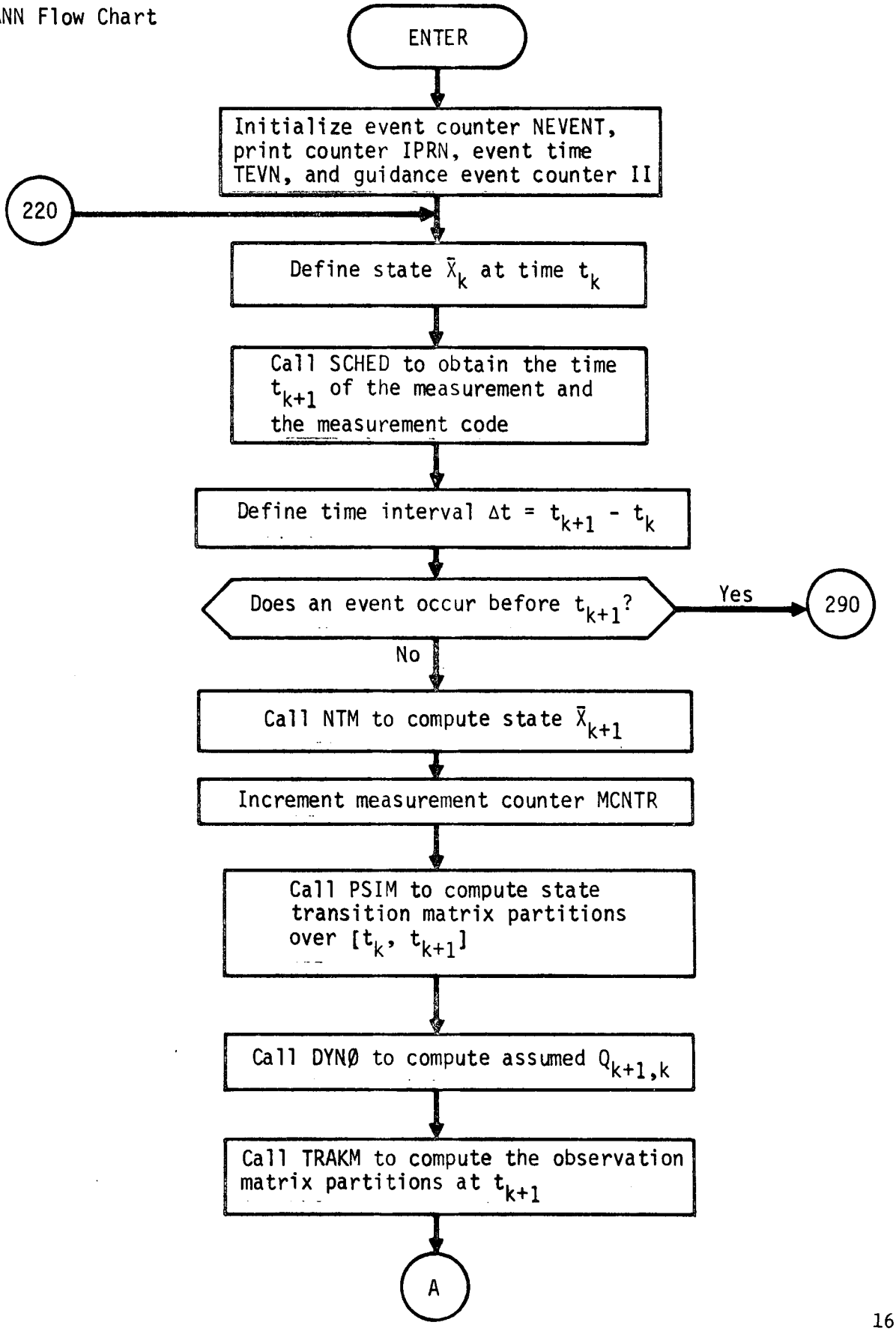
ERRANN Analysis

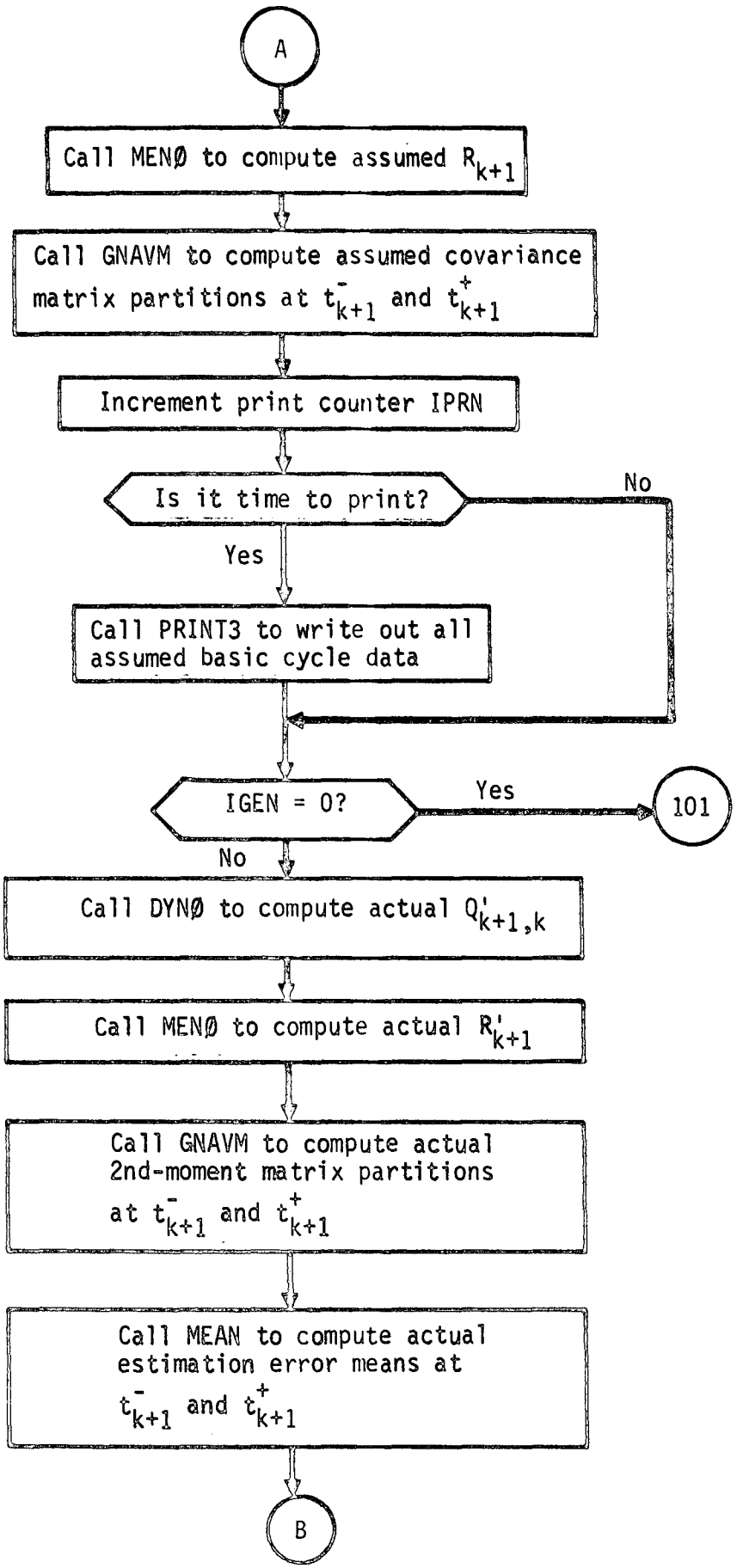
Subroutine ERRANN controls the computational flow through the basic cycle (measurement processing) and all events in the error analysis/ generalized covariance analysis program.

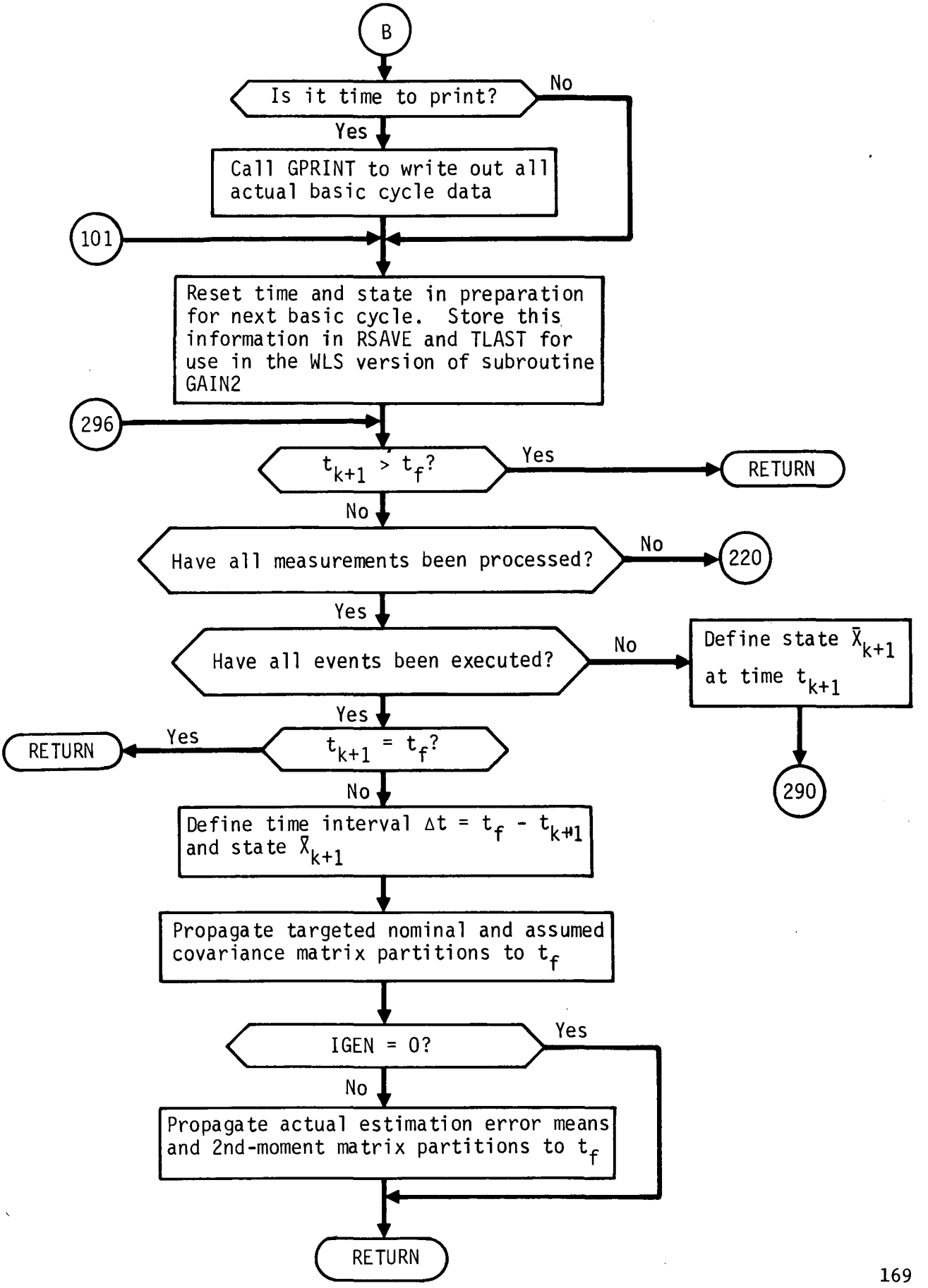
In the basic cycle the first task of ERRANN is to control the generation of the targeted nominal spacecraft state $\bar{X}_{k+1}$ at time t_{k+1}, given the state $\bar{X}_k$ at time t_k. Then calling PSIM, DYNØ, TRAKM, and MENØ, successively, ERRANN controls the computation of all matrix information required by subroutine GNAVM to compute the actual and assumed knowledge covariance matrix partitions at time t^+_{k+1} immediately following the measurement.

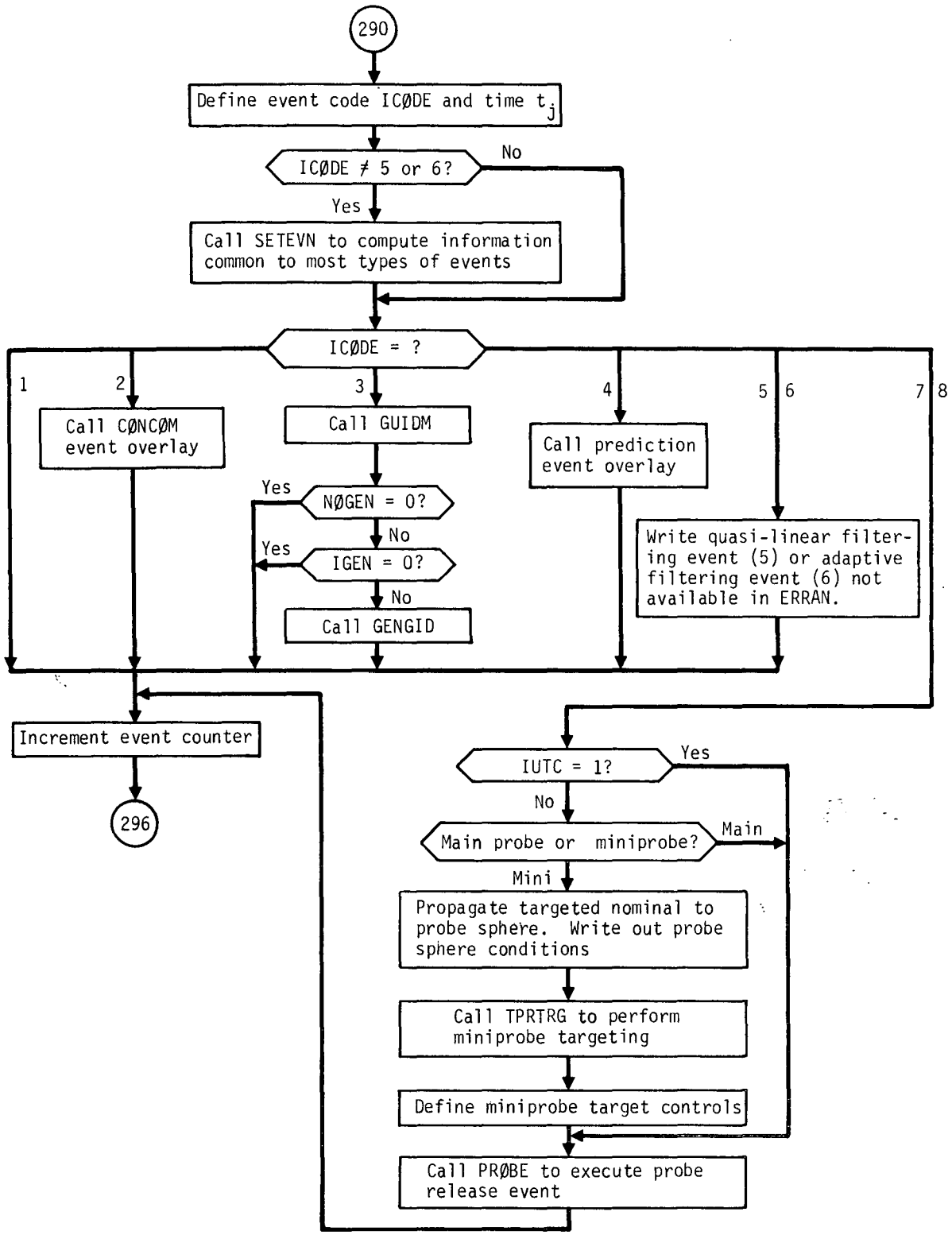
At an event, ERRANN simply calls the proper event subroutine or overlay where all required computations are performed. Subroutine ERRANN also controls miniprobe targeting in the error analysis program.

ERRANN Flow Chart




A
Call MENØ to compute assumed R_{k+1}
Call GNAVM to compute assumed covariance matrix partitions at t^-_{k+1} and t^+_{k+1}
Increment print counter IPRN
Is it time to print?
No
Yes
Call PRINT3 to write out all assumed basic cycle data
IGEN = 0?
Yes
101
No
Call DYNØ to compute actual $Q'_{k+1,k}$
Call MENØ to compute actual R'_{k+1}
Call GNAVM to compute actual 2nd-moment matrix partitions at t^-_{k+1} and t^+_{k+1}
Call MEAN to compute actual estimation error means at t^-_{k+1} and t^+_{k+1}
B


B
Is it time to print?
No
Yes
Call GPRINT to write out all actual basic cycle data
101
Reset time and state in preparation for next basic cycle. Store this information in RSAVE and TLAST for use in the WLS version of subroutine GAIN2
296
$t_{k+1} > t_f$?
Yes
RETURN
No
Have all measurements been processed?
No
220
Yes
Have all events been executed?
No
Define state $\bar{X}_{k+1}$ at time t_{k+1}
290
Yes
$t_{k+1} = t_f$?
Yes
RETURN
No
Define time interval $\Delta t = t_f - t_{k+1}$ and state $\bar{X}_{k+1}$
Propagate targeted nominal and assumed covariance matrix partitions to t_f
IGEN = 0?
Yes
No
Propagate actual estimation error means and 2nd-moment matrix partitions to t_f
RETURN


290
Define event code ICØDE and time t_j
ICØDE ≠ 5 or 6?
No
Yes
Call SETEVN to compute information common to most types of events
ICØDE = ?
1
2
3
4
5
6
7
8
Call CØNCØM event overlay
Call GUIDM
Call prediction event overlay
Yes
NØGEN = 0?
No
Yes
IGEN = 0?
No
Call GENGID
Write quasi-linear filtering event (5) or adaptive filtering event (6) not available in ERRAN.
Increment event counter
296
IUTC = 1?
Yes
No
Main probe or miniprobe?
Main
Mini
Propagate targeted nominal to probe sphere. Write out probe sphere conditions
Call TPRTRG to perform miniprobe targeting
Define miniprobe target controls
Call PRØBE to execute probe release event

ESTMT Analysis

The initial values of the state variables are first set equal to the values at the end of the previous interval. The nominal time interval to be used during the current step is computed from

$$\Delta t_k = \frac{c_2 \, r_{VS_B}}{V_{VS_B}} \tag{1}$$

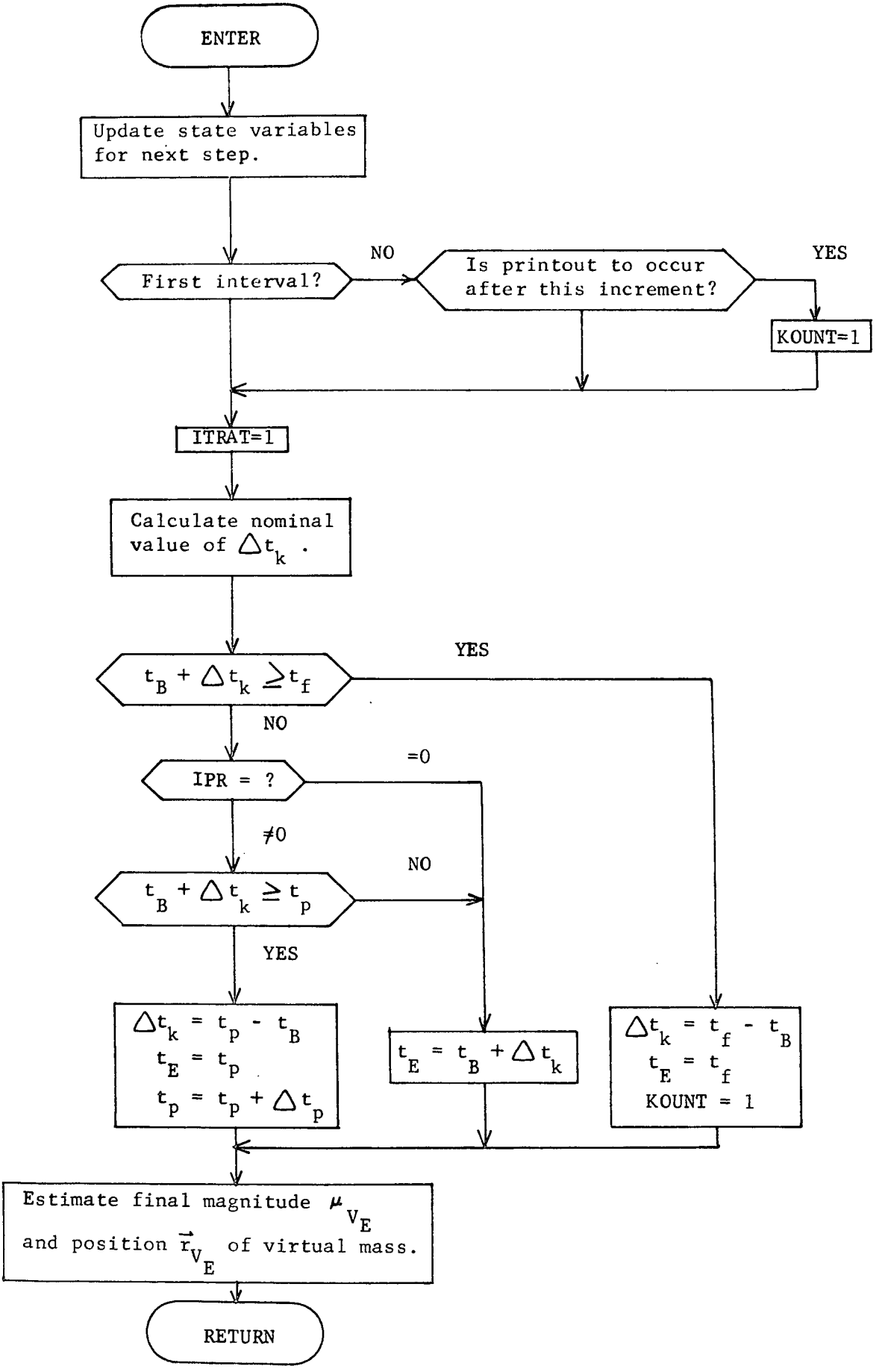
where c_2 is the constant input true anomaly increment relative to the virtual mass trajectory.

The time interval to the final time t_f or to the next time printout t_p is computed and the current time interval Δt is adjusted if necessary.

Finally the virtual mass final position and magnitude are estimated by the expansions

$$\begin{aligned} \mu_{V_E} &= \mu_{V_B} + \dot{\mu}_{V_B} \Delta t + \ddot{\mu}_V \Delta t^2 \\ \vec{r}_{V_E} &= \vec{r}_{V_B} + \dot{\vec{r}}_{V_B} \Delta t + \ddot{\vec{r}}_{V_{av}} \Delta t^2 \end{aligned} \tag{2}$$

ESTMT Flow Chart



EXCUTE Analysis

EXCUTE is the executive subroutine controlling the actual execution of the velocity increment $\vec{\Delta v}$. The $\vec{\Delta v}$ is computed by TARGET or INSERS or read in by the user.

Before executing the correction EXCUTE computes peripheral information of interest to the user. It first determines the dominant body acting on the spacecraft. If the spacecraft is in the moon's SOI (with respect to the earth), the moon is the dominant body. If not in the moon's SOI but in any of the planets' SOI (with respect to the sun) that planet is the dominant body. Otherwise the sun is the dominant body.

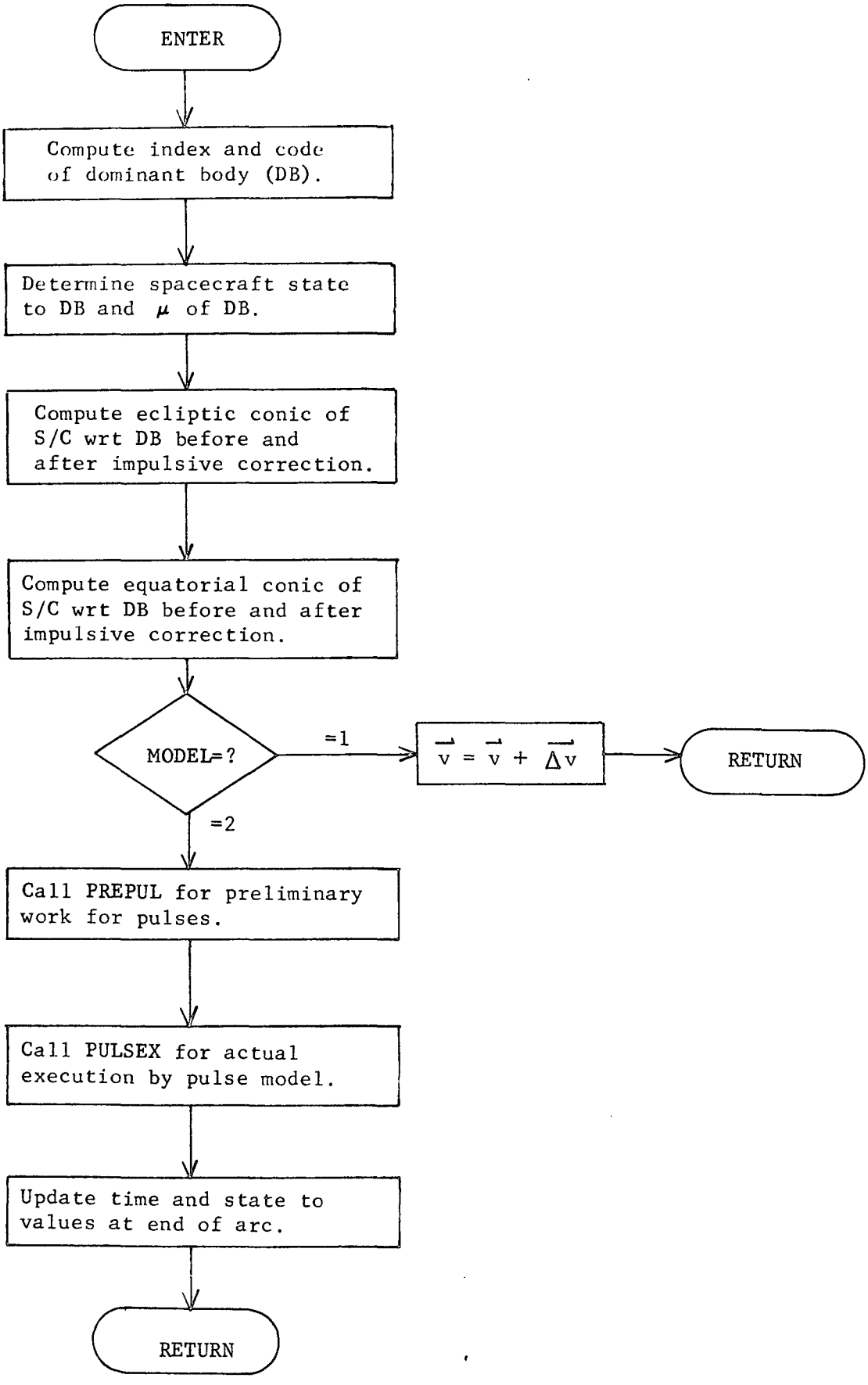
Having determined the dominant body EXCUTE computes the state of the spacecraft relative to that body. It then computes the conic elements of the trajectory both before and after an impulsive addition of the $\vec{\Delta v}$ in ecliptic coordinates.

If the dominant body is not the sun, it makes the same computations in equatorial coordinates.

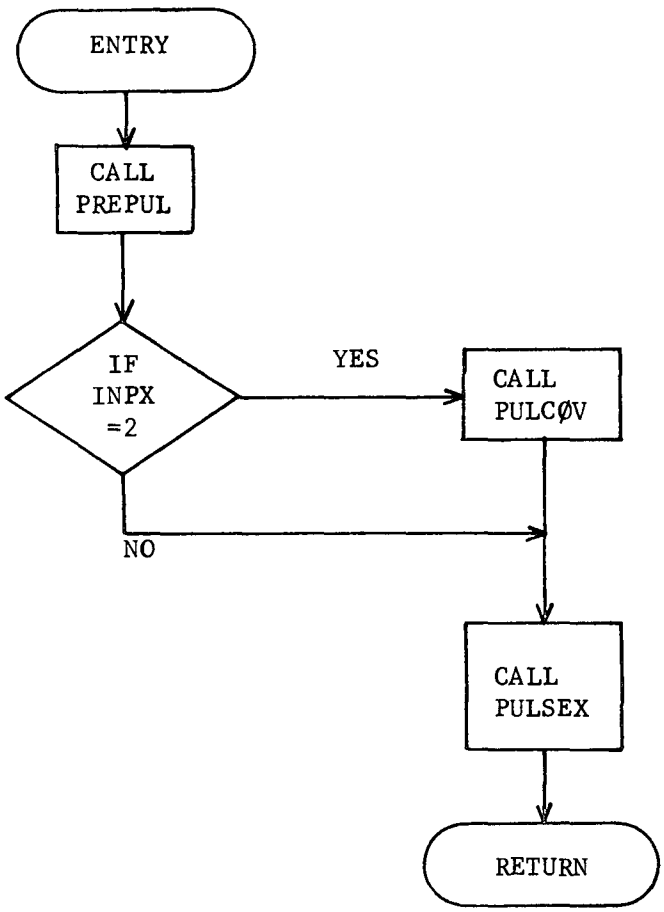
EXCUTE then operates on the current value MODEL of the array MDL. If MODEL = 1, the impulsive model of execution is commanded. The $\vec{\Delta v}$ is therefore added to the current inertial ecliptic velocity before returning to GIDANS.

If MODEL = 2, the pulsing arc model of execution is required. PREPUL is called to perform the preliminary work needed for the pulsing arc. PULSEX then actually propagates the trajectory through the series of pulses. At the completion of the arc EXCUTE updates the time and inertial ecliptic state (both position and velocity) of the nominal trajectory to the state determined by PULSEX.

EXCUTE Flow Chart



EXCUTS Flow Chart



FLITE Analysis

FLITE solves the time of flight equation (Lambert's theorem) using Battin's universal equation formulation. Stated functionally Lambert's theorem states that the time of flight t_f is a function

$$t_f = t_f (r_1 + r_2, c, a) \tag{1}$$

solely of the sum $r_1 + r_2$ of the distances of the initial and final points of the trajectory from the central body, the length c of the chord joining these points, and the length of the semimajor axis a of the trajectory. Usually the time of flight is known and it is desired to solve for the semimajor axis. The standard formulation involves different equations for the elliptic, parabolic, and hyperbolic cases, all of which then iterate on a to determine the solution.

In Battin's approach the semimajor axis a is replaced by a new variable x. By further introducing two new transcendental functions $S(x)$ and $C(x)$, the special cases of the flight-time equation are combined into one single, better behaved formula. The functions $S(x)$ and $C(x)$ are defined by

$$\begin{aligned}
S(x) &= \frac{\sqrt{x} - \sin\sqrt{x}}{\sqrt{x}^3} & C(x) &= \frac{1 - \cos\sqrt{x}}{x} & x > 0 \\
&= \frac{\sinh\sqrt{-x} - \sqrt{-x}}{\sqrt{x}^3} & &= \frac{\cosh\sqrt{-x} - 1}{-x} & x < 0 \\
&= \frac{1}{6} & &= \frac{1}{2} & x = 0
\end{aligned} \tag{2}$$

A parameter Q is introduced as

$$Q = \frac{s-c}{s}$$

where

$$c = (r_1^2 + r_2^2 - 2 r_1 r_2 \cos\theta)^{\frac{1}{2}}$$

$$s = \frac{1}{2}(r_1 + r_2 + c) \tag{3}$$

The universal flight-time formula is

$$T = \frac{S(x)}{C^{3/2}(x)} \mp Q^{3/2} \frac{S(y)}{C^{3/2}(y)}$$

$$yC(y) = Q \times C(x) \qquad (4)$$

where $T = \sqrt{\frac{\mu}{s^3}}\, t_f$. The choice of the upper or lower sign is made according to whether the transfer angle θ is less or greater than 180° respectively.

The development of equations (4) is too long and complex to be given here. It may be obtained from the first reference listed below. The following steps of that reference are noted:

(1) the two body problem on pp. 15,16
(2) the "vis viva" equation and Kepler's equation on pp. 50,51
(3) Lambert's theorem proved from Kepler's equation on p. 71
(4) the basic flight-time formula and detailed analysis on pp. 72-78
(5) The universal formulation on pp. 80,81.

Instead of using the equations (4) the authors of refence 2 (listed below) determined y as a function of x as

$$y = 4 \arcsin^2 \sqrt{\left|\frac{xsC(x)}{2}\right|} \qquad x \geq 0$$

$$= -4 \ln^2 \left\{ \sqrt{\left|\frac{xsC(x)}{2}\right|} + \sqrt{\left|\frac{xsC(x)}{2}\right| + 1} \right\}^{\frac{1}{2}} \qquad x < 0 \qquad (5)$$

Therefore a single variable iteration is possible. Newton's method is used to solve (4a) given T and Q as

$$x = x - \frac{T(x_n) - T}{T'(x_n)} \qquad (6)$$

where $$T(x) = \frac{S(x)}{C^{3/2}(x)} + Q^{3/2} \frac{S(y)}{C^{3/2}(y)} \qquad (7)$$

$$T'(x) = \frac{1 + k\left[\mp Q^{3/2} - 1.5\sqrt{2\text{-}yC(y)}\; T(x)\right]}{2x\sqrt{C(x)}} \qquad (8)$$

$$k = \operatorname{sgn}(\pi^2 - x)\sqrt{\frac{2-xC(x)}{2-yC(y)}}\ , \tag{9}$$

As $|2-yC(y)| \longrightarrow 0$, $k \longrightarrow 1$. Therefore if $|2-yC(y)| < 10^{-4}$ k is set to 1. Also $T'(x)$ breaks down as $x \longrightarrow 0$. Therefore the approximation is used:

$$T'(x) = \frac{1 \mp Q^{3/2}}{2\pi^2} \qquad |x| < 10^{-6} \tag{10}$$

The starting value for x is given by $x = x_1 - \Delta x(T,Q)$ where

$$x_1 = 82.1678 + 352.8045\ T - (123954.8504\ T^2 + 43904.0083\ T + 13423.6819)^{\frac{1}{2}}$$

$$\Delta x(T,Q) = \mp\left(\frac{2.36}{T^2 + .15} + \frac{3.5}{T + .1}\right)(0.3\ Q^2 + 0.7\ Q) \tag{11}$$

To insure that the routine will not fail for large or small values of T certain restrictions on T are built into the program. The nominal value of T is forced to be no larger than 950,000 and no smaller than 10^{-6}. This forces the corresponding limits for x of $-823.0473 \leq x \leq 39.14553$.

Finally convergence is achieved when $\left|T(x_n) - T\right| < \frac{T}{100000}$.

Having solved for semimajor axis a, the semilatus rectum p is given by

$$p = \frac{1}{2}\left\{\frac{r_1 r_2 \sin\theta}{c}\sqrt{\frac{1}{s-c} - \frac{1}{2a}} \pm \operatorname{sgn}(t_m - t)\sqrt{\frac{1}{s} - \frac{1}{2a}}\right\}^2 \tag{12}$$

Then the eccentricity e is given by

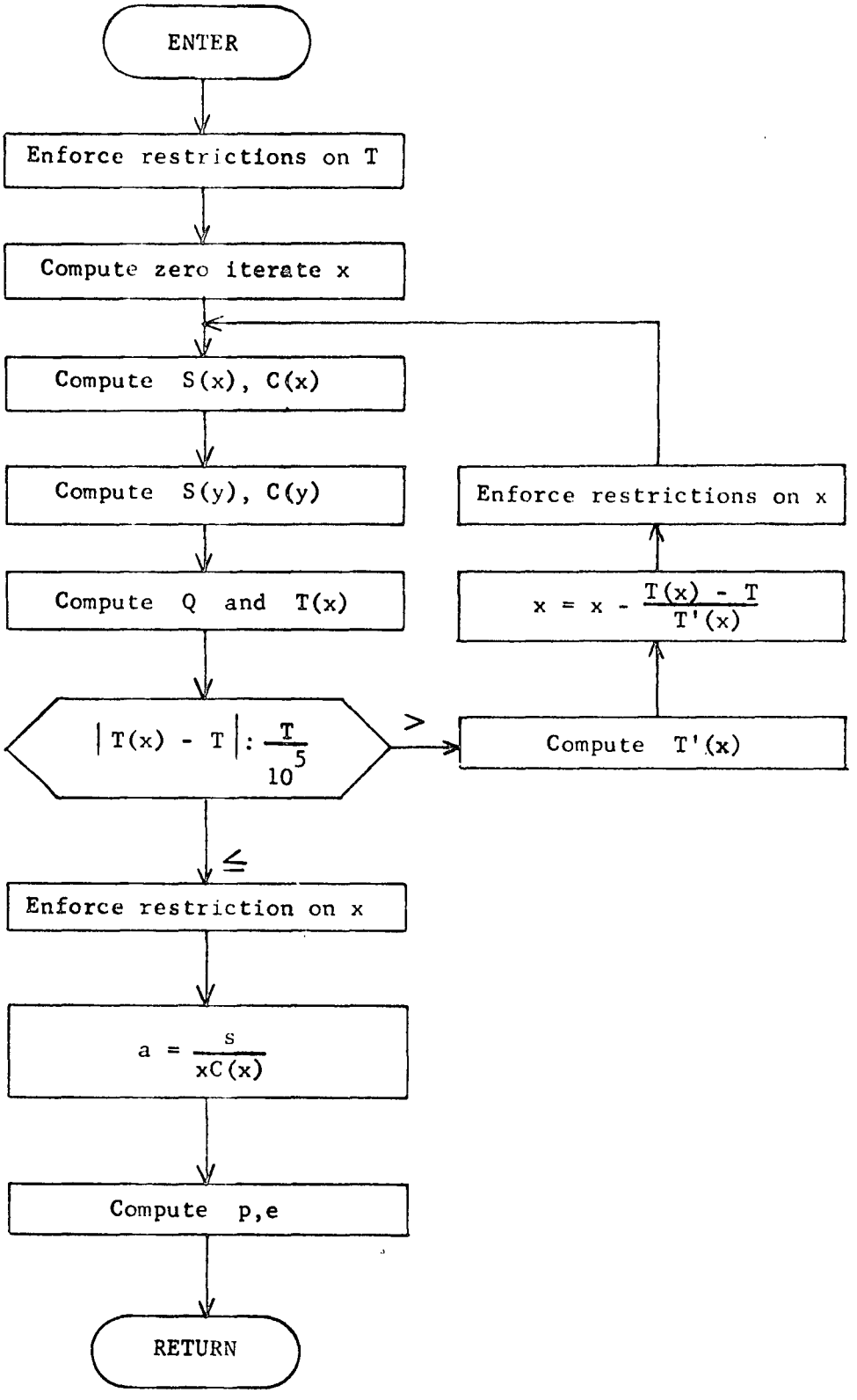
$$e = 1 - \frac{p}{a} \tag{13}$$

References:

(1) Battin, R. H., Astronomical Guidance, McGraw Hill Book Co., Inc., New York, 1964.

(2) Lesh, H. F., and Travis, C., FLIGHT: a Subroutine to Solve the Flight Time Problem, JPL Space Programs Summary 37-53, Vol. II.

FLITE Flow Chart



GAIN1 Analysis

Subroutine GAIN1 computes the Kalman-Schmidt filter gain matrices K_{k+1} and S_{k+1} that are used in subroutines GNAVM and NAVM to update estimation error covariance matrices after a measurement has been processed.

The measurement residual covariance matrix J_{k+1} and the auxiliary matrices A_{k+1} and B_{k+1} are assumed to be available (from GNAVM or NAVM) when GAIN1 is called. Subroutine GAIN1 then evaluates the following equations to determine the filter gain matrices:

$$K_{k+1} = A_{k+1} J_{k+1}^{-1} \tag{1}$$

$$S_{k+1} = B_{k+1} J_{k+1}^{-1} \tag{2}$$

GAIN2 Analysis

Subroutine GAIN2 computes filter gain matrices K_{k+1} and S_{k+1} for an equivalent recursive weighted-least-squares (WLS) consider filter. The equations required to compute K_{k+1} and S_{k+1} are identical to those used to compute the Kalman filter gains, but with all consider parameter covariances removed.

Subroutine GAIN2 propagates and updates (at a measurement) a set of covariance matrix partitions that are completely independent of those processed in subroutines NAVM or GNAVM for the sole purpose of generating filter gain matrices K_{k+1} and S_{k+1}.

The propagation and update equations employed in GAIN2, which are a subset of the NAVM and GNAVM propagation and update equations, are summarized below. For definitions of all matrices, see either the subroutine NAVM or GNAVM analysis section.

Propagation equations:

$$P^-_{k+1} = \left(\Phi P^+_k + \theta_{xx_s} C^{+T}_{xx_{s_k}}\right) \Phi^+ + C^-_{xx_{s_{k+1}}} \theta^T_{xx_s} + Q_k \tag{1}$$

$$C^-_{xx_{s_{k+1}}} = \Phi C^+_{xx_{s_k}} + \theta_{xx_s} P^+_{s_k} \tag{2}$$

$$P^-_{s_{k+1}} = P^+_{s_k} . \tag{3}$$

Gain equations:

$$A_{k+1} = P^-_{k+1} H^T_{k+1} + C^-_{xx_{s_{k+1}}} M^T_{k+1} \tag{4}$$

$$B_{k+1} = P^-_{s_{k+1}} M^T_{k+1} + C^-_{xx_{s_{k+1}}} H^T_{k+1} \tag{5}$$

$$J_{k+1} = H_{k+1} A_{k+1} + M_{k+1} B_{k+1} + R_{k+1} \tag{6}$$

$$K_{k+1} = A_{k+1} J_{k+1}^{-1} \tag{7}$$

$$S_{k+1} = B_{k+1} J_{k+1}^{-1} \tag{8}$$

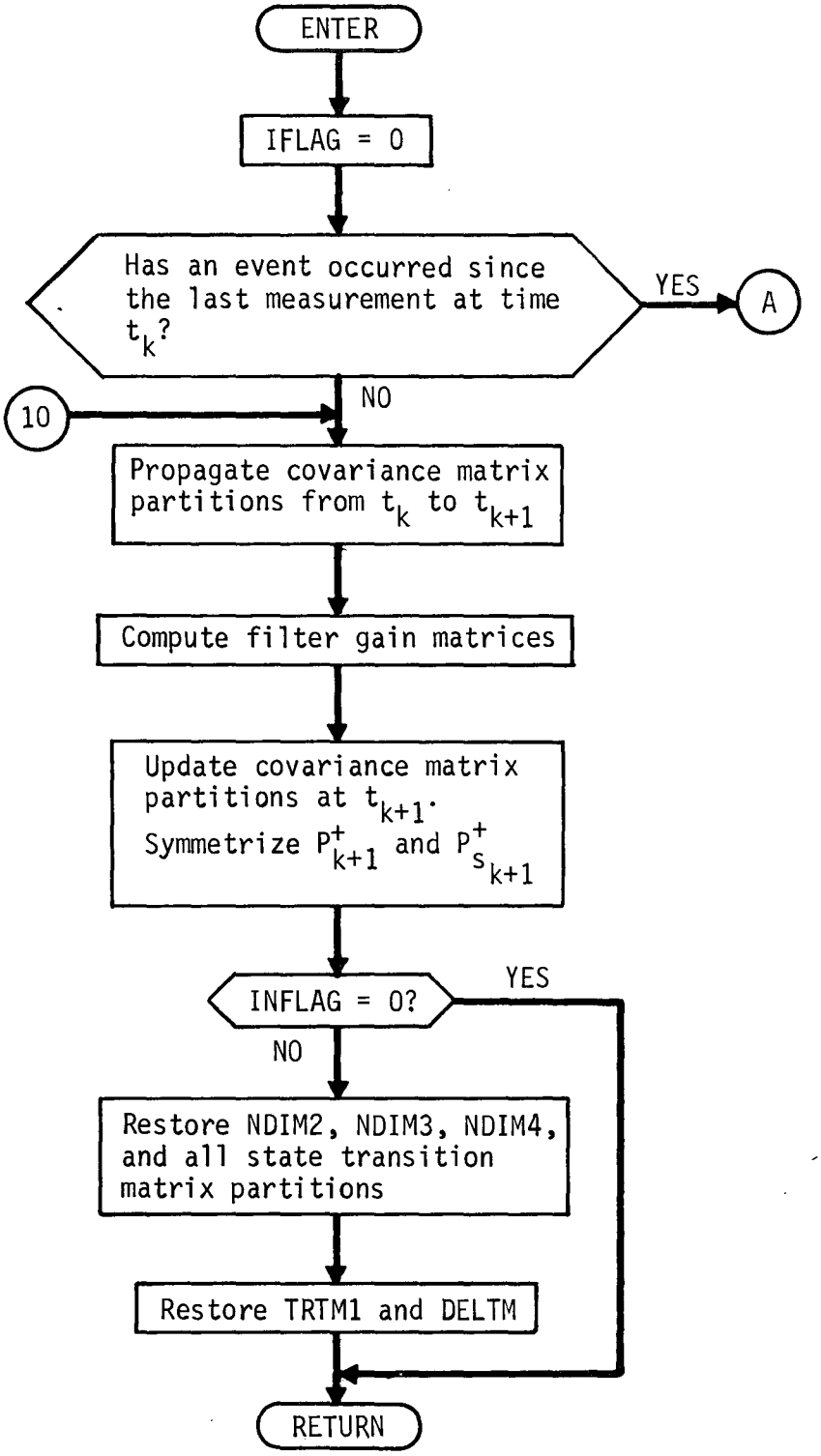
Update equations:

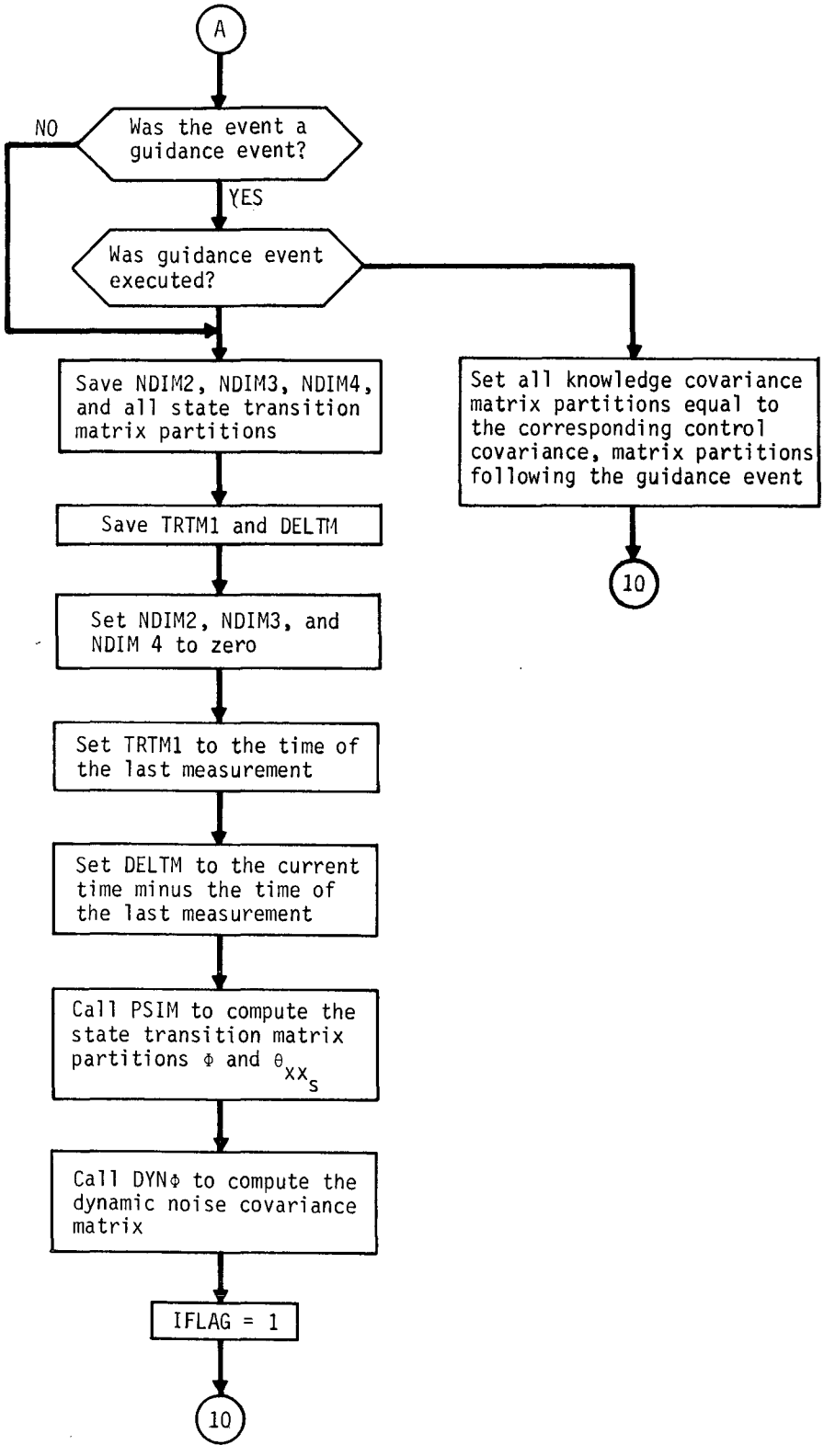
$$P_{k+1}^{+} = P_{k+1}^{-} - K_{k+1} A_{k+1}^{T} \tag{9}$$

$$C_{xx_{s_{k+1}}}^{+} = C_{xx_{s_{k+1}}}^{-} - K_{k+1} B_{k+1}^{T} \tag{10}$$

$$P_{s_{k+1}}^{+} = P_{s_{k+1}}^{-} - S_{k+1} B_{k+1}^{T} \tag{11}$$

GAIN2 Flow Chart




A
NO
Was the event a guidance event?
YES
Was guidance event executed?
Save NDIM2, NDIM3, NDIM4, and all state transition matrix partitions
Save TRTM1 and DELTM
Set NDIM2, NDIM3, and NDIM 4 to zero
Set TRTM1 to the time of the last measurement
Set DELTM to the current time minus the time of the last measurement
Call PSIM to compute the state transition matrix partitions Φ and θ_{xx_s}
Call DYNΦ to compute the dynamic noise covariance matrix
IFLAG = 1
10
Set all knowledge covariance matrix partitions equal to the corresponding control covariance, matrix partitions following the guidance event
10

GAUSLS Analysis

GAUSLS is a hybrid algorithm to obtain a least-squares solution to the system

$$\underline{\phi}\ (\underline{x}) = 0 \qquad (1)$$

where $\underline{x}$ is an n-dimensional control vector, $\underline{\phi}$ is an m-dimensional constraint vector, and $m \geq n$. Current array dimensions in GAUSLS require that $n \leq 5$ and $m \leq 10$. By least-squares we mean that the square of the standard Euclidean norm of $\underline{\phi}$, namely

$$||\underline{\phi}||^2 = \underline{\phi}^T \underline{\phi} \ , \qquad (2)$$

is minimized. The principal algorithm used is the well-known pseudo-inverse scheme originally due to Gauss. When, however, the dependence of $\underline{\phi}$ on $\underline{x}$ deviates substantially from the approximate linearity tactily assumed by the Gauss method, a best-step steepest descent algorithm is invoked. Either of two indications of nonlinearity can cause GAUSLS to transfer from the normal pseudo-inverse mode to best-step steepest descent technique: (1) the Gauss control correction is larger in norm than an input upper bound , s_0, or (2) the Gauss step actually increases the miss index, $||\underline{\phi}||^2$, over the previous iterate.

The Gauss procedure can readily be derived since it is simply the exact one-step solution to equation (1) when $\underline{\phi}$ depends on $\underline{x}$ linearly. Let J represent the Jacobian or sensitivity matrix of $\underline{\phi}$ with respect to $\underline{x}$; that is

$$J_{ij} = \frac{\partial \phi_i}{\partial x_j} \qquad \begin{matrix} i=1,\ldots,m \\ j=1,\ldots,n \\ m \geq n \end{matrix} \ . \qquad (3)$$

Next let y denote the least-squares miss index; that is

$$y = ||\underline{\phi}\ ||^2 \ . \qquad (4)$$

Then the gradient of the miss-index is simply

$$\underline{\nabla}y = 2\ J^T\ \underline{\phi} \quad . \tag{5}$$

Now a necessary condition for the miss-index to be minimized after a control correction of $\Delta\underline{x}$ is

$$\underline{\nabla}y\ (\underline{x} + \Delta\underline{x}) = \underline{0} \quad . \tag{6}$$

Substituting equation (5) into (6) gives

$$2\ J^T\ (\underline{x} + \Delta\underline{x})\ \underline{\phi}\ (\underline{x} + \Delta\underline{x}) = \underline{0} \quad . \tag{7}$$

Assuming J either constant or approximately so and using the first two terms of the Taylor's series for $\underline{\phi}$ yields the approximation

$$2\ J^T\ [\underline{\phi}(\underline{x}) + J\Delta\underline{x}] = \underline{0} \quad . \tag{8}$$

Solving for the control correction then yields the pseudo-inverse control correction

$$\Delta x = -(J^TJ)^{-1}\ J^T\underline{\phi}(\underline{x}) \quad . \tag{9}$$

Clearly equation (9) is exact if $\underline{\phi}$ is a linear function of $\underline{x}$ so that the Taylor series of $\underline{\phi}\ (\underline{x} + \Delta\underline{x})$ has only two terms and J is independent of $\underline{x}$. Since one can reasonably expect that if the dependence of $\underline{\phi}$ on $\underline{x}$ is approximately linear, formula (9) can be applied iteratively to yield a convergent sequence of control vectors converging to the least-squares solution and one arrives at the Gauss algorithm, namely,

$$\Delta\underline{x}_k = (J^TJ)^{-1}\ J^T\underline{\phi}(\underline{x}_k) \tag{10}$$

$$k = 0,1,2,\ldots$$

$$\underline{x}_{k+1} = \underline{x}_k + \Delta\underline{x}_k \tag{11}$$

where $\underline{x}_0$ is an initial control estimate suggested by other sources. GAUSLS requires as an input parameter this zero-iterate control estimate, together with the corresponding constraints $\underline{\phi}(\underline{x}_0)$.

Since equation (6) only guarantees an extremum of the miss index (i.e., a minimum, a maximum, or an inflection point), no more can be said for the Gauss algorithm. It must be assumed that the initial estimate of $\underline{x}_0$ is sufficiently near a local minimum and that y is well enough behaved that the algorithm indeed leads to that minimum. It is interesting to note that in the case that m = n, equations (10) and (11) reduce to the familiar Newton-Raphson scheme for solving nonlinear systems of equations.

The logic behind the steepest-descent mode is less elegant but more straightforward than the Gauss procedure. First, the gradient of the miss index is computed via equation (5). Next a search is conducted in the negative gradient direction until the miss index is observed to begin increasing. Let a denote the step length in the search direction where y is first observed to increase. Then the subroutine THPØSM is called to find a minimum of y on the step length interval from 0 to a by cubic interpolation. Let λ_m denote the step length value corresponding to the minimum returned by THPØSM. Then the control correction for the kth iterate is taken to be

$$\Delta\underline{x}_k = -\lambda_m \underline{\nabla}y/||\underline{\nabla}y|| \quad . \tag{12}$$

The convergence of this scheme is only asymptotic with no acceleration as the minimum-miss controls are approached. Nevertheless, the steepest descent algorithm seems to be the best available for extremely nonlinear miss indices since it involves no linear extrapolation and since it searches in the only direction in which improvement is guaranteed. Its poor terminal convergence is no handicap in the hybrid GAUSLS routine because once the iteration sequence falls inside a suitably linear region about the miss-index minimum, the rapidly convergent Gauss scheme takes over.

GAUSLS calculates the Jacobian matrix J by numerical differencing through a call to the subroutine JACØB. Hence the user is required to supply a perturbation size δ to GAUSLS for us by JACØB in approximating J by the forward-divided difference

$$J_{ij} \approx \frac{[\phi_i (x_j + \delta) - \phi_i (x_j)]}{\delta} \tag{13}$$

The user could conceivably use an analytical Jacobian matrix by replacing the call to JACØB by formulae for the appropriate partial derivatives.

The convergence criterion in either mode of GAUSLS is the same. Adequate convergence is assumed when a weighted sum of the length of the change in the control vector and the magnitude of the change in the miss index fall below a preassigned value; i.e.,

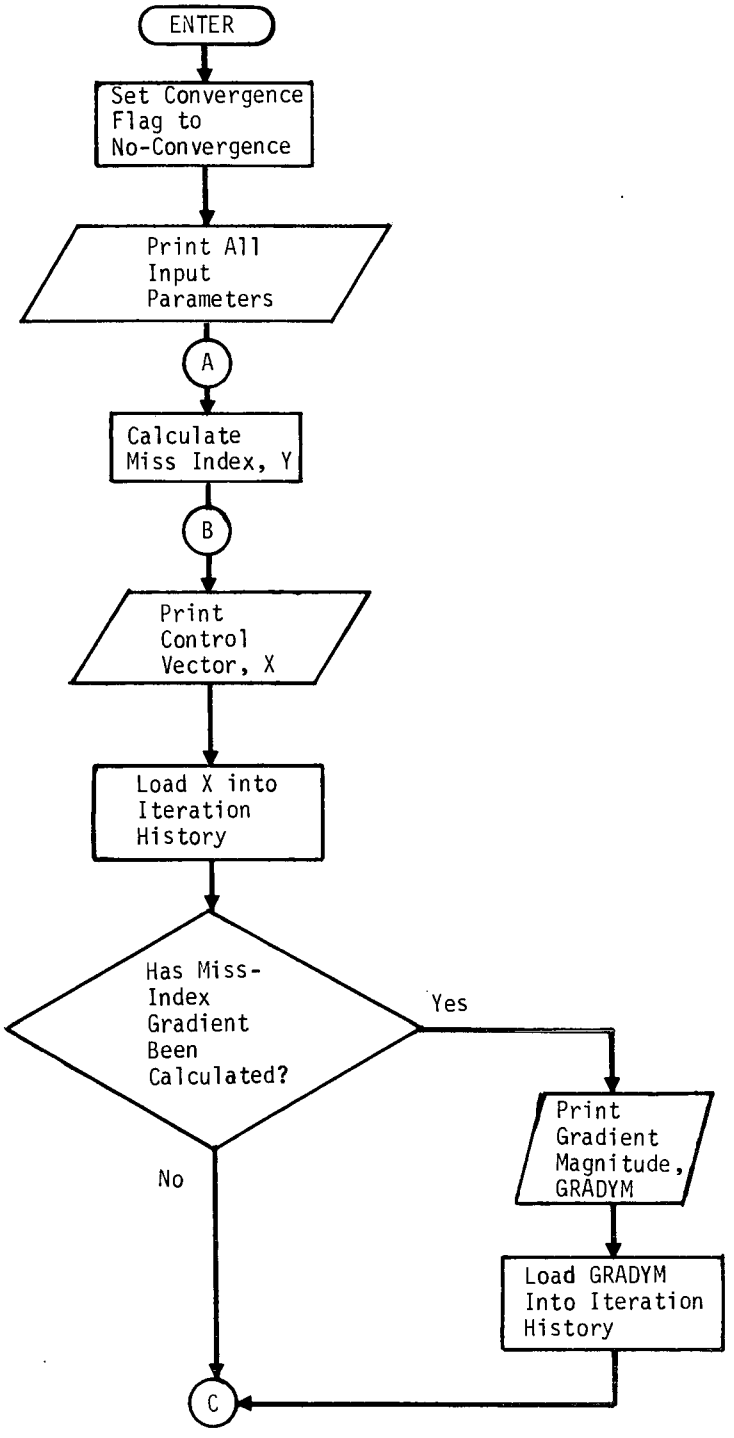
$$C_1 \, ||\Delta \underline{x}|| + C_2 \, |\Delta y| < \varepsilon \quad . \tag{14}$$

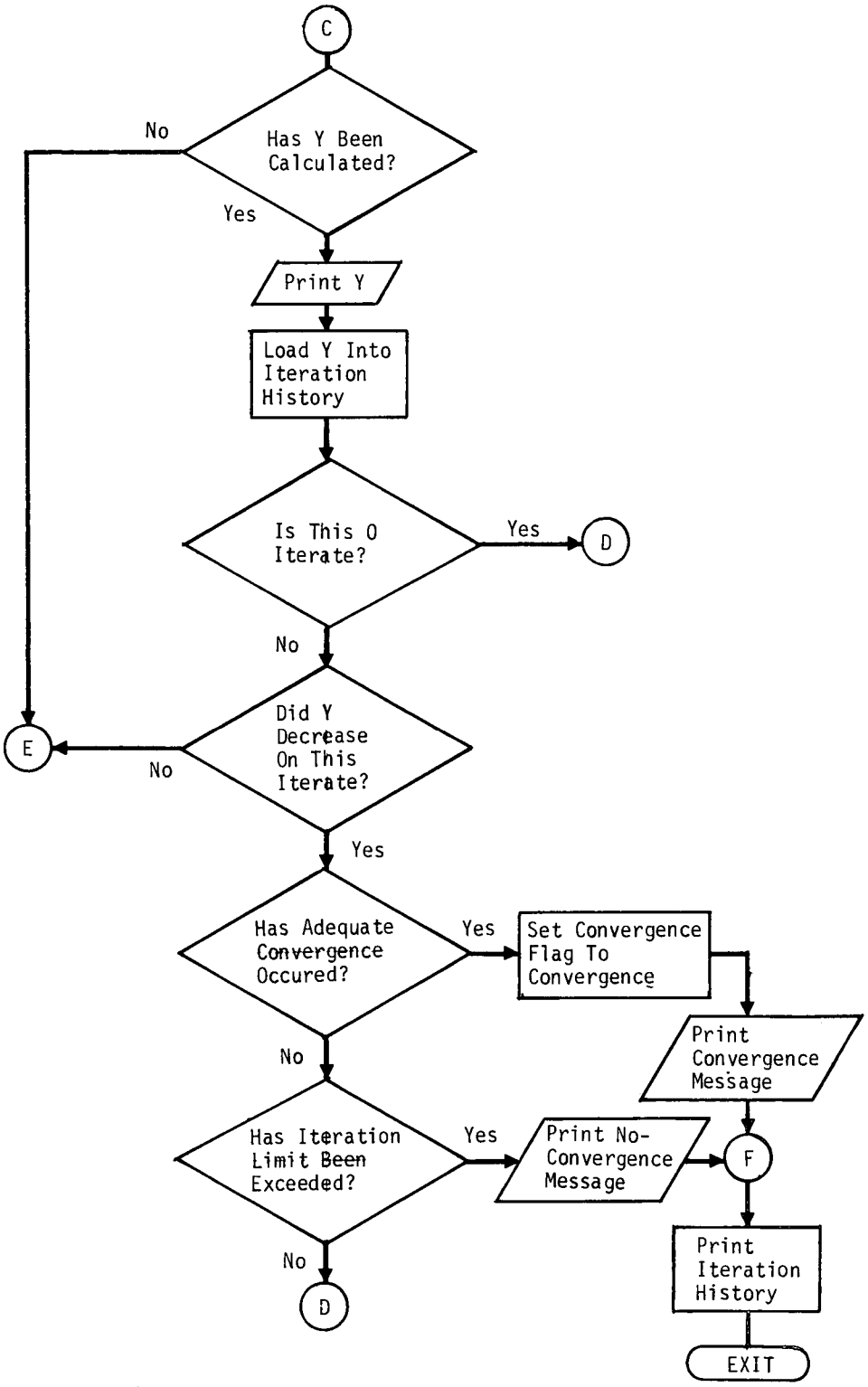
The user must supply C_1, C_2, and ε as input parameters to GAUSLS. To expedite convergence, the user should scale the components of $\underline{x}$ so they are, as far as possible, all of the same order of magnitude, say in the range from 0.1 to 10. This scaling makes meaningful the use of a single perturbation size δ for all components of $\underline{x}$ in approximating J and avoids numerical problems in matrix inversion and search direction calculation. Further he must supply as an input parameter the maximum number of iterations k_{max} he will allow before terminating the algorithm.

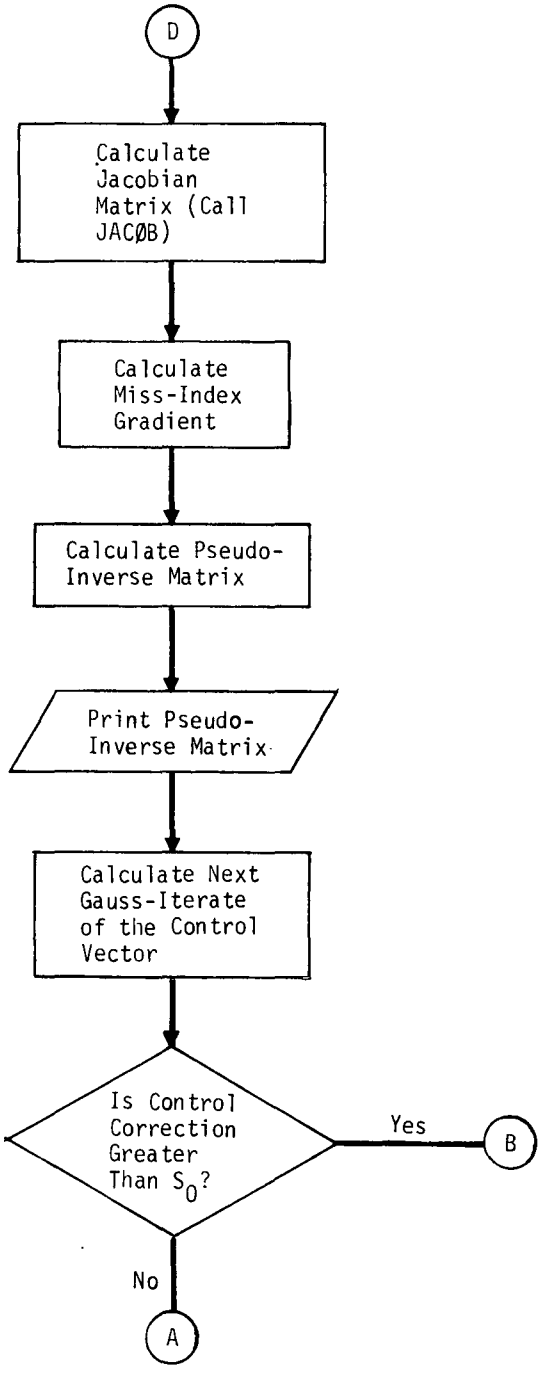
GAUSLS supplies enough output to adequately monitor either mode of the iterative least-squares process. Initially under the heading "Gauss Least-Squares Routine," it prints out all of the input parameters. These include n, m, δ, C_1, C_2, ε, s_0, and k_{max}. Next the user-supplied initial-control estimate $\underline{x}_0$, together with the corresponding miss index $y(\underline{x}_0)$, are printed out under the heading "Gauss Iteration Point." Then the printout relative to the general kth iterate begins. All data concerning the Jacobian matrix J are printed from the subroutine JACØB under the heading "Jacobian Matrix Routine." Each iterate, of course, starts with a Jacobian matrix computation even if it eventually ends in a steepest-descent step. All of the control vectors and corresponding constraint vectors that go into the approximation of the Jacobian matrix are printed under the heading "Nominal and Perturbed Function Values." The divided-difference approximation to J is then printed under the heading "Jacobian Matrix." Next GAUSLS prints out the Gauss pseudo inverse matrix, $(J^T J)^{-1} J^T$, under the heading "Projection Matrix." Finally the next Gauss control vector iterate, the corresponding miss index, and the gradient magnitude of the previous iterate are printed out under the heading "Gauss Iteration Point." If the length of the control correction $\Delta \underline{x}$ exceeds s_0, however, the miss index is neither calculated nor printed.

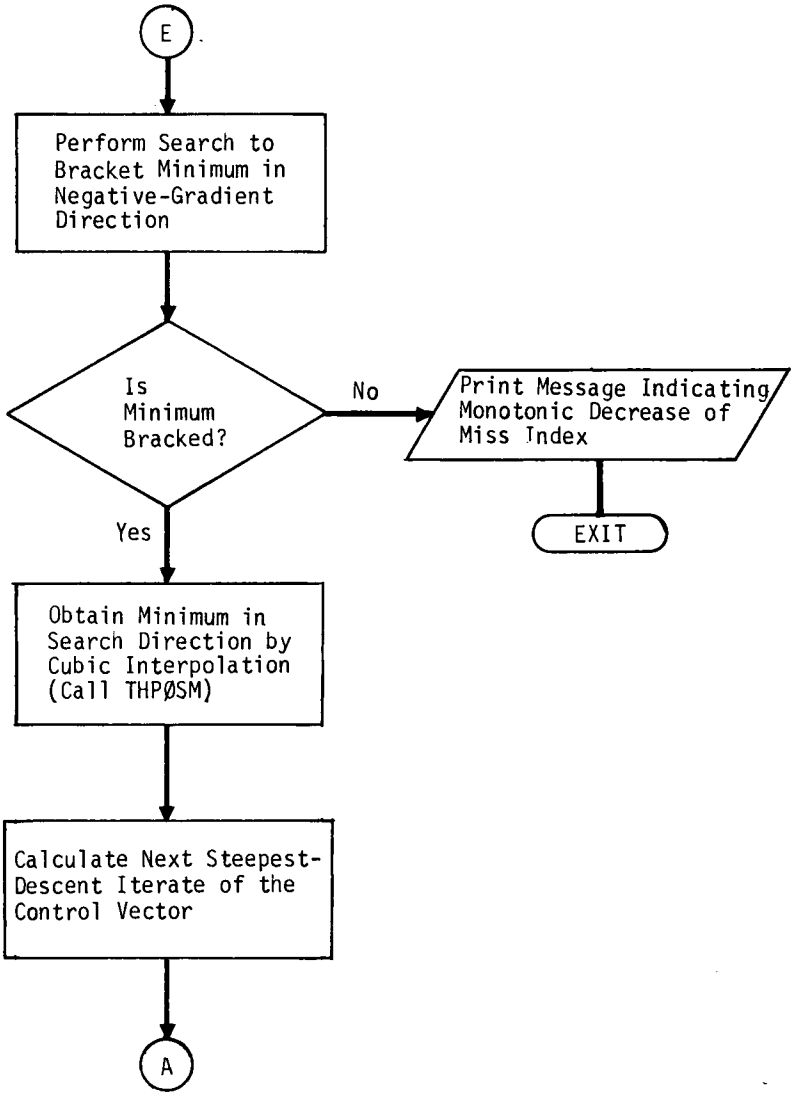
If the Gauss iterate is such that a steepest-descent step is required, GAUSLS prints out all of the pertinent data. Under the heading "Cubic Interpolation Routine" are printed all trial step lengths and corresponding miss indices used in bracketing the minimum, the input parameters to the routine THPØSM, and the minimum miss step length and index are returned by THPØSM. If the miss index decreases monotonically in the search direction, a message to that effect is printed out and execution of the program is stopped. Finally the steepest descent control iterate and the corresponding miss index is printed out under the heading "Best-Step Steepest-Descent Iteration Point." The iteration printout then is repeated with each successive iterate. When convergence finally occurs, the message "Adequate convergence occurred on previous step" is printed after the last iterate and the convergence flag, ICØNV1 is set to 1. If, on the other hand, convergence fails to occur in k_{max} iterations, the message "Convergence did not occur" is supplied after the last iteration point and ICØNV1 is set to 2. After either of these two stopping conditions is reached, a summary of the iteration points is printed under the heading "Iteration History." This summary contains the control vector, the miss index, and the gradient to the miss index at each of the iterates in consecutive order.

GAUSLS Flow Chart




C
Has Y Been Calculated?
No
Yes
Print Y
Load Y Into Iteration History
Is This 0 Iterate?
Yes
D
No
Did Y Decrease On This Iterate?
E
No
Yes
Has Adequate Convergence Occured?
Yes
Set Convergence Flag To Convergence
Print Convergence Message
No
Has Iteration Limit Been Exceeded?
Yes
Print No-Convergence Message
F
No
D
Print Iteration History
EXIT


D
Calculate Jacobian Matrix (Call JACØB)
Calculate Miss-Index Gradient
Calculate Pseudo-Inverse Matrix
Print Pseudo-Inverse Matrix
Calculate Next Gauss-Iterate of the Control Vector
Is Control Correction Greater Than S_0?
Yes
B
No
A


E
Perform Search to Bracket Minimum in Negative-Gradient Direction
Is Minimum Bracked?
No
Print Message Indicating Monotonic Decrease of Miss Index
EXIT
Yes
Obtain Minimum in Search Direction by Cubic Interpolation (Call THPØSM)
Calculate Next Steepest-Descent Iterate of the Control Vector
A

GENGID Analysis

Subroutine GENGID controls the execution of generalized guidance events. Generalized guidance has been extended to all guidance options defined for subroutine GUIDM except for biased aimpoint guidance and impulse series thrusting.

Unlike GUIDM, which computes target dispersions and fuel budgets based on filter-generated statistics, subroutine GENGID computes target dispersions and fuel budgets based on actual statistics. In other words, the generalized covariance technique as applied to the guidance process is programmed in GENGID. The required equations are summarized below.

Before the guidance event at time t_j can be executed, it is necessary to propagate the actual control mean and control 2nd-moment matrix partitions forward to t_j from the previous guidance event at time t_{j-1}. The control mean propagates according to

$$\bar{x}'^{-}_{j} = \Phi\, \bar{x}'^{+}_{j-1} + \theta_{xx_s}\, \bar{x}'_{s_o} + \theta_{xu}\, \bar{u}'_{o} + \theta_{xw}\, \bar{w}'_{o} \qquad (1)$$

where Φ, θ_{xx_s}, θ_{xu}, and θ_{xw} are state transition matrix partitions over the interval $\left[t_{j-1},\ t_j\right]$, and $\bar{x}'$, $\bar{x}'_s$, $\bar{u}'$, and $\bar{w}'$ denote actual position/velocity and solve-for, dynamic-consider, and ignore parameter deviation means. The notation $(\,)'$ indicates actual values as opposed to the unprimed assumed values, while $(\,)^-$ and $(\,)^+$ indicate values immediately before and after the execution of the guidance event, respectively. The actual control position/velocity 2nd-moment matrix is defined by

$$P_{c_j} = E\left[x'_j\, x'^{T}_j\right]. \qquad (2)$$

The remaining control 2nd-moment matrix partitions are defined similarly. The propagation equations appearing in subroutine GNAVM are used to propagate the control 2nd-moment matrix partitions over the interval $\left[t_{j-1},\ t_j\right]$.

The actual target state deviation $\delta\tau_j'$ is related to the actual state deviation x_j' at time t_j according to

$$\delta\tau_j' = \eta_j \, x_j' \tag{3}$$

where η_j is the variation matrix for the appropriate midcourse guidance policy. The mean of $\delta\tau_j'$ is given by

$$E\left[\delta\tau_j'\right] = \eta_j \, E\left[x_j'\right]. \tag{4}$$

The statistical target dispersions are represented by the actual target condition 2nd-moment matrix W_j', which is defined as

$$W_j' = E\left[\delta\tau_j' \, \delta\tau_j'^{T}\right]. \tag{5}$$

Substitution of equation (3) into equation (5) yields

$$W_j' = \eta_j \, P_{c_j}' \, \eta_j^{T} . \tag{6}$$

Equations (4) and (6) are evaluated immediately before and after the guidance correction to determine how much the target errors have actually been reduced by the velocity correction at t_j.

The actual commanded velocity correction 2nd-moment matrix is defined by

$$S_j' = E\left|\Delta\hat{V}_j' \, \Delta\hat{V}_j'^{T}\right| \tag{7}$$

where the actual commanded velocity correction is given by

$$\Delta\hat{V}_j' = \Gamma_j \, \hat{x}_j' = \Gamma_j \left(x_j' + \tilde{x}_j'\right). \tag{8}$$

The guidance matrix Γ_j corresponds to the appropriate linear midcourse guidance policy. The equation used to evaluate S_j' is given by

$$S_j' = \Gamma_j \left(P_{c_j}' - P_{k_j}'\right) \Gamma_j^{T} \tag{9}$$

where all $E\left[\hat{x}_j' \tilde{x}_j'^T\right]$ terms have been neglected in the derivation of equation (9).

The mean of the actual commanded velocity correction is obtained by applying the expectation operator to equation (8):

$$E\left[\Delta\hat{V}_j'\right] = \Gamma_j \left\{E\left[x_j'\right] + E\left[\tilde{x}_j'\right]\right\}. \tag{10}$$

Since this equation gives no useful information for fuel-sizing studies, the Hoffman-Young formula will be used to evaluate $E\left[\left|\Delta\hat{V}_j'\right|\right]$

$$E\left[\left|\Delta\hat{V}_j'\right|\right] = \sqrt{\frac{2A}{\pi}}\left(1 + \frac{B\ (\pi - 2)}{A^2\sqrt{5.4}}\right) \tag{11}$$

where

$$A = \text{trace } S_j'$$

$$B = \lambda_1' \lambda_2' + \lambda_1' \lambda_3' + \lambda_2' \lambda_3',$$

and λ_1', λ_2', and λ_3' are the eigenvalues of the 2nd-moment matrix S_j'.

The actual effective or statistical ΔV is defined as

$$\text{"}E\left[\Delta\hat{V}_j'\right]\text{"} = E\left[\left|\Delta\hat{V}_j'\right|\right] \cdot \alpha_j' \tag{12}$$

where α_j' denotes a unit vector in the most likely direction of the velocity correction. The most likely direction is assumed to be aligned with the eigenvector associated with the maximum eigenvalue of S_j'.

With "$E\left[\Delta\hat{V}_j'\right]$" available, the actual execution error statistics can be computed (by calling subroutine GQCØMP). These are the actual execution error mean $E\left[\delta\Delta V_j'\right]$ and 2nd-moment maxtrix $\tilde{Q}_j'$ defined as

$$\tilde{Q}_j' = E\left[\delta\Delta V_j' \ \delta\Delta V_j'^T\right]. \tag{13}$$

It remains to summarize the equations which are used to update all actual control and knowledge means and 2nd-moment matrix partitions immediately following the execution of a guidance event. The actual estimation error means and 2nd-moment matrix partitions are updated using the following equations:

$$E\left[\tilde{x}_j'^{+}\right] = E\left[\tilde{x}_j'^{-}\right] - A \cdot E\left[\delta\Delta V_j'\right] \tag{14}$$

$$E\left[\tilde{x}_{s_j}'^{+}\right] = E\left[\tilde{x}_{s_j}'^{-}\right] \tag{15}$$

$$P_{k_j}'^{+} = P_{k_j}'^{-} + A\,Q_j'\,A^T - A \cdot E\left[\delta\Delta V_j'\right] \cdot E\left[\tilde{x}_j'^{-T}\right] - E\left[\tilde{x}_j'^{-}\right] \cdot E\left[\delta\Delta V_j'^{T}\right] \cdot A^T \tag{16}$$

$$P_{s_{k_j}}'^{+} = P_{s_{k_j}}'^{-} \tag{17}$$

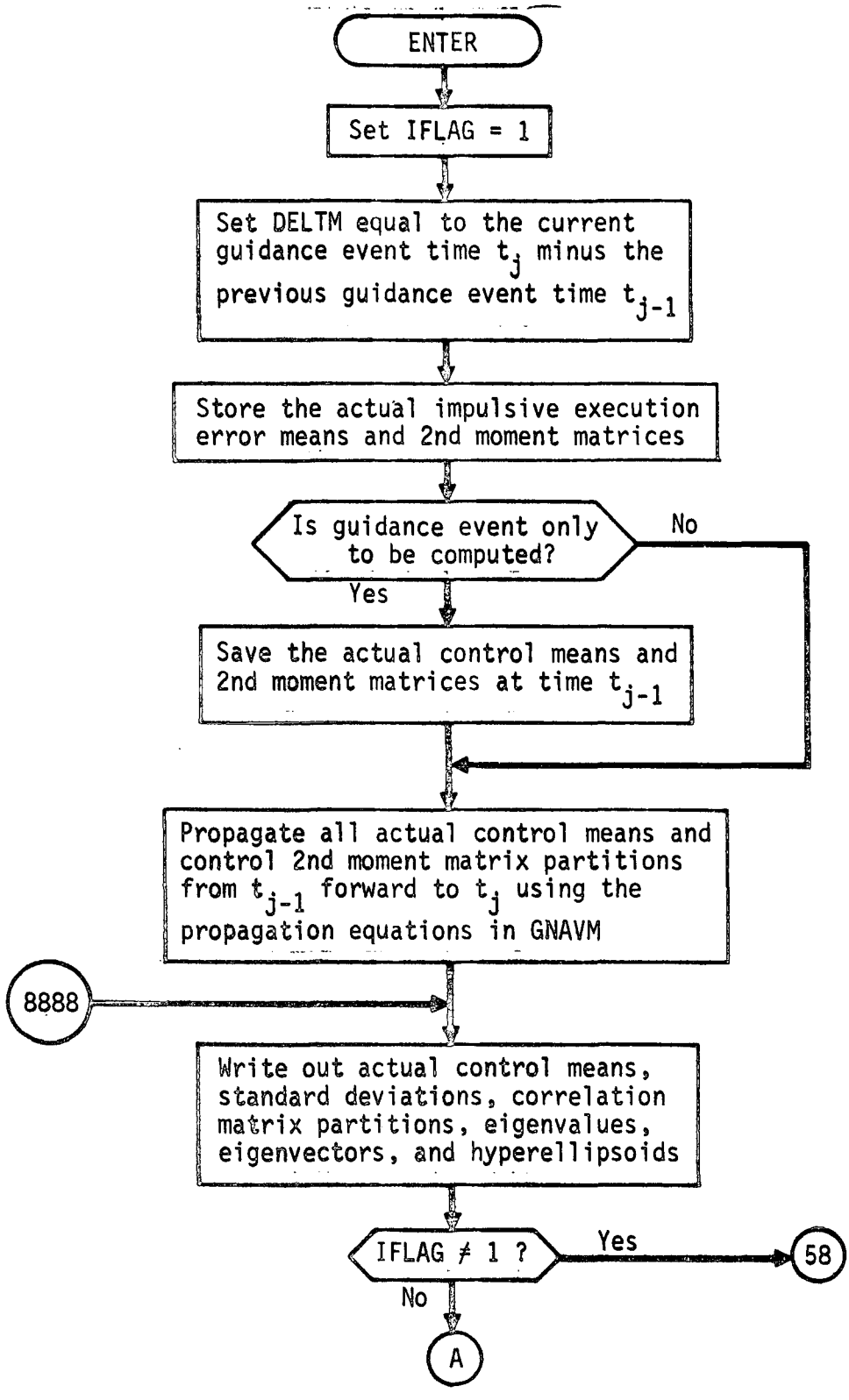
where $A = [0 \mid I]^T$. The actual deviation means are updated using the following equations:

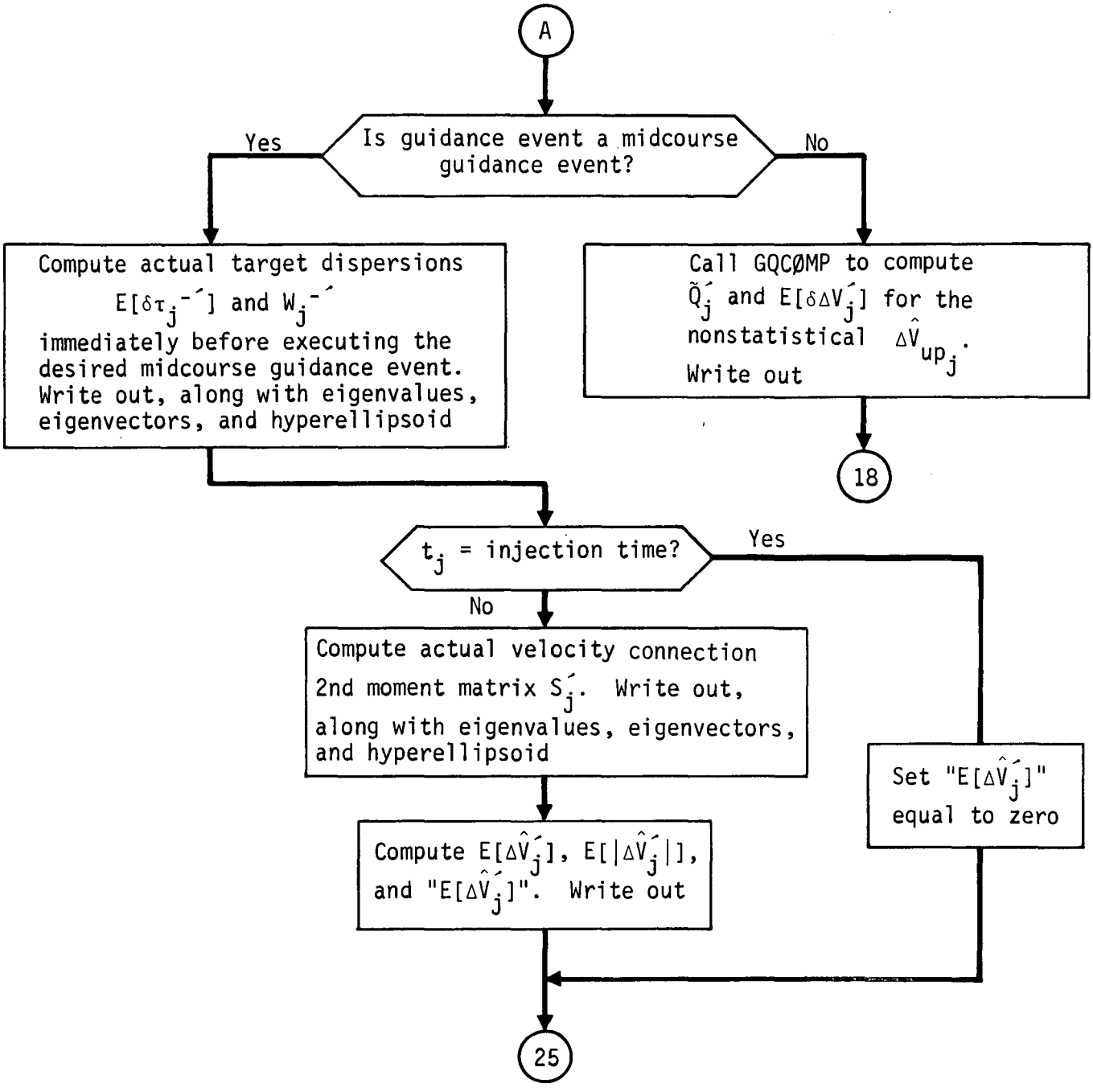
$$E\left[x_j'^{+}\right] = -E\left[\tilde{x}_j'^{+}\right] \tag{18}$$

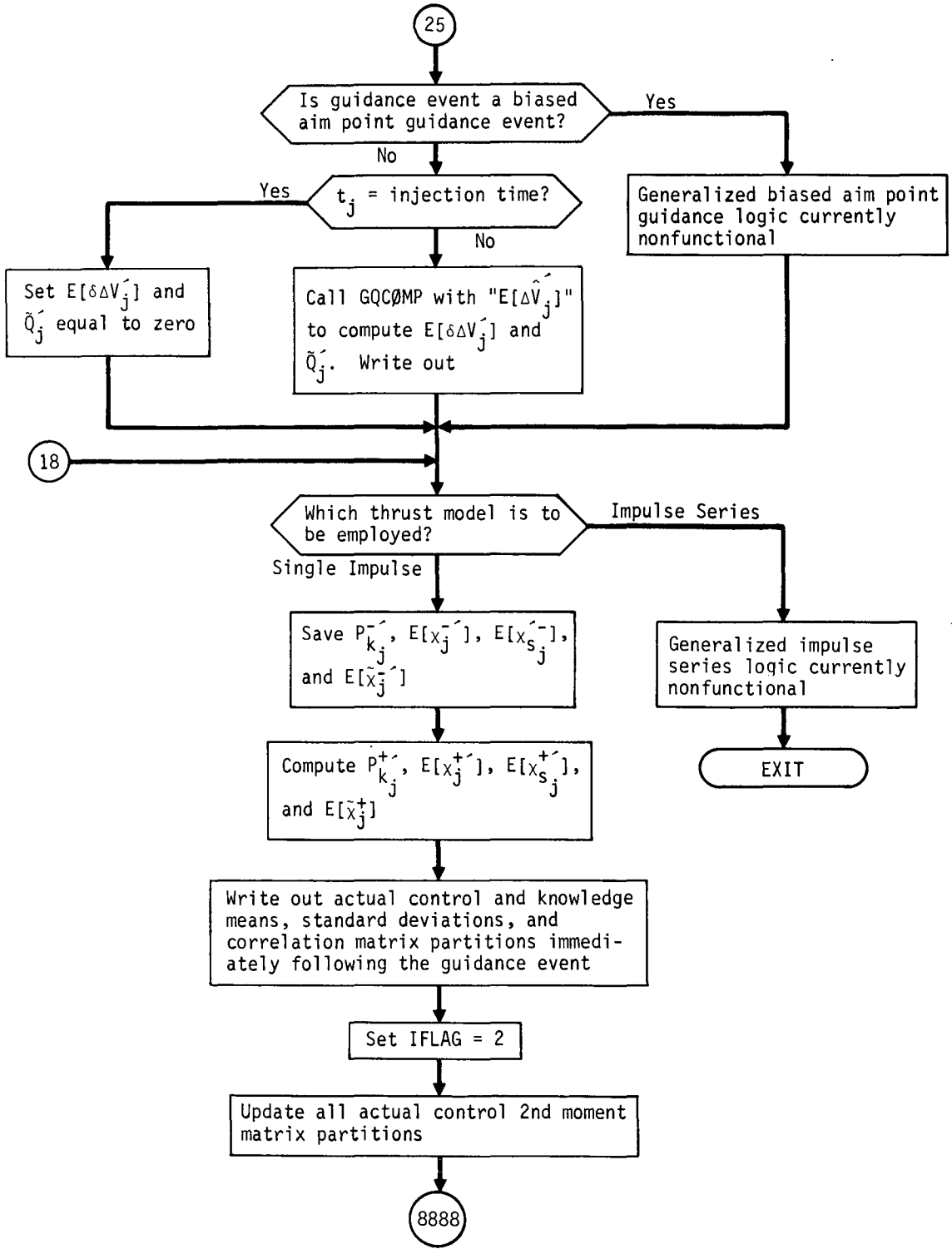
$$E\left[x_{s_j}'^{+}\right] = -E\left[\tilde{x}_{s_j}'^{+}\right] \tag{19}$$

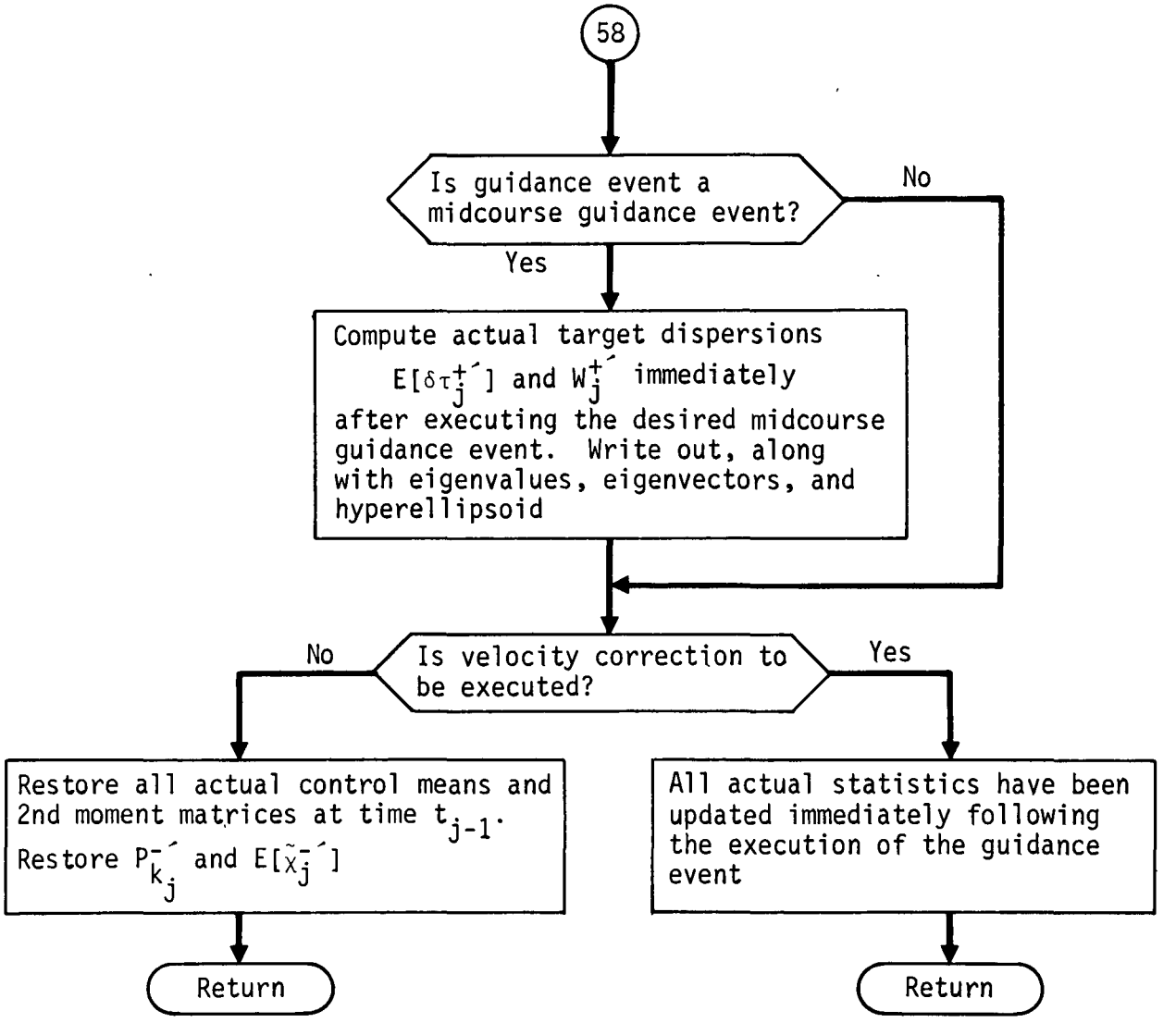
The entire set of actual control 2nd-moment matrix partitions is updated by equating them to the corresponding actual knowledge 2nd-moment matrix partitions at t_j^+.

GENGID Flow Chart




A
Is guidance event a midcourse guidance event?
Yes
No
Compute actual target dispersions $E[\delta\tau_j^{-\prime}]$ and $W_j^{-\prime}$ immediately before executing the desired midcourse guidance event. Write out, along with eigenvalues, eigenvectors, and hyperellipsoid
Call GQCØMP to compute $\tilde{Q}_j'$ and $E[\delta\Delta V_j']$ for the nonstatistical $\Delta\hat{V}_{up_j}$. Write out
18
t_j = injection time?
Yes
No
Compute actual velocity connection 2nd moment matrix S_j'. Write out, along with eigenvalues, eigenvectors, and hyperellipsoid
Set "$E[\Delta\hat{V}_j']$" equal to zero
Compute $E[\Delta\hat{V}_j']$, $E[|\Delta\hat{V}_j'|]$, and "$E[\Delta\hat{V}_j']$". Write out
25


25
Is guidance event a biased aim point guidance event?
Yes
No
Generalized biased aim point guidance logic currently nonfunctional
t_j = injection time?
Yes
No
Set $E[\delta\Delta V_j']$ and $\tilde{Q}_j'$ equal to zero
Call GQCØMP with "$E[\Delta\hat{V}_j']$" to compute $E[\delta\Delta V_j']$ and $\tilde{Q}_j'$. Write out
18
Which thrust model is to be employed?
Impulse Series
Single Impulse
Save $P_{k_j}^{-\prime}$, $E[x_j^{-\prime}]$, $E[x_{s_j}^{-\prime}]$, and $E[\tilde{x}_j^{-\prime}]$
Generalized impulse series logic currently nonfunctional
EXIT
Compute $P_{k_j}^{+\prime}$, $E[x_j^{+\prime}]$, $E[x_{s_j}^{+\prime}]$, and $E[\tilde{x}_j^{+}]$
Write out actual control and knowledge means, standard deviations, and correlation matrix partitions immediately following the guidance event
Set IFLAG = 2
Update all actual control 2nd moment matrix partitions
8888


58
Is guidance event a midcourse guidance event?
No
Yes
Compute actual target dispersions $E[\delta\tau_j^{+\prime}]$ and $W_j^{+\prime}$ immediately after executing the desired midcourse guidance event. Write out, along with eigenvalues, eigenvectors, and hyperellipsoid
No
Is velocity correction to be executed?
Yes
Restore all actual control means and 2nd moment matrices at time t_{j-1}.
Restore $P_{k_j}^{-\prime}$ and $E[\tilde{x}_j^{-\prime}]$
All actual statistics have been updated immediately following the execution of the guidance event
Return
Return

GHA Analysis

Subroutine GHA computes the Greenwich hour angle in degrees and days at some epoch T* referenced to 1950 January 1^d0^h. Epoch T* is computed from

$$T^* = J.D._0 + 2415020.0 - J.D._{REF}$$

where

$J.D._0$ = Julian date at launch time t_o referenced to 1900 January 0^d12^h.

$J.D._{REF}$ = Reference Julian date 2433282.5

= 1950 January 1^d0^h referenced to January 0^d12^h of the year 4713 B.C.

and 2415020.0 = 1900 January 0^d12^h referenced to January 0^d12^h of the year 4713 B.C.

Then T* is the Julian date at launch time t_o referenced to 1950 January 1^d0^h.

The Greenwich hour angle corresponding to T* is given by

$$GHA(T^*) = 100.0755426 + 0.985647346d + 2.9015 \times 10^{-13}\, d^2 + \omega t$$

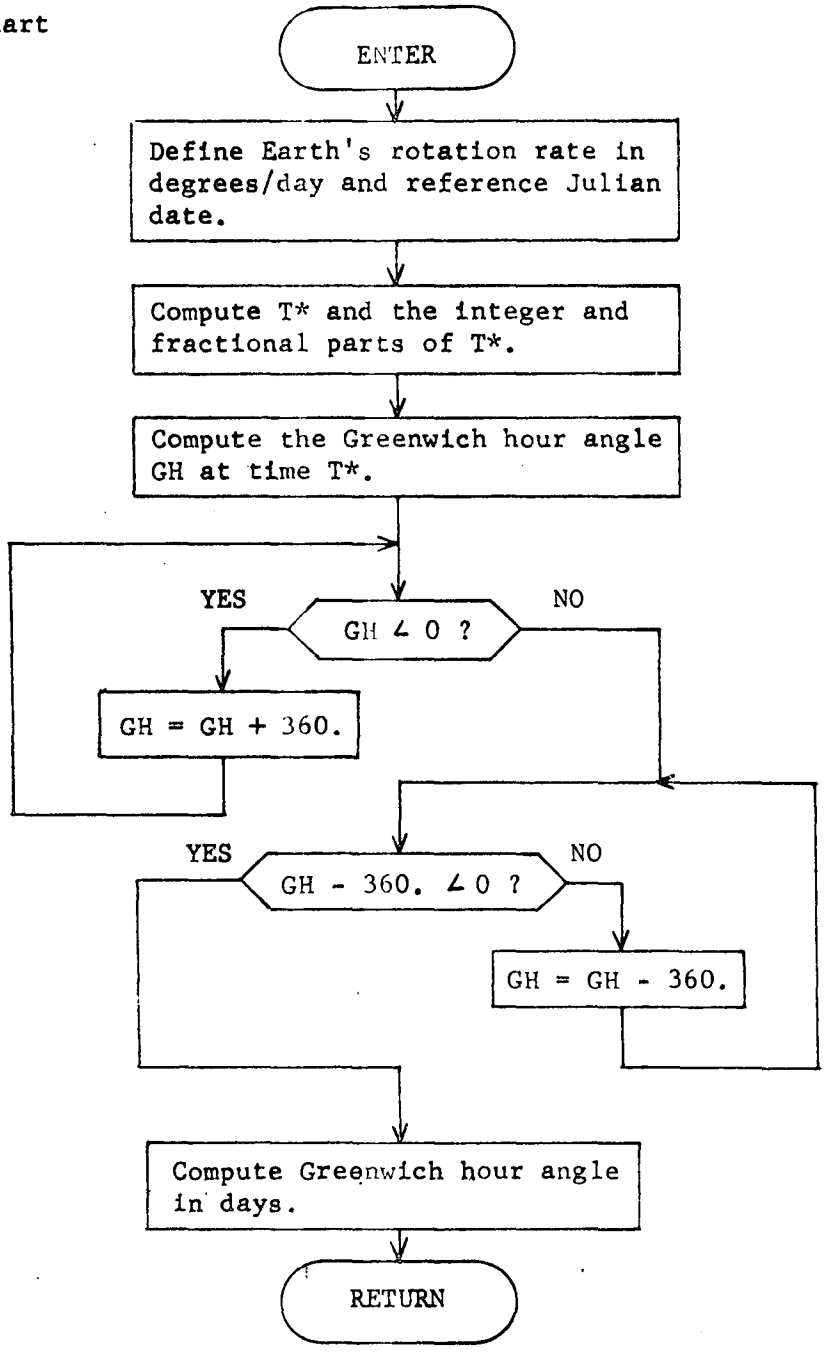
where $0 \leq GHA(T^*) < 360^\circ$.

and d = integer part of T*, t = fractional part of T*,

and ω = Earth's rotation rate is degrees/day.

The Greenwich hour angle in days is given by $\frac{GHA}{\omega}$.

GHA Flow Chart



GIDANS Analysis

GIDANS is an executive routine responsible for processing a guidance manuever for the computation of the velocity increment Δv to the execution of the correction.

Before entry to GIDANS, TRJTRY has computed the index of the current event (KUR) and has integrated the nominal trajectory to the time of the event. GIDANS now evaluates the KUR component of two integer arrays -- KTYP and KMXQ. The values of these flags determine the operation of GIDANS. The flag KTYP specifies the type of guidance event to be performed, while KMXQ prescribes the compute/execute mode to be used according to

KTYP = -1 Termination event
1 Targeting event
2 Retargeting event
3 Orbit insertion
4 Main probe propagation
5 Miniprobe targeting

KMXQ = 1 Compute $\Delta \vec{V}$ only
2 Execute Δv only
3 Compute and execute $\Delta \vec{V}$
4 Compute but execute Δv later

GIDANS first checks for a termination event. If the current index prescribes such an event, the flag KWIT is set to 1 and a return is made to the main program NØMNAL.

In prepration for a normal guidance event, GIDANS calls VMP with the current spacecraft heliocentric state and a time increment of zero to restore the F and V arrays providing the current geometry of spacecraft and planets. If the current event is an execute-only mode, the transfer is made to the execution section of GIDANS for the addition of the preset velocity increment.

Otherwise GIDANS interrogates KTYP for the type of maneuver to be computed. For a targeting event, subroutine TARGET is called directly for the computation of the Δv necessary to satisfy input target conditions. After calling TARGET, the F and V arrays are restored as indicated above.

A retargeting event is defined as a targeting event that requires computation of a new zero iterate. Thus a retargeting event is an event in which the current nominal state when integrated forward would miss the target conditions badly. Such an event would be the broken-plane correction. For this event TRJTRY stores the current position (and possibly the target position) in the ZDAT array. It then calls ZERIT for the computation of the massless-planet's initial velocity consistent with the target conditions. It then operates identically to the targeting event.

The third guidance maneuver is the insertion event. GIDANS calls INSERS for the computation of the velocity increment $\Delta\underset{\sim}{v}$ and the time interval Δt before it is to be executed.

The main-probe propagation event involves storing the current spacecraft state, propagating the main probe to an appropriate stopping condition while printing a time history, and restoring the original state in preparation for the next event. It is carried out in a single call to the subroutine MPPRØP. Upon return to GIDANS, the F and V arrays are restored as indicated above.

The miniprobe targeting event, although somewhat complicated, is completely executed by the single subroutine TPRTRG. The current bus state is first stored. Next the miniprobe release controls are calculated to apply at the current time to target three miniprobes respectively to three target sites characterized by imput values of declination and right ascension. Using the minimum-miss release controls, each miniprobe is then propagated from release to a stopping condition while a time history is concurrently printed. Finally, the original bus state is restored. On returning to GIDANS, the F and V arrays are restored as usual.

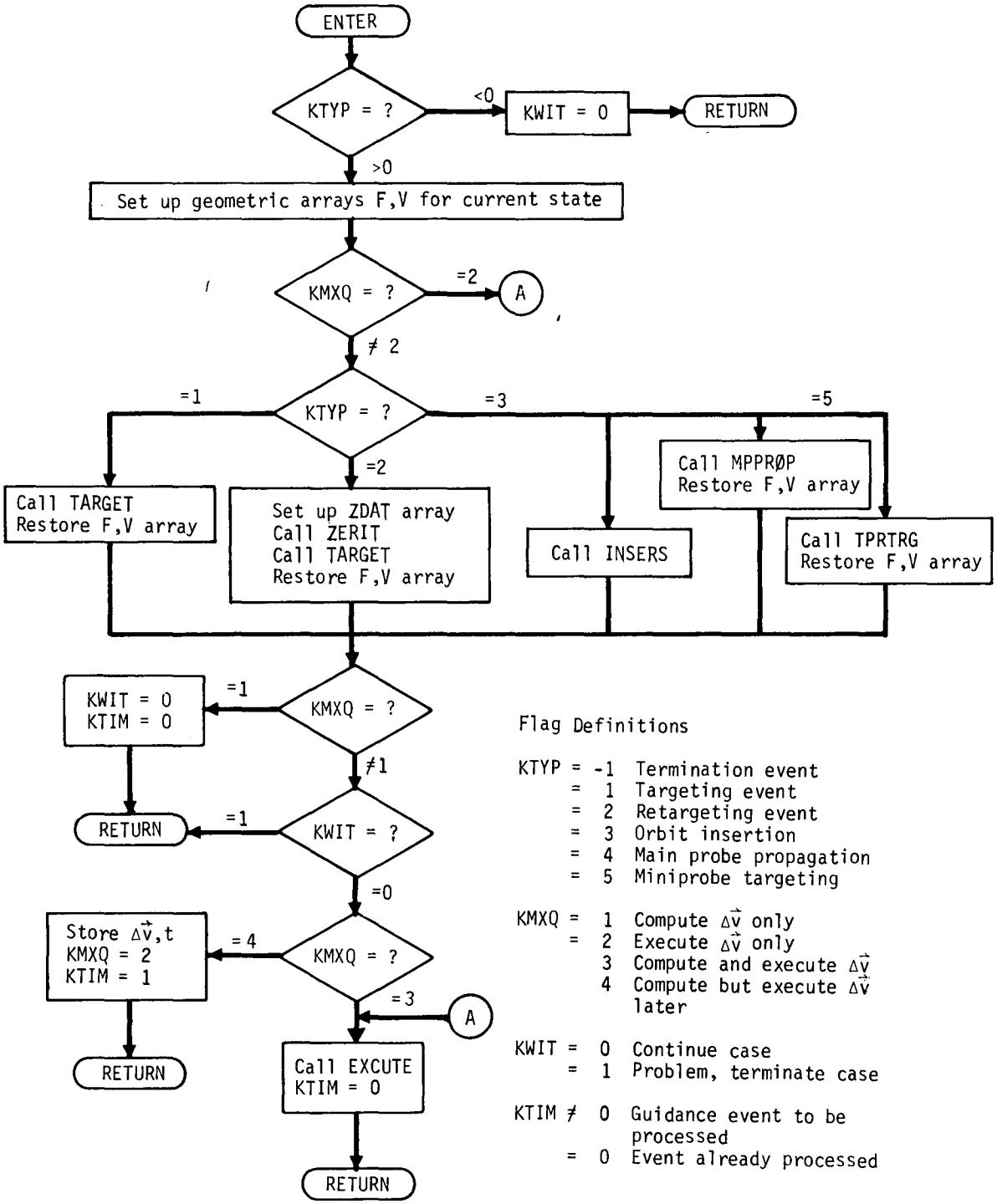
The three subroutines TARGET, INSERS, and TPRTRG signal trouble to GIDANS via the flag KWIT. If problems are encountered in their execution, e.g., failure to converge in TARGET or TPRTRG or the impossibility of insertion in INSERS, KWIT is set to 1. Otherwise KWIT = 0. On return to NØMNAL, if KWIT = 1 the current case is terminated while if KWIT = 0 it is continued.

If the current event is a compute-only mode, TRJTRY now sets KWIT = 0 (so that the program will continue regardless of whether the correction computations were successful) and returns to NØMNAL. However if the current event failed (KWIT = 1) and was to be executed (KMXQ $\neq$ 1) GIDANS consideres this a fatal error for the current case and returns with KWIT = 1.

If the compute/execute mode is compute-execute later (KMXQ = 4) as is the insertion event, GIDANS now sets up for the subsequent execute-only event. The $\Delta\underset{\sim}{v}$ computed is stored in the DELV array, the time of the execution is computed $\left(t_{ex} = t_k + \Delta t\right)$ and stored in the TIMG array, and the KMXQ flag is set to a 2 (execute-only). The return is then made to NØMINAL.

For an event to be executed at the current time (KMXQ = 2,3), GIDANS now calls EXCUTE for the completion of that task.

It should be noted that for all events that are completed at this time, the KUR components of the KTIM array are set equal to 0 so they are no longer considered in determining the next event in TRJTRY. Only in the case of KMXQ = 4 is the KTIM flag nonzero on exit from GIDANS.



Flag Definitions

KTYP = -1 Termination event
= 1 Targeting event
= 2 Retargeting event
= 3 Orbit insertion
= 4 Main probe propagation
= 5 Miniprobe targeting

KMXQ = 1 Compute $\Delta\vec{v}$ only
= 2 Execute $\Delta\vec{v}$ only
3 Compute and execute $\Delta\vec{v}$
4 Compute but execute $\Delta\vec{v}$ later

KWIT = 0 Continue case
= 1 Problem, terminate case

KTIM ≠ 0 Guidance event to be processed
= 0 Event already processed

GNAVM Analysis

Subroutine GNAVM propagates and updates (at a measurement) both assumed (or filter) covariance matrix partitions and actual 2nd moment matrix partitions. The equations programmed in GNAVM are independent of the filter algorithm employed to generate gain matrices.

The covariance and 2nd moment matrix partitions manipulated by GNAVM are defined as follows:

$$
\begin{aligned}
P &= E[\tilde{x}\,\tilde{x}^T] & P_s &= E\left[\tilde{x}_s\,\tilde{x}_s^T\right] \\
C_{xx_s} &= E\left[\tilde{x}\,\tilde{x}_s^T\right] & C_{x_s u} &= E[\tilde{x}_s\,\tilde{u}^T] \\
C_{xu} &= E[\tilde{x}\,\tilde{u}^T] & C_{x_s v} &= E[\tilde{x}_s\,\tilde{v}^T] \\
C_{xv} &= E[\tilde{x}\,\tilde{v}^T] & C_{x_s w} &= E[\tilde{x}_s\,\tilde{w}^T] \quad . \\
C_{xw} &= E[\tilde{x}\,\tilde{w}^T]
\end{aligned}
\tag{1}
$$

The following matrix partitions are used in GNAVM, but are not changed in GNAVM:

$$
\begin{aligned}
C_{uv} &= E[\tilde{u}\,\tilde{v}^T] \\
C_{uw} &= E[\tilde{u}\,\tilde{w}^T] \\
C_{vw} &= E[\tilde{v}\,\tilde{w}^T] \\
U &= E[\tilde{u}\,\tilde{u}^T] \\
V &= E[\tilde{v}\,\tilde{v}^T] \\
W &= E[\tilde{w}\,\tilde{w}^T] \quad .
\end{aligned}
\tag{2}
$$

In these definitions $\tilde{x}$, $\tilde{x}_s$, $\tilde{u}$, $\tilde{v}$, and $\tilde{w}$ represent, respectively, the estimation errors in position/velocity state, solve-for parameters, dynamic consider parameters, measurement consider parameters, and ignore parameters. Ignore parameters, of course, are not defined when assumed (or filter) covariance matrix partitions are being propagated or updated. Furthermore, the assumed C_{uv} has been set to zero.

The equations used to propagate covariances or 2nd moment matrices from time t_k to t_{k+1} are summarized:

$$P^-_{k+1} = \left(\Phi P^+_k + \theta_{xx_s} C^{+T}_{xx_s{}_k} + \theta_{xu} C^{+T}_{xu_k} + \theta_{xw} C^{+T}_{xw_k}\right)\Phi^T$$

$$+ C^-_{xx_s{}_{k+1}} \theta^T_{xx_s} + C^-_{xu_{k+1}} \theta^T_{xu} + C^-_{xw_{k+1}} \theta^T_{xw} + Q_{k+1} \qquad (3)$$

$$C^-_{xx_s{}_{k+1}} = \Phi C^+_{xx_s{}_k} + \theta_{xx_s} P^+_{s_k} + \theta_{xu} C^{+T}_{x_s u_k} + \theta_{xw} C^{+T}_{x_s w_k} \qquad (4)$$

$$C^-_{xu_{k+1}} = \Phi C^+_{xu_k} + \theta_{xx_s} C^+_{x_s u_k} + \theta_{xu} U_o + \theta_{xw} C^T_{uw_o} \qquad (5)$$

$$C^-_{xv_{k+1}} = \Phi C^+_{xv_k} + \theta_{xx_s} C^+_{x_s v_k} + \theta_{xu} C_{uv_o} + \theta_{xw} C^T_{vw_o} \qquad (6)$$

$$C^-_{xw_{k+1}} = \Phi C^+_{xw_k} + \theta_{xx_s} C^+_{x_s w_k} + \theta_{xu} C_{uw_o} + \theta_{xw} W_o \qquad (7)$$

$$P^-_{s_{k+1}} = P^+_{s_k} \qquad (8)$$

$$C^-_{x_s u_{k+1}} = C^+_{x_s u_k} \qquad (9)$$

$$C^{-}_{x_s v_{k+1}} = C^{+}_{x_s v_k} \tag{10}$$

$$C^{-}_{x_s w_{k+1}} = C^{+}_{x_s w_k} \quad . \tag{11}$$

In these equations $(\ )^{-}$ indicates immediately prior to processing a measurement; $(\ )^{+}$, immediately after. The state transition matrices over the interval $[t_k, t_{k+1}]$ are indicated by Φ, θ_{xx_s}, θ_{xu}, and θ_{xw}. The dynamic noise covariance or 2nd moment matrix is denoted by Q_{k+1}.

Before covariance (or 2nd moment) matrix partitions can be updated at a measurement, the measurement residual covariance (or 2nd moment) matrix, defined by

$$J_{k+1} = E\left[\varepsilon_{k+1}\ \varepsilon^{T}_{k+1}\right] \tag{12}$$

must be computed. The required equations are summarized

$$J_{k+1} = HA_{k+1} + MB_{k+1} + GD_{k+1} + LE_{k+1} + NF_{k+1} + R_{k+1} \tag{13}$$

$$A_{k+1} = P^{-}_{k+1} H^{T} + C^{-}_{xx_s{}_{k+1}} M^{T} + C^{-}_{xu_{k+1}} G^{T} + C^{-}_{xv_{k+1}} L^{T} + C^{-}_{xw_{k+1}} N^{T} \tag{14}$$

$$B_{k+1} = P^{-}_{s_{k+1}} M^{T} + C^{-T}_{xx_s{}_{k+1}} H^{T} + C^{-}_{x_s u_{k+1}} G^{T} + C^{-}_{x_s v_{k+1}} L^{T} + C^{-}_{x_s w_{k+1}} N^{T} \tag{15}$$

$$D_{k+1} = C^{-T}_{xu_{k+1}} H^{T} + C^{-T}_{x_s u_{k+1}} M^{T} + U_o G^{T} + C^{-}_{uw_o} N^{T} + C^{-}_{uv_o} L^{T} \tag{16}$$

$$E_{k+1} = C^{-T}_{xv_{k+1}} H^{T} + C^{-T}_{x_s v_{k+1}} M^{T} + C^{-}_{vw_o} N^{T} + V_o L^{T} + C^{-T}_{uv_o} G^{T} \tag{17}$$

$$F_{k+1} = W_o N^T + C^{-T}_{xw_{k+1}} H^T + C^{-T}_{x_s w_{k+1}} M^T + C^{-T}_{vw_o} L^T + C^{-T}_{uw_o} G^T \quad . \qquad (18)$$

In these equations H, M, G, L, and N represent observation matrix partitions, and R_{k+1} represents the measurement noise covariance (or 2nd moment) matrix.

Gain matrices K_{k+1} and S_{k+1} are also required before covariance (or 2nd moment) matrix partitions can be updated. These are not computed in GNAVM but are obtained by calling either subroutine GAIN1 or GAIN2, depending on which recursive estimation algorithm is desired.

With J_{k+1}, K_{k+1}, and S_{k+1} available, the following equations are used in the updating process:

$$P^+_{k+1} = P^-_{k+1} - K_{k+1} A^T - A K^T_{k+1} + K_{k+1} J_{k+1} K^T_{k+1} \qquad (19)$$

$$C^+_{xx_{s_{k+1}}} = C^-_{xx_{s_{k+1}}} - K_{k+1} B^T - A S^T_{k+1} + K_{k+1} J_{k+1} S^T_{k+1} \qquad (20)$$

$$C^+_{xu_{k+1}} = C^-_{xu_{k+1}} - K_{k+1} D^T \qquad (21)$$

$$C^+_{xv_{k+1}} = C^-_{xv_{k+1}} - K_{k+1} D^T \qquad (22)$$

$$C^+_{xw_{k+1}} = C^-_{xw_{k+1}} - K_{k+1} F^T \qquad (23)$$

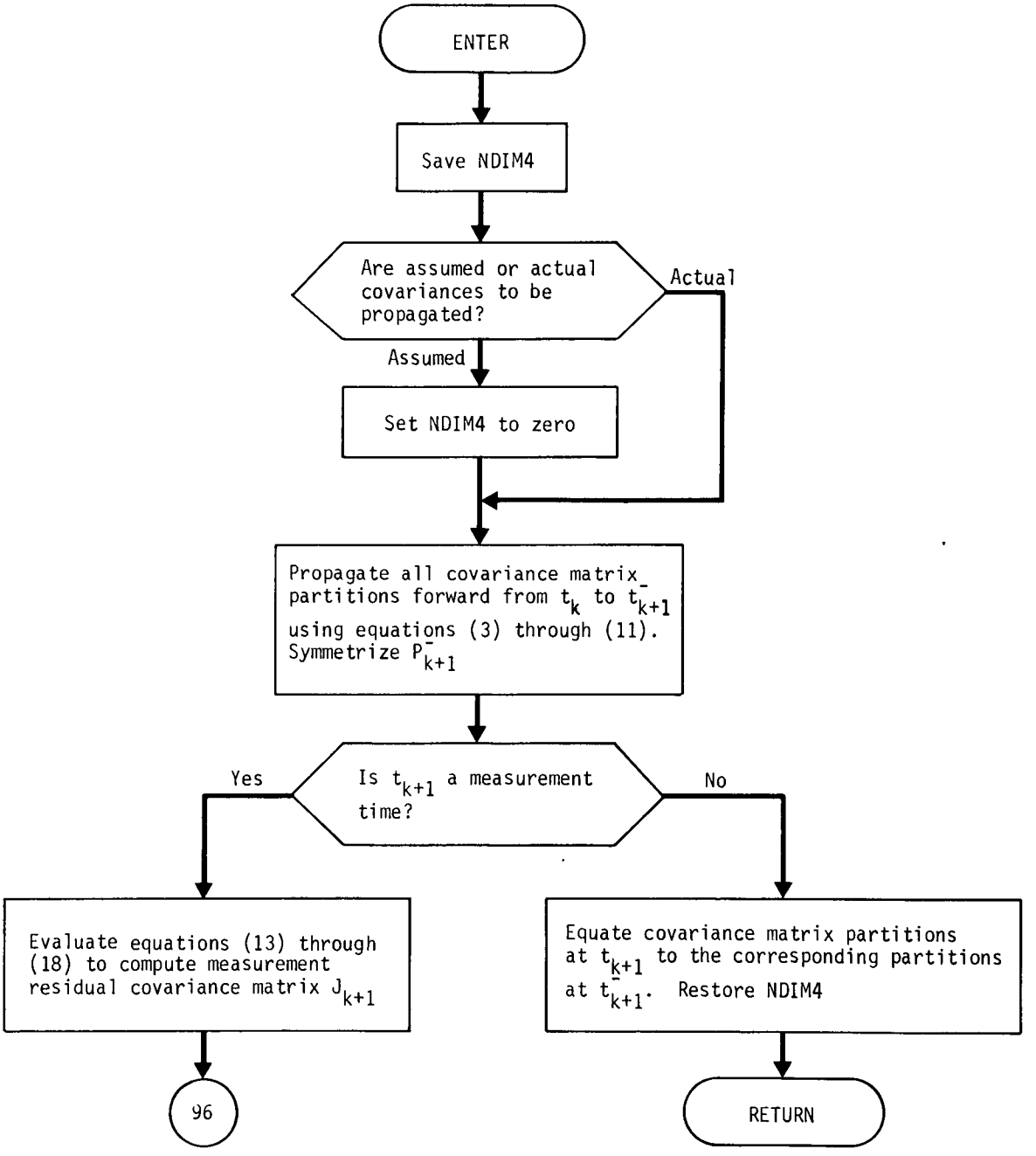
$$P^+_{s_{k+1}} = P^-_{s_{k+1}} - S_{k+1} B^T - B S^T_{k+1} + S_{k+1} J_{k+1} S^T_{k+1} \qquad (24)$$

$$C^+_{x_s u_{k+1}} = C^-_{x_s u_{k+1}} - S_{k+1} D^T \qquad (25)$$

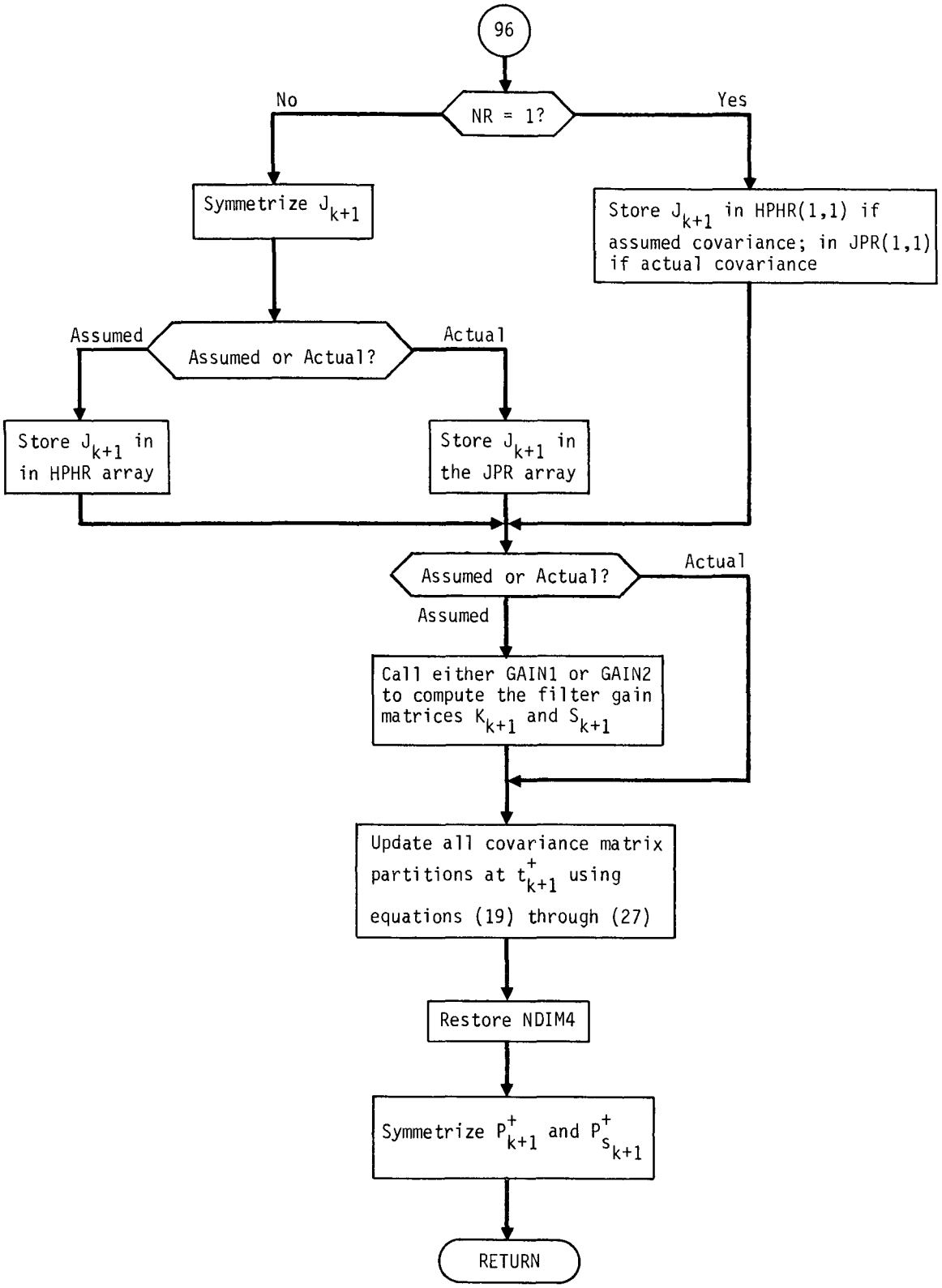
$$C^{+}_{x_s v_{k+1}} = C^{-}_{x_s v_{k+1}} - S_{k+1} E^T \tag{26}$$

$$C^{+}_{x_s w_{k+1}} = C^{-}_{x_s w_{k+1}} - S_{k+1} F^T \quad . \tag{27}$$

It should be noted that propagation equations (3) through (11) are also used to propagate both assumed control covariance and actual 2nd moment matrix partitions over the time interval separating two successive guidance events. The update equations, of course, are not used in this situation.



GNAVM Flow Chart


96
NR = 1?
No
Yes
Symmetrize J_{k+1}
Store J_{k+1} in HPHR(1,1) if assumed covariance; in JPR(1,1) if actual covariance
Assumed or Actual?
Assumed
Actual
Store J_{k+1} in in HPHR array
Store J_{k+1} in the JPR array
Assumed or Actual?
Actual
Assumed
Call either GAIN1 or GAIN2 to compute the filter gain matrices K_{k+1} and S_{k+1}
Update all covariance matrix partitions at t^+_{k+1} using equations (19) through (27)
Restore NDIM4
Symmetrize P^+_{k+1} and $P^+_{s_{k+1}}$
RETURN

GQCØMP Analysis

Subroutine GQCØMP computes the actual execution error mean and 2nd moment matrix for use in the generalized covariance analysis of a guidance event. The actual execution error $\delta\Delta V'_j$ is assumed to have the form

$$\delta\Delta V'_j = k' \, \Delta\hat{V}'_j + s' \frac{\Delta\hat{V}'_j}{|\Delta\hat{V}'_j|} + \delta\Delta V'_{pointing} \tag{1}$$

where k' denotes the actual proportionality error; s', the actual resolution error; $\delta\Delta V'_{pointing}$, the actual pointing error; and $\Delta\hat{V}'_j$, the actual commanded velocity correction.

The means of the three ecliptic components of $\delta\Delta V'_j$ are given as:

$$E[\delta\Delta V'_x] = \left(\bar{k}' + \frac{\bar{s}'}{\rho'}\right) \Delta\hat{V}'_x + \frac{\rho'\Delta\hat{V}'_y \, \overline{\delta\alpha}' + \Delta\hat{V}'_x \, \Delta\hat{V}'_z \, \overline{\delta\beta}'}{\mu'} \tag{2}$$

$$E[\delta\Delta V'_y] = \left(\bar{k}' + \frac{\bar{s}'}{\rho'}\right) \Delta\hat{V}'_y + \frac{\Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta}' - \rho'\Delta\hat{V}'_x \, \overline{\delta\alpha}'}{\mu'} \tag{3}$$

$$E[\delta\Delta V'_z] = \left(\bar{k}' + \frac{\bar{s}'}{\rho'}\right) \Delta\hat{V}'_z - \mu' \, \overline{\delta\beta}' \tag{4}$$

where $\rho' = |\Delta\hat{V}'|$, $\mu' = [\Delta\hat{V}'^2_x + \Delta\hat{V}'^2_y]^{\frac{1}{2}}$, and $\delta\alpha'$ and $\delta\beta'$ are the actual pointing angle errors, and both E( ) and $\bar{(\ )}$ indicate mean values.

The actual execution error 2nd moment matrix is defined by

$$\tilde{Q}'_j = E\left[\delta\Delta V'_j \, \delta\Delta V'^T_j\right] \quad . \tag{5}$$

the elements Q'_{ik} of matrix Q'_j are given as:

$$\tilde{Q}'_{11} = \xi' \, \Delta\hat{V}'^2_x + \frac{1}{\mu'^2}\left(\rho'^2\Delta\hat{V}'^2_y \, \overline{\delta\alpha' \, \delta\alpha'} + \Delta\hat{V}'^2_x \, \Delta\hat{V}'^2_z \, \overline{\delta\beta' \, \delta\beta'} + \right.$$

$$\left. 2\rho' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\alpha}' \, \overline{\delta\beta}'\right) + \frac{2\Delta\hat{V}'_x}{\mu'} \zeta' \left(\rho\Delta\hat{V}'_y \, \overline{\delta\alpha}' + \Delta\hat{V}'_x \, \Delta\hat{V}'_z \, \overline{\delta\beta}'\right) \tag{5}$$

$$\tilde{Q}'_{22} = \xi' \, \Delta\hat{V}'^2_y + \frac{1}{\mu'^2}\left(\Delta\hat{V}'^2_y \, \Delta\hat{V}'^2_z \, \overline{\delta\beta' \, \delta\beta'} + \rho'^2 \, \Delta\hat{V}'^2_x \, \overline{\delta\alpha' \, \delta\alpha'} - 2\rho' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\alpha'} \, \overline{\delta\beta'}\right) + \frac{2\Delta\hat{V}'_y}{\mu'} \zeta' \left(\Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta'} - \rho' \, \Delta\hat{V}'_x \, \overline{\delta\alpha'}\right) \quad (6)$$

$$\tilde{Q}'_{33} = \xi' \, \Delta\hat{V}'^2_z + \mu'^2 \, \overline{\delta\beta' \, \delta\beta'} - 2\Delta\hat{V}'_z \, \mu' \, \zeta' \, \overline{\delta\beta'} \quad (7)$$

$$\tilde{Q}'_{12} = \tilde{Q}'_{21} = \xi' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y + \frac{\zeta'}{\mu'}\left[2\Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta'} - \rho'\left(\Delta\hat{V}'^2_x - \Delta\hat{V}'^2_y\right)\overline{\delta\alpha'}\right] + \frac{1}{\mu'^2}\left[-\rho'^2 \, \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \overline{\delta\alpha' \, \delta\alpha'} + \rho' \, \Delta\hat{V}'_z \left(\Delta\hat{V}'^2_y - \Delta\hat{V}'^2_x\right) \overline{\delta\alpha'} \, \overline{\delta\beta'} + \Delta\hat{V}'_x \, \Delta\hat{V}'_y \, \Delta V'^2_z \, \overline{\delta\beta' \, \delta\beta'}\right] \quad (8)$$

$$\tilde{Q}'_{13} = \tilde{Q}'_{31} = \xi' \, \Delta\hat{V}'_x \, \Delta\hat{V}'_z + \zeta'\left[\frac{\Delta\hat{V}'_z}{\mu'}\left(\rho'\Delta\hat{V}'_y \, \overline{\delta\alpha'} + \Delta\hat{V}'_x \, \Delta\hat{V}'_z \, \overline{\delta\beta'}\right) - \mu' \, \Delta\hat{V}'_x \, \overline{\delta\beta'}\right] - \rho'\Delta\hat{V}'_y \, \overline{\delta\alpha'} \, \overline{\delta\beta'} - \Delta\hat{V}'_x \, \Delta\hat{V}'_z \, \overline{\delta\beta' \, \delta\beta'} \quad (9)$$

$$\tilde{Q}'_{23} = \tilde{Q}'_{32} = \xi' \, \Delta\hat{V}'_y \, \Delta\hat{V}'_z + \zeta'\left[\frac{\Delta\hat{V}'_z}{\mu'}\left(\Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta'} - \rho'\Delta\hat{V}'_x \, \overline{\delta\alpha'}\right) - \mu' \, \Delta\hat{V}'_y \, \overline{\delta\beta'}\right] + \rho'\Delta\hat{V}'_x \, \overline{\delta\alpha'} \, \overline{\delta\beta'} - \Delta\hat{V}'_y \, \Delta\hat{V}'_z \, \overline{\delta\beta' \, \delta\beta'} \quad (10)$$

where

$$\xi' = \overline{k' \, k'} + \frac{2}{\rho'} \, \overline{k'} \, \overline{s'} + \frac{\overline{s' \, s'}}{\rho'^2} \quad (11)$$

and

$$\zeta' = \overline{k'} + \frac{\overline{s'}}{\rho'} \quad (12)$$

GUID Analysis

Subroutine GUID is used in the error analysis mode to compute the same quantities which subroutine GUIS computes in the simulation mode. Subroutine GUID differs from GUIS in that instead of calling NTMS and VARSIM as does GUIS, subroutine GUID calls NTM and VARADA. In addition, the state transition and variation matrices computed in GUID are referenced to the targeted nominal since the most recent nominal is not defined for the error analysis mode. These differences entail only minor logic differences in the flow chart for GUID, and for this reason no GUID flow chart is presented. See subroutine GUIS analysis and flow chart for further details.

GUIDM Analysis

Subroutine GUIDM is the executive guidance subroutine in the error analysis program. In addition to controlling the computational flow for all types of guidance events, GUIDM also performs many of the required guidance computations itself.

Before considering each type of guidance event, the treatment of a general guidance event will be discussed. Let t_j be the time at which the guidance event occurs. Before any guidance event can be executed the targeted nominal state $\bar{X}_j^-$, knowledge covariance $P_{K_j}^-$, and control covariance $P_{C_j}^-$ must all be available, where $(\ )^-$ indicates values immediately before the event. The first two quantities are available prior to entering GUIDM. However, GUIDM controls the propagation of the control covariance over the interval $[t_{j-1}, t_j]$, where t_{j-1} denotes the time of the previous guidance event.

The next step in the treatment of a general guidance event is concerned with the computation of the commanded velocity correction and the execution error covariance. In the error analysis program a non-statistical velocity correction is computed whenever the nominal target conditions are changed; otherwise, only a statistical velocity correction can be computed. The commanded velocity correction $\hat{\Delta V}_j$ is then used to compute the execution error covariance matrix $\tilde{Q}_j$. A summary of the execution error model and the equations used to compute $\tilde{Q}_j$ can be found in the subroutine QCØMP analysis section.

The last step is concerned with the updating of required quantities prior to returning to the basic cycle. An assumption underlying the modeled guidance process is that the targeted nominal is always updated by the commanded velocity correction. In the error analysis program only the non-statistical component is used to perform the state update and is indicated by the variable $\hat{\Delta V}_{UP_j}$. Thus, the targeted nominal state immediately following the guidance event is given by

$$\bar{X}_j^+ = \bar{X}_j^- + \begin{bmatrix} 0 \\ \hat{\Delta V}_{UP_j} \end{bmatrix} .$$

The knowledge covariance is updated using the equation

$$P_{K_j}^{+} = P_{K_j}^{-} + \begin{bmatrix} \underline{0} & \vdots & \underline{0} \\ 0 & \vdots & \tilde{\underline{Q}}_j \end{bmatrix}$$

if an impulsive thrust model is assumed. If the thrust is modeled as a series of impulses, then an effective execution error covariance $\tilde{Q}_{eff}$ is computed and the knowledge covariance is updated using the equation

$$P_{K_j}^{+} = P_{K_j}^{-} + \tilde{Q}_{eff} .$$

In either case the control covariance is updated simply by setting

$$P_{c_j}^{+} = P_{K_j}^{+} .$$

This equation is a direct consequence of the assumption that the targeted nominal state is always updated at a guidance event.

A "compute only" option is available in GUIDM in which all of the $(\ )^{+}$ quantities will still be computed and printed. However, the state and all covariances are then reset to their former $(\ )^{-}$ values prior to returning to the basic cycle.

Each specific type of guidance event involves the computation of other quantities not discussed above. These will be covered in the following discussion of specific guidance events.

1. Midcourse and Biased Aimpoint Guidance

   Linear midcourse guidance policies have form

$$\Delta\hat{V}_{N_j} = \Gamma_j \, \delta\hat{X}_j$$

where the subscript N indicates that this is the velocity correction required to null out deviations from the nominal target state. This notation is required to differentiate between this type of velocity correction and velocity corrections required to achieve an altered target

state. Linear midcourse guidance policies are discussed in more detail in the subroutine GUIS analysis section.

Subroutine GUIDM calls GUID to compute the guidance matrix, Γ_j, and the target condition covariance immediately prior to the guidance event, W_j, and then uses Γ_j to compute the velocity correction covariance S_j, which is defined as

$$S_j = E\left[\Delta\hat{V}_{N_j}\ \Delta\hat{V}_{N_j}^T\right],$$

and is given by the equation

$$S_j = \Gamma_j\ (\bar{P_{c_j}} - \bar{P_{K_j}})\ \Gamma_j^T .$$

This equation assumes that an optimal estimation algorithm is employed in the navigation process, since the derivation of this equation requires the orthogonality of the estimate and the estimation error.

In the error analysis program $\Delta\hat{V}_{N_j}$ is never available since no estimates $\delta\hat{X}_j$ are ever generated. Only the ensemble statistics of $\delta\hat{X}_j$ are available which means only a statistical or effective velocity correction "$E\left[\Delta\hat{V}_{N_j}\right]$" can be computed. In the STEAP error analysis program this effective velocity correction is assumed to have form

$$"E\left[\Delta\hat{V}_{N_j}\right]" = \rho_j\ \frac{\alpha_j}{|\alpha_j|} .$$

The magnitude ρ_j is given by the Hoffman-Young approximation

$$\rho_j = \sqrt{\frac{2A}{\pi}}\left(1 + \frac{B(\pi-2)}{A^2\sqrt{5.4}}\right)$$

where

$$A = \text{trace } S_j = \lambda_1 + \lambda_2 + \lambda_3,$$

$$B = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3,$$

and λ_1, λ_2, λ_3 are the eigenvalues of S_j. The direction of the effective velocity correction is assumed to coincide with the eigenvector corresponding to the maximum eigenvalue of S_j. This eigenvector is denoted by α_j. An alternate model assumes the direction coincides with the vector $(\lambda_1, \lambda_2, \lambda_3)$.

If planetary quarantive constraints must be satisfied at a midcourse correction, GUIDM calls BIAIM to compute the new aimpoint μ_j and the (non-statistical) bias velocity correction $\widehat{\Delta V}_{B_j}$. All computations in BIAIM are based on linear guidance theory. However, an option is available in GUIDM to recompute $\Delta\hat{V}_{B_j}$, but not μ_j, using nonlinear techniques. This option is recommended if a biased aimpoint guidance event occurs at t_j = injection time. It should also be noted that $\tilde{Q}_j$ is set to zero if t_j = injection time since it is assumed that the injection covariance does not change for small changes in injection velocity.

After the updated control covariance $P_{c_j}^+$ has been computed, the target condition covariance matrix W_j^+ following the guidance correction is computed using the equation

$$W_j^+ = \eta_j \, P_{c_j}^+ \, \eta_j^T$$

where variation matrix η_j has been previously computed in subroutine GUID.

2. Re-targeting

In the error analysis (and simulation) program a re-targeting event is defined to be the computation of a velocity correction $\Delta\hat{V}_{RT}$ required to achieve a new set of target conditions using nonlinear techniques. Since the original targeted nominal will be used as the zero-th iterate in the re-targeting process, the new target conditions must be close enough to the original nominal target condition to ensure a covergent process.

It should be noted that after a re-targeting event the new target conditions are henceforth treated as the nominal target conditions.

3. Orbital insertion

An orbital insertion event is divided into a decision event and an execution event. At a decision event the orbital insertion velocity correction $\Delta\hat{V}_{\oslash I}$ and the time interval Δt separating decision and execution are computed based on the targeted nominal state at t_j. The relevant equations can be found in the subroutine CØPINS analysis section for co-planar orbital insertion; in NØPINS, for non-planar orbital insertion. Before returning to the basic cycle, GUIDM schedules the orbital insertion execution event to occur at $t_j + \Delta t$ and re-orders the necessary event arrays accordingly.

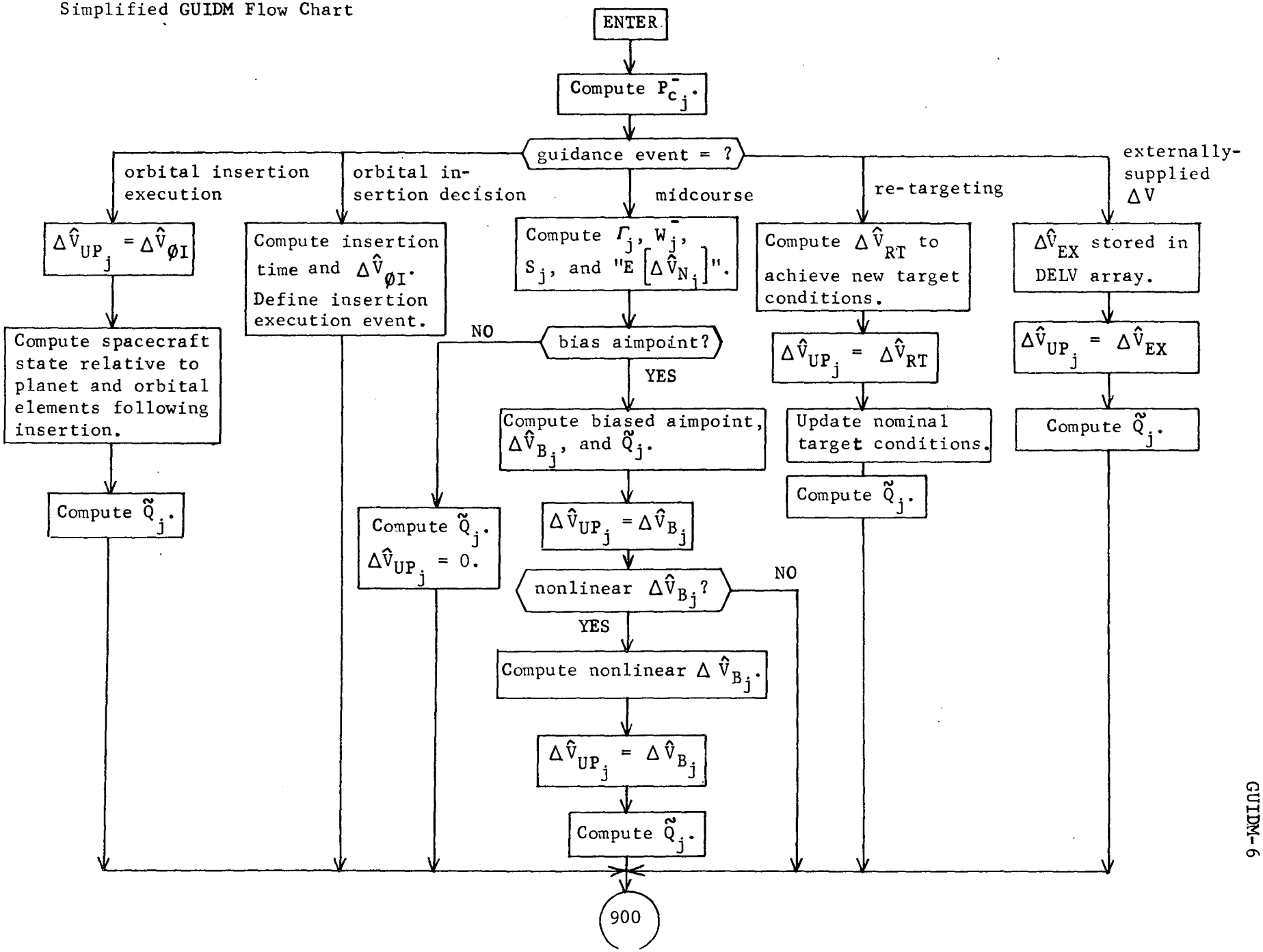
At an orbital insertion execution event the targeted nominal state is updated using the previously computed $\Delta\hat{V}_{\oslash I}$. In addition, the planeto-centric equatorial components of $\Delta\hat{V}_{\oslash I}$ and the nominal spacecraft cartesian and orbital element state following the insertion maneuver are computed.

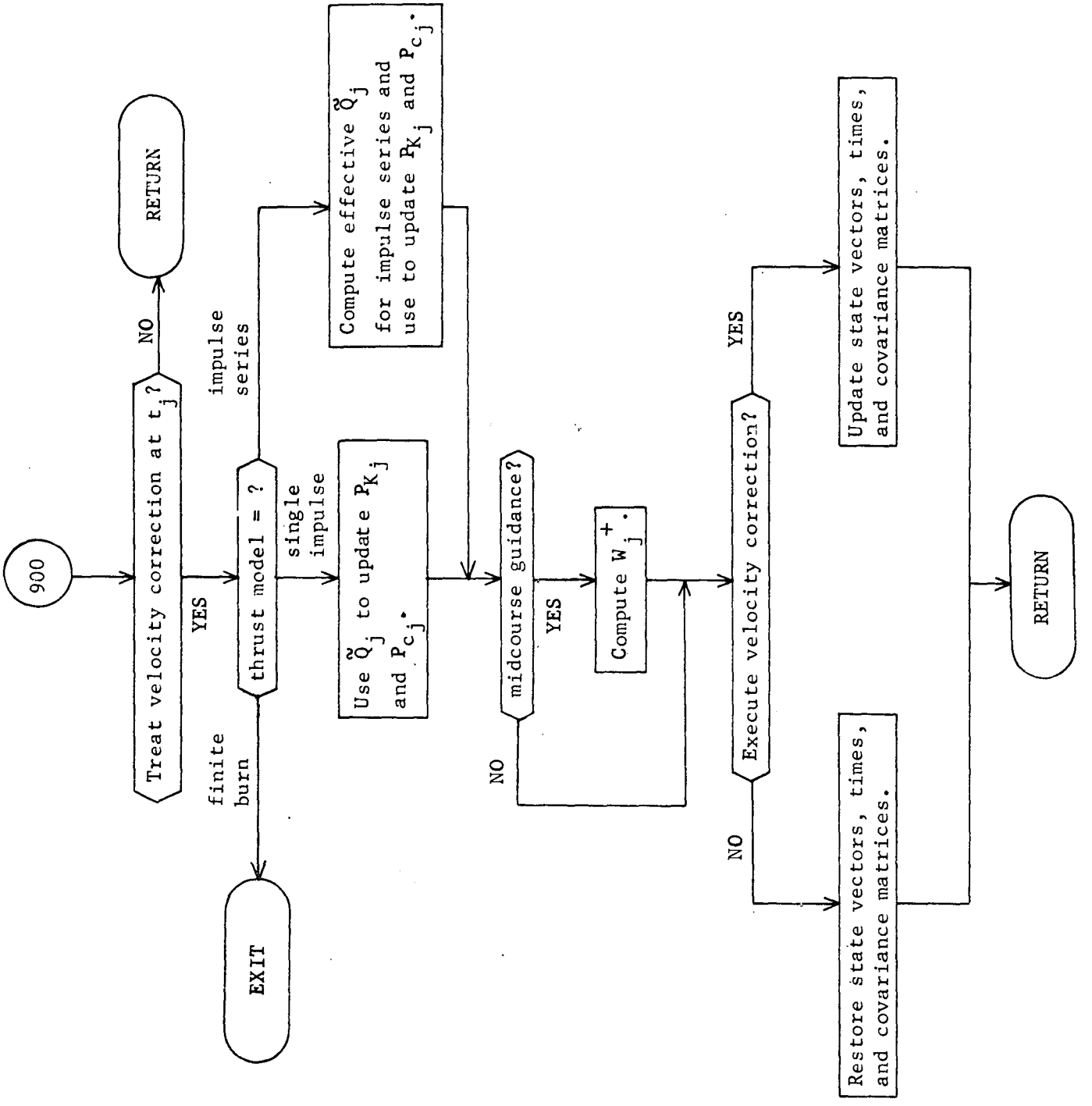
4. Externally-supplied velocity correction

At this type of guidance event the targeted nominal state is simply updated using the externally-supplied velocity correction $\Delta\hat{V}_{EX}$.

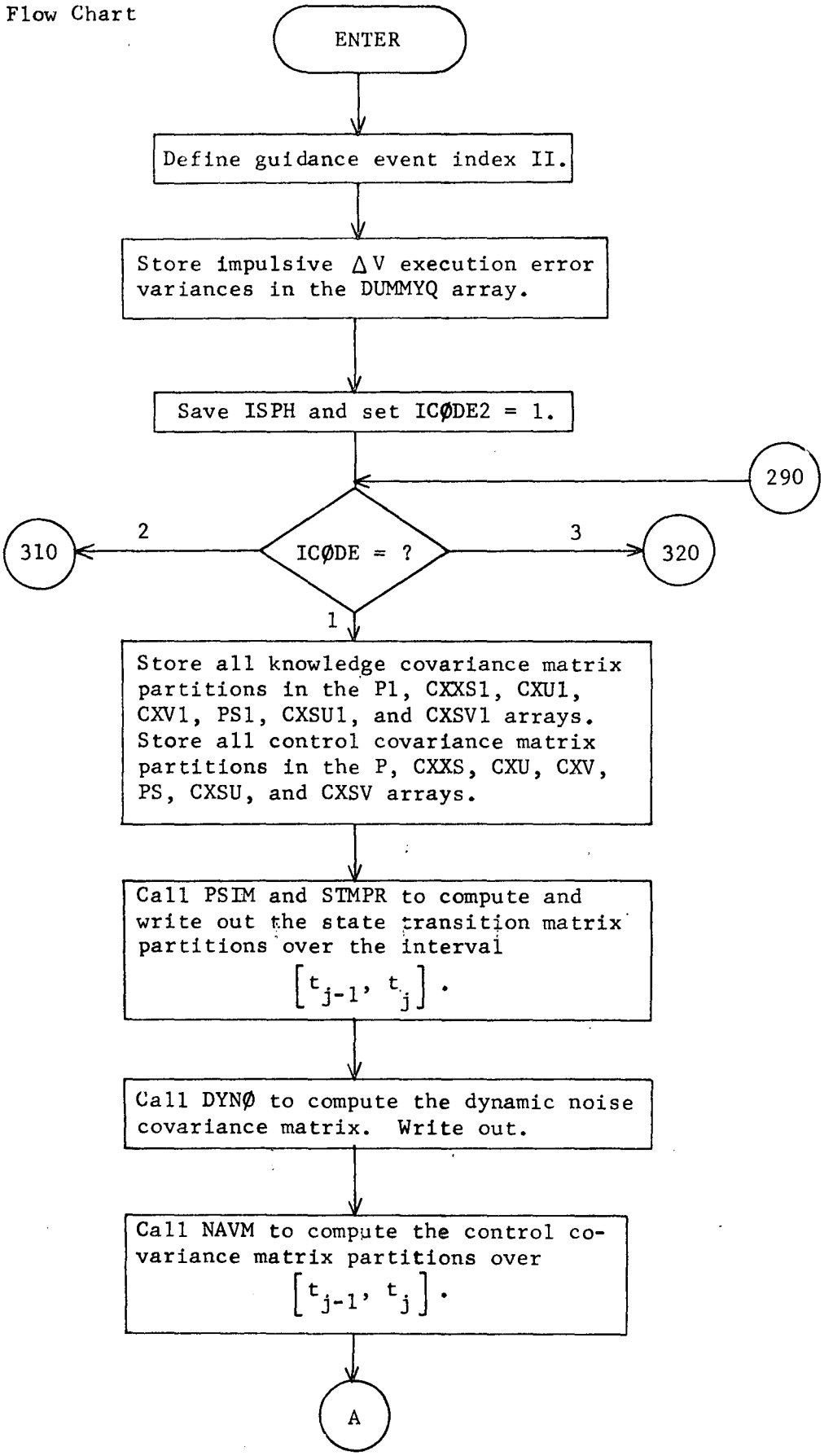
Because of the complexity of the GUIDM flow chart, a simplified flow chart depicting the main elements of the GUIDM structure precedes the complete GUIDM flow chart.

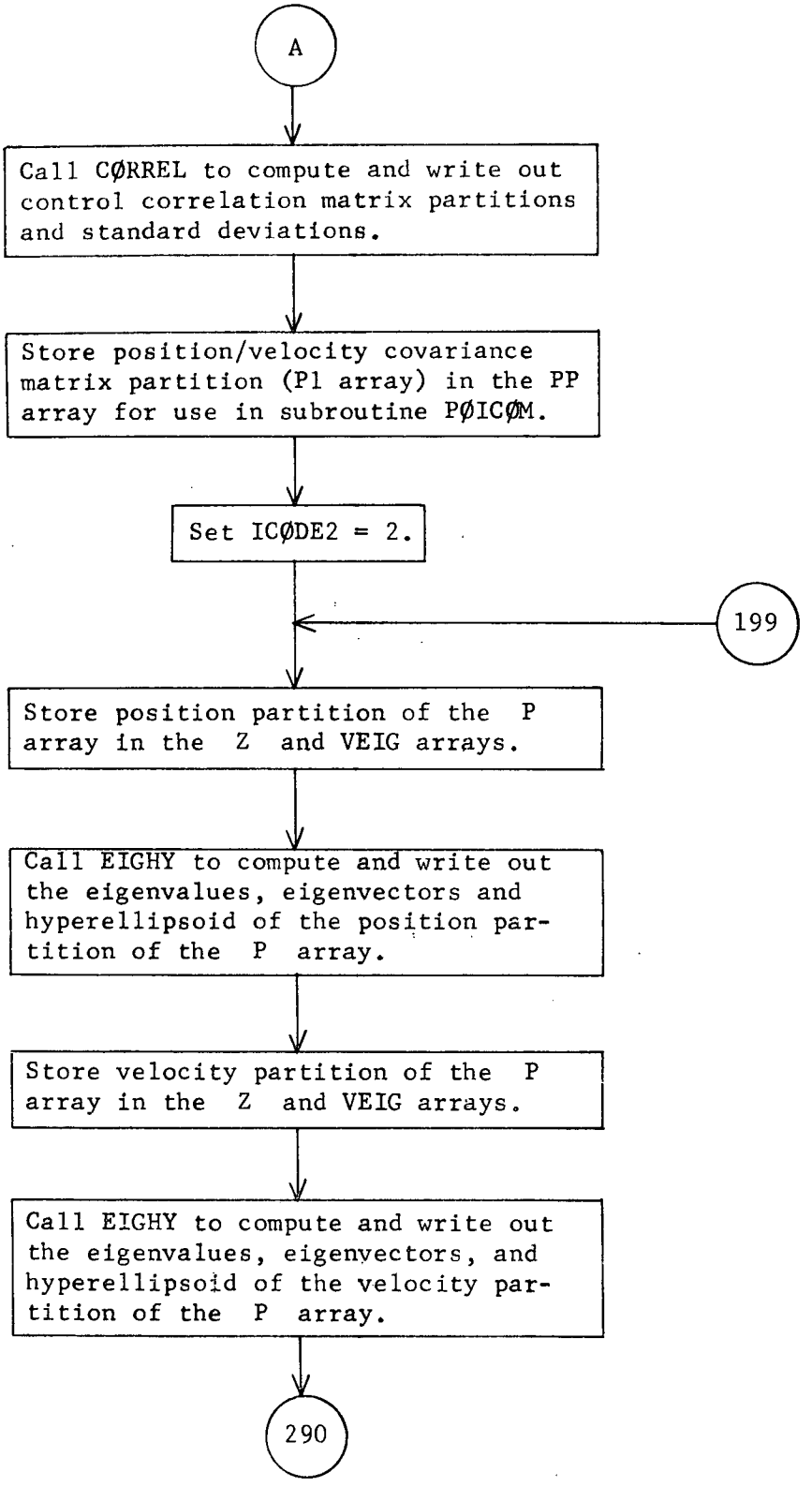
Simplified GUIDM Flow Chart

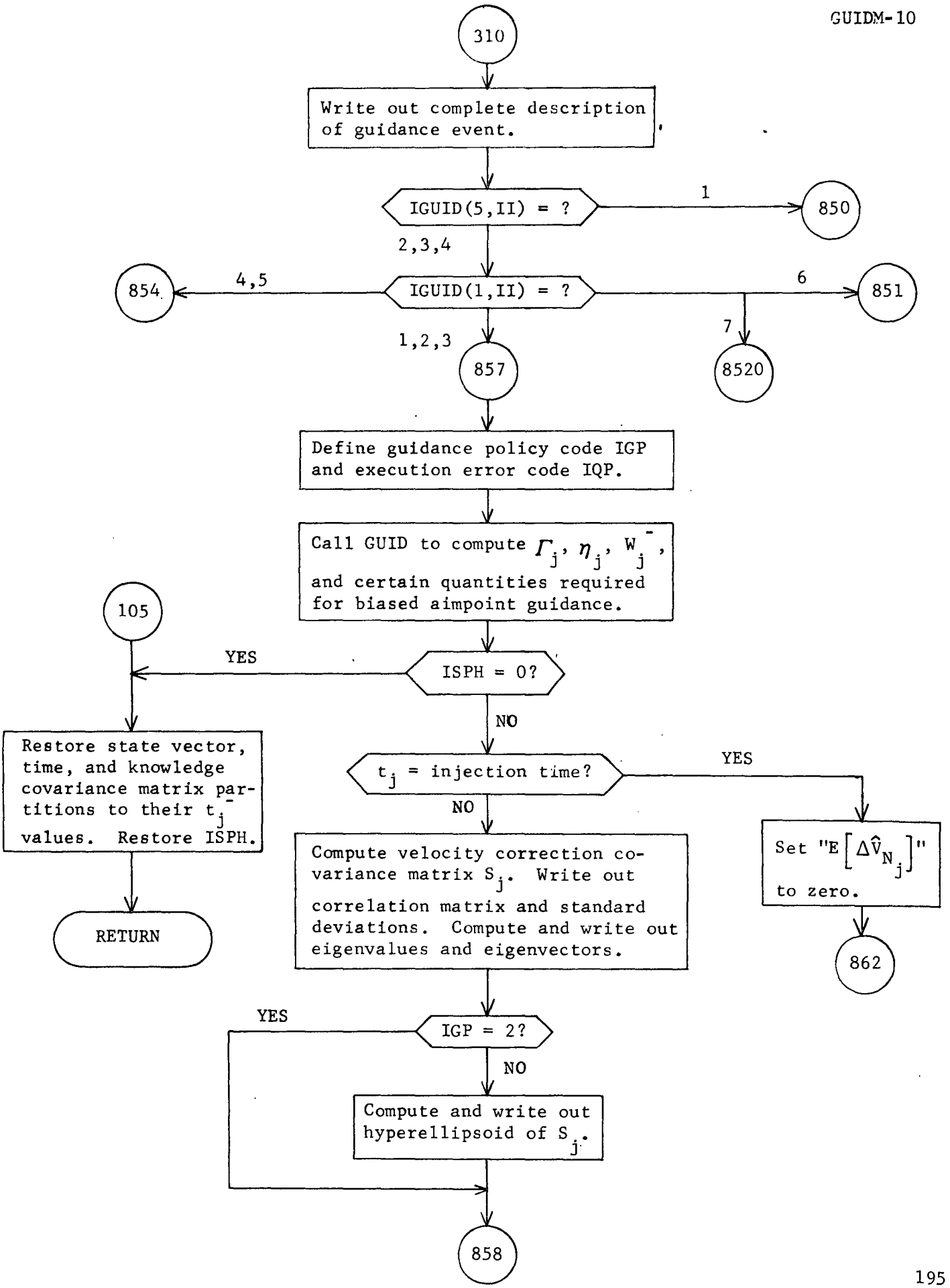


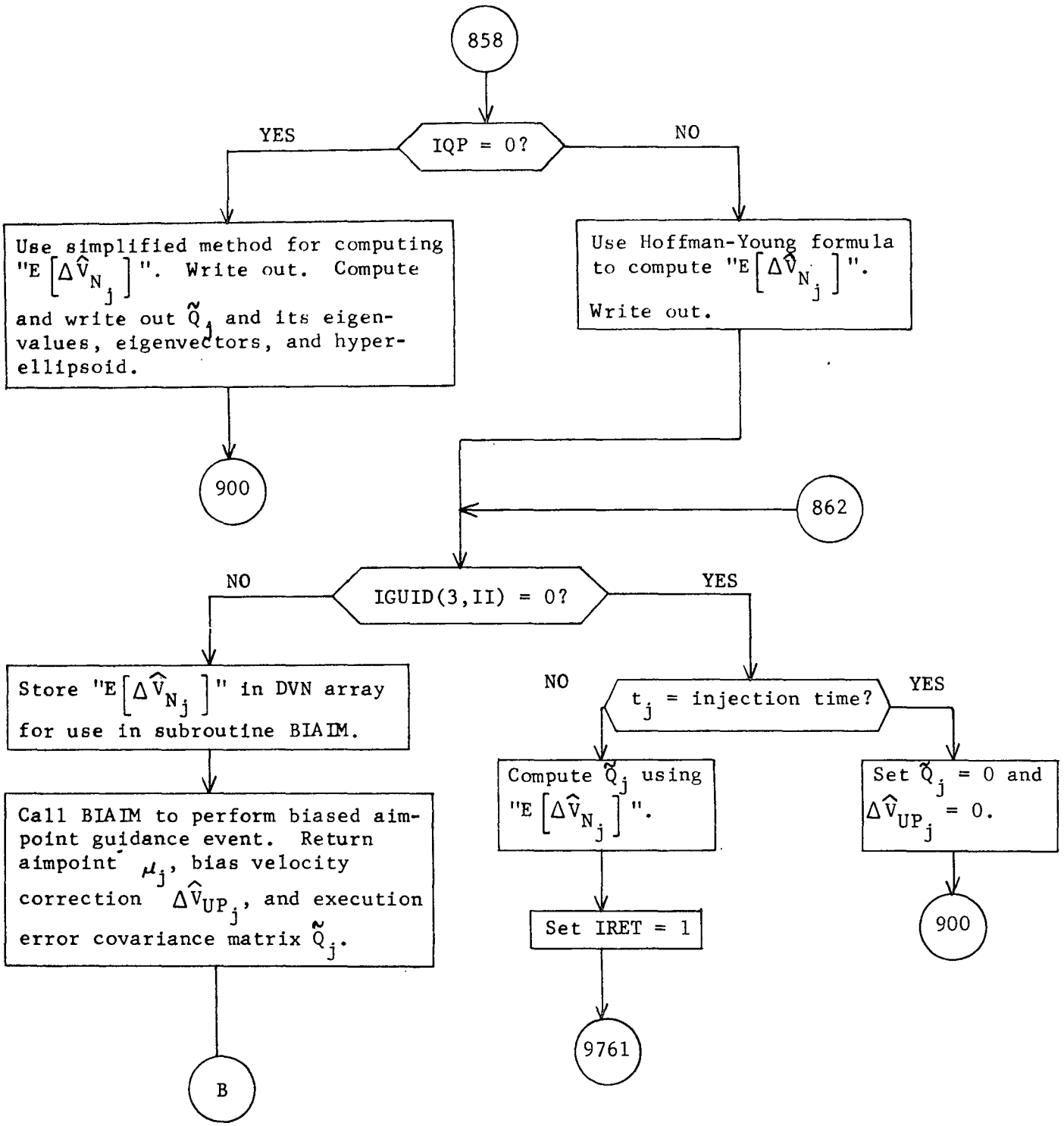

900
Treat velocity correction at t_j?
NO
RETURN
YES
thrust model = ?
finite burn
EXIT
single impulse
Use $\tilde{Q}_j$ to update P_{K_j} and P_{c_j}.
impulse series
Compute effective $\tilde{Q}_j$ for impulse series and use to update P_{K_j} and P_{c_j}.
midcourse guidance?
YES
Compute W_j^+.
NO
Execute velocity correction?
YES
Update state vectors, times, and covariance matrices.
NO
Restore state vectors, times, and covariance matrices.
RETURN

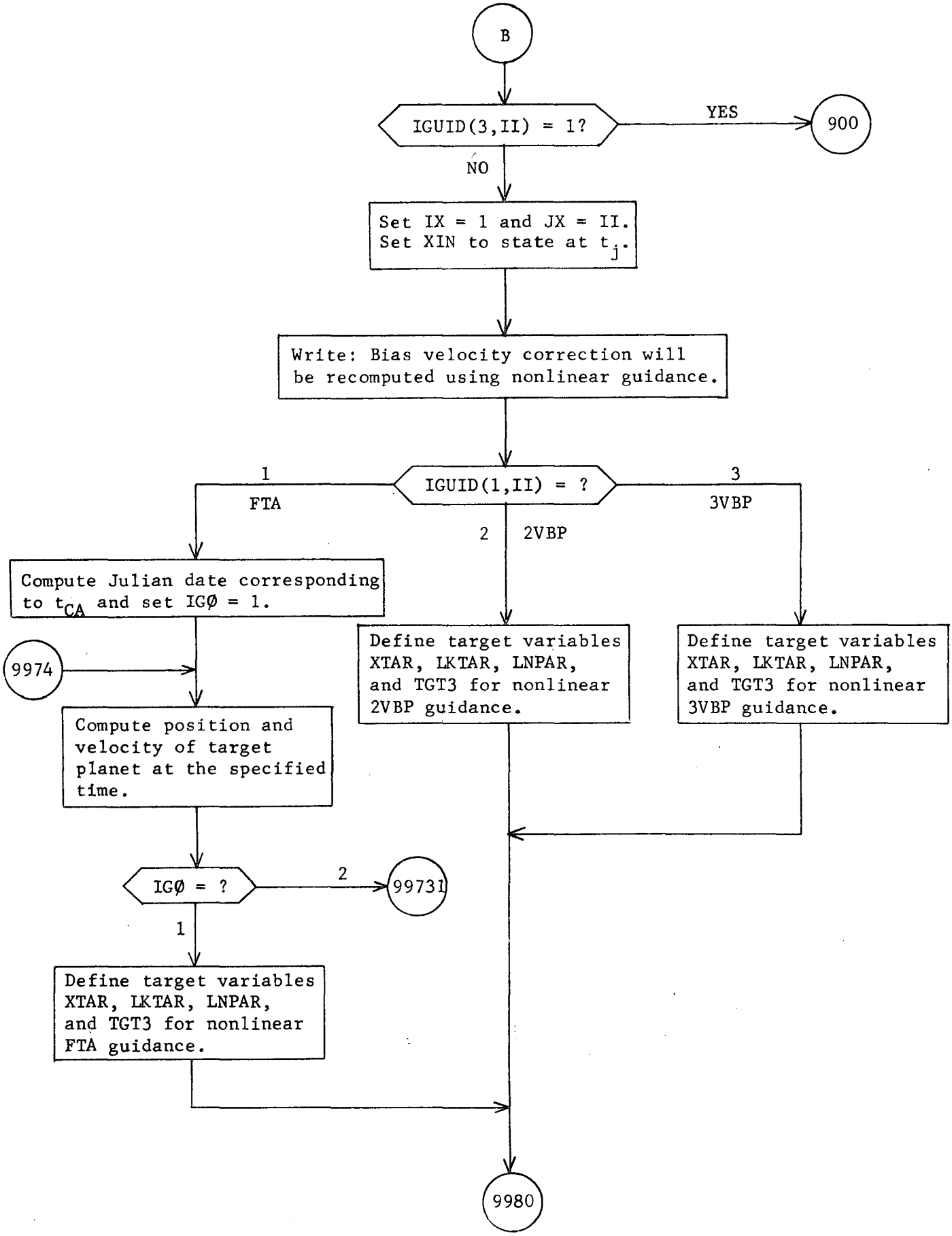
GUIDM Flow Chart

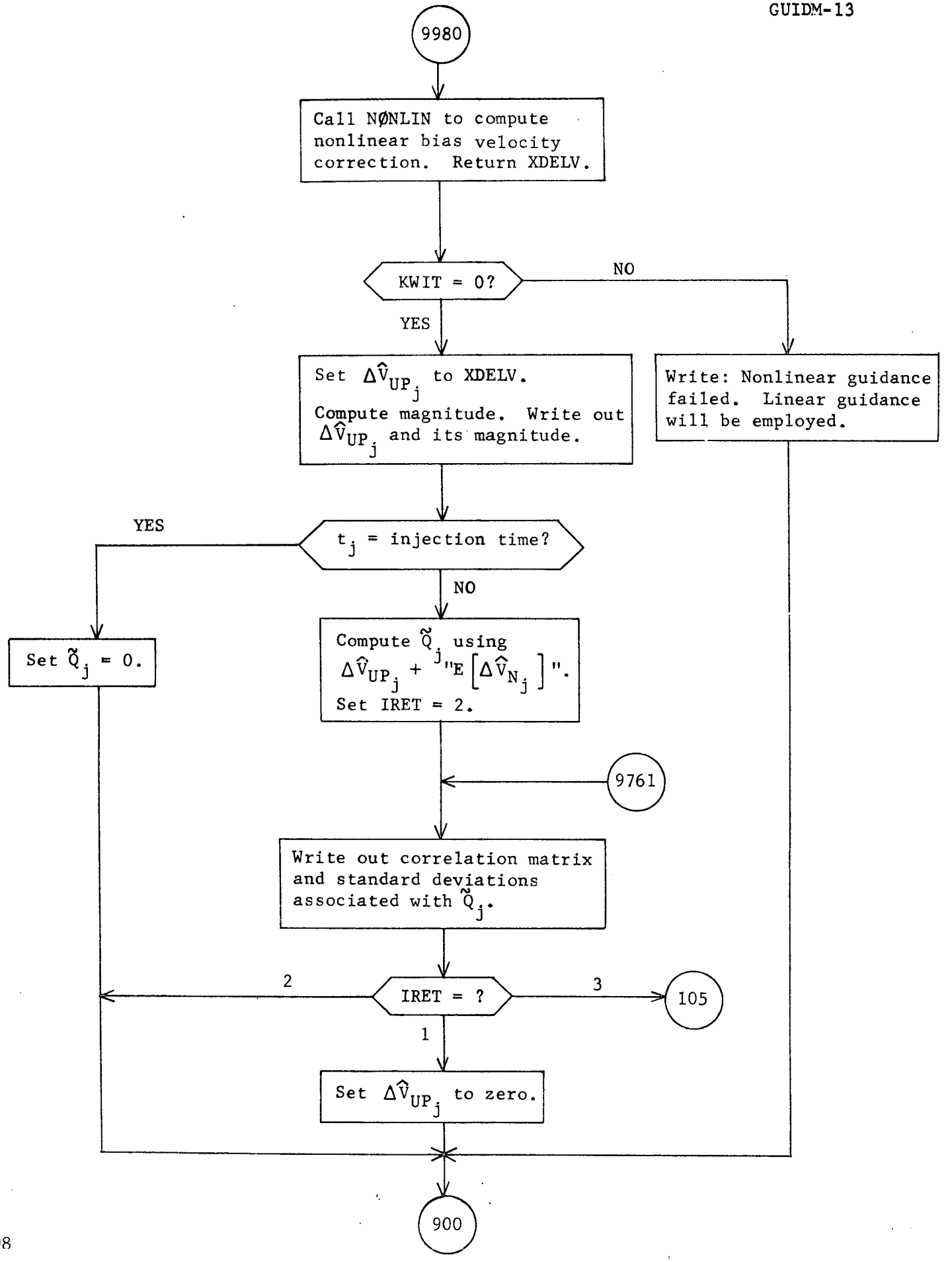


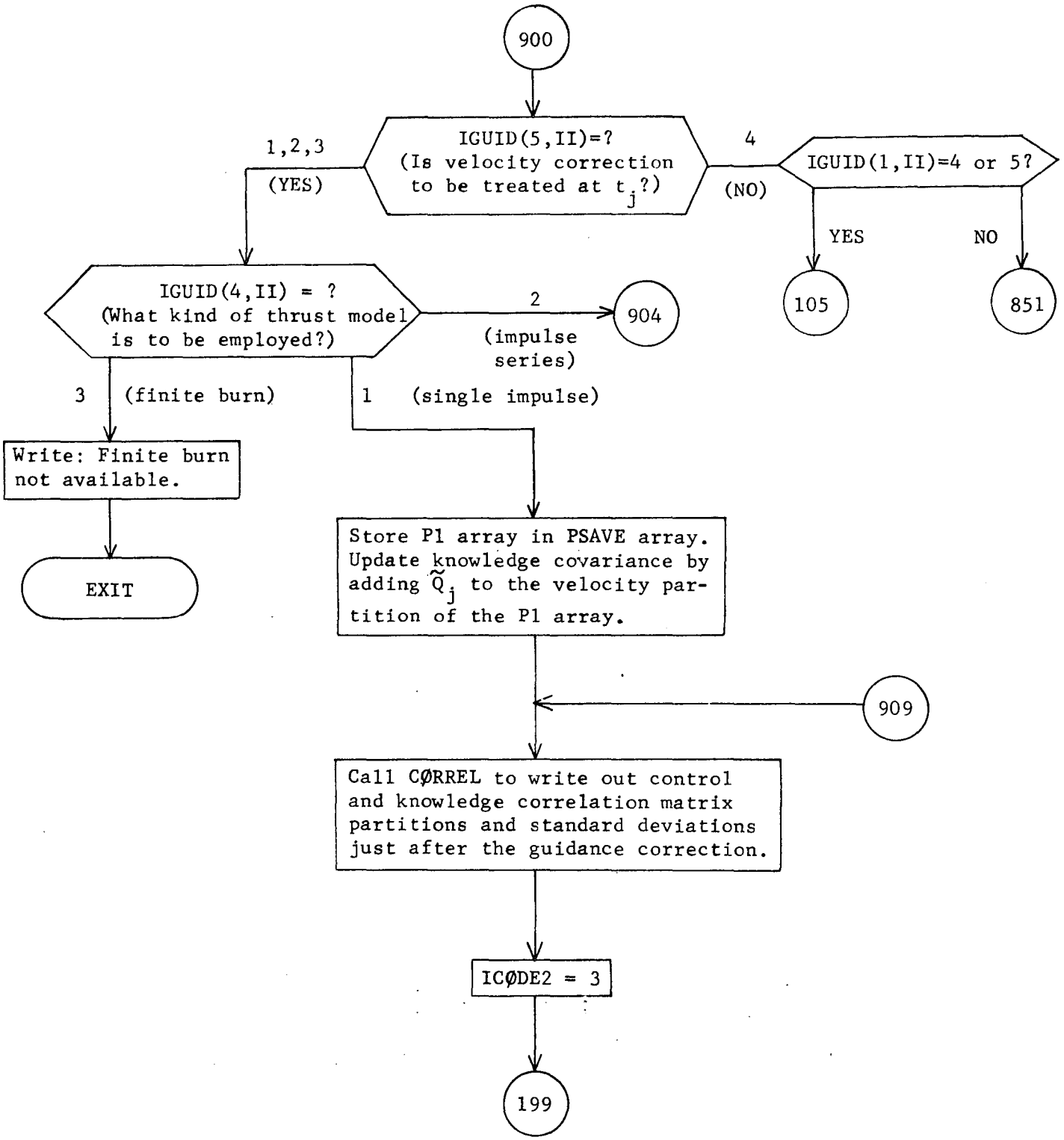

A
Call CØRREL to compute and write out control correlation matrix partitions and standard deviations.
Store position/velocity covariance matrix partition (P1 array) in the PP array for use in subroutine PØICØM.
Set ICØDE2 = 2.
199
Store position partition of the P array in the Z and VEIG arrays.
Call EIGHY to compute and write out the eigenvalues, eigenvectors and hyperellipsoid of the position partition of the P array.
Store velocity partition of the P array in the Z and VEIG arrays.
Call EIGHY to compute and write out the eigenvalues, eigenvectors, and hyperellipsoid of the velocity partition of the P array.
290

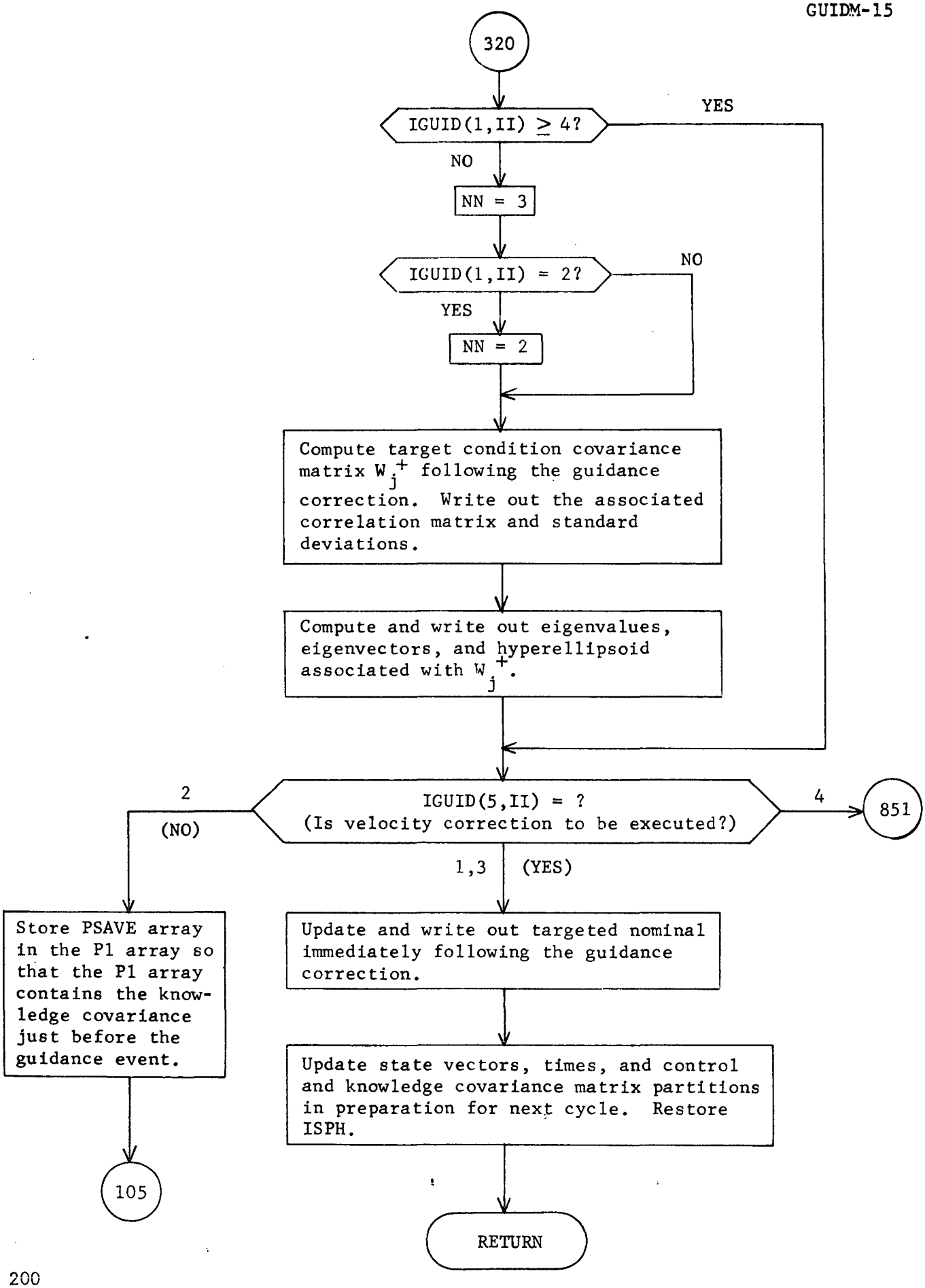

310
Write out complete description of guidance event.
IGUID(5,II) = ?
1
850
2,3,4
854
4,5
IGUID(1,II) = ?
6
851
7
8520
1,2,3
857
Define guidance policy code IGP and execution error code IQP.
Call GUID to compute Γ_j, η_j, $\bar{W}_j$, and certain quantities required for biased aimpoint guidance.
105
YES
ISPH = 0?
NO
Restore state vector, time, and knowledge covariance matrix partitions to their $\bar{t}_j$ values. Restore ISPH.
RETURN
t_j = injection time?
YES
NO
Compute velocity correction covariance matrix S_j. Write out correlation matrix and standard deviations. Compute and write out eigenvalues and eigenvectors.
Set "$E\left[\Delta\hat{V}_{N_j}\right]$" to zero.
862
YES
IGP = 2?
NO
Compute and write out hyperellipsoid of S_j.
858

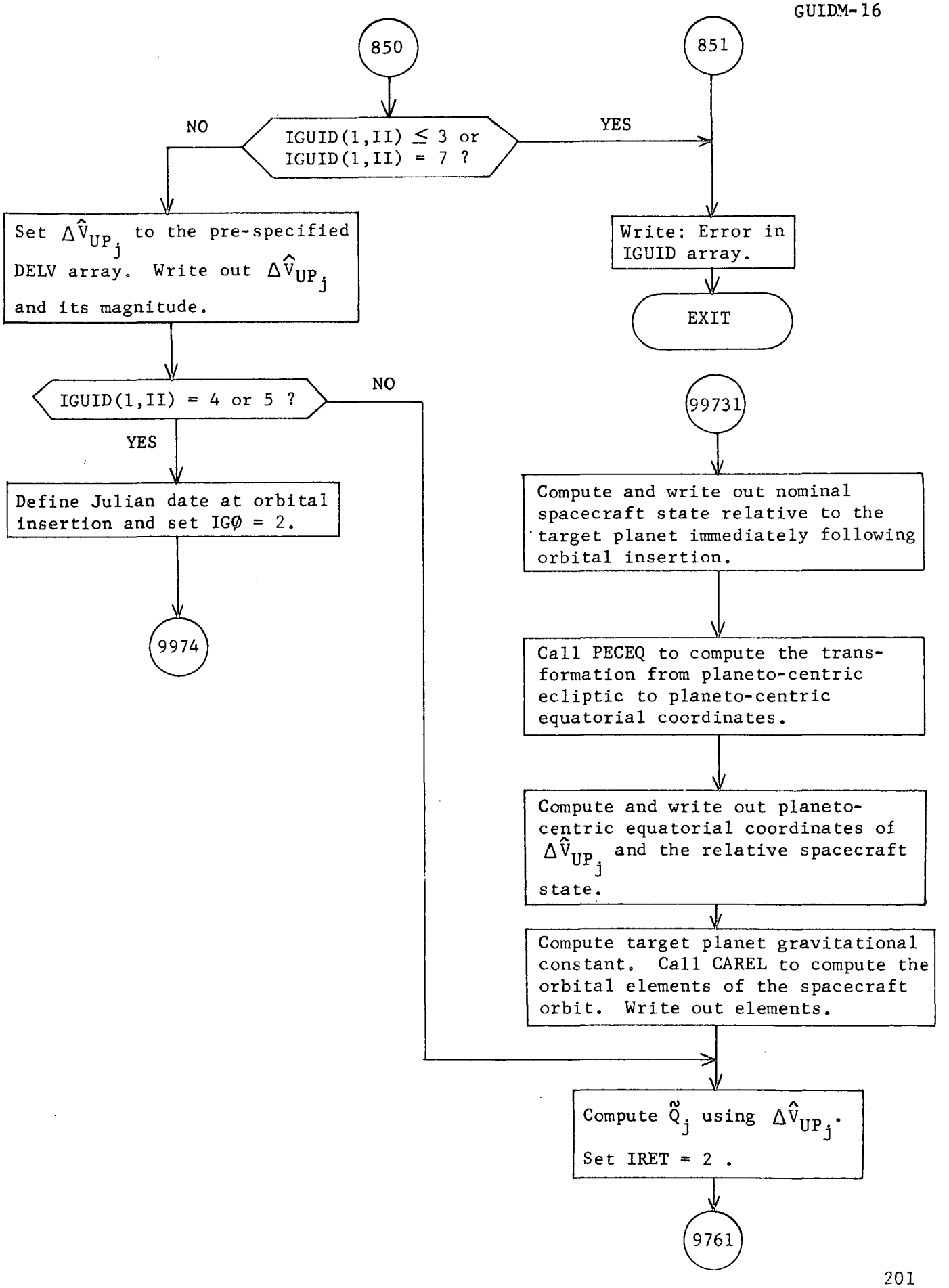

858
IQP = 0?
YES
NO
Use simplified method for computing "E[ΔV̂N_j]". Write out. Compute and write out Q̃_j and its eigenvalues, eigenvectors, and hyperellipsoid.
Use Hoffman-Young formula to compute "E[ΔV̂N_j]". Write out.
900
862
IGUID(3,II) = 0?
NO
YES
Store "E[ΔV̂N_j]" in DVN array for use in subroutine BIAIM.
Call BIAIM to perform biased aimpoint guidance event. Return aimpoint μ_j, bias velocity correction ΔV̂UP_j, and execution error covariance matrix Q̃_j.
B
t_j = injection time?
NO
YES
Compute Q̃_j using "E[ΔV̂N_j]".
Set Q̃_j = 0 and ΔV̂UP_j = 0.
Set IRET = 1
900
9761

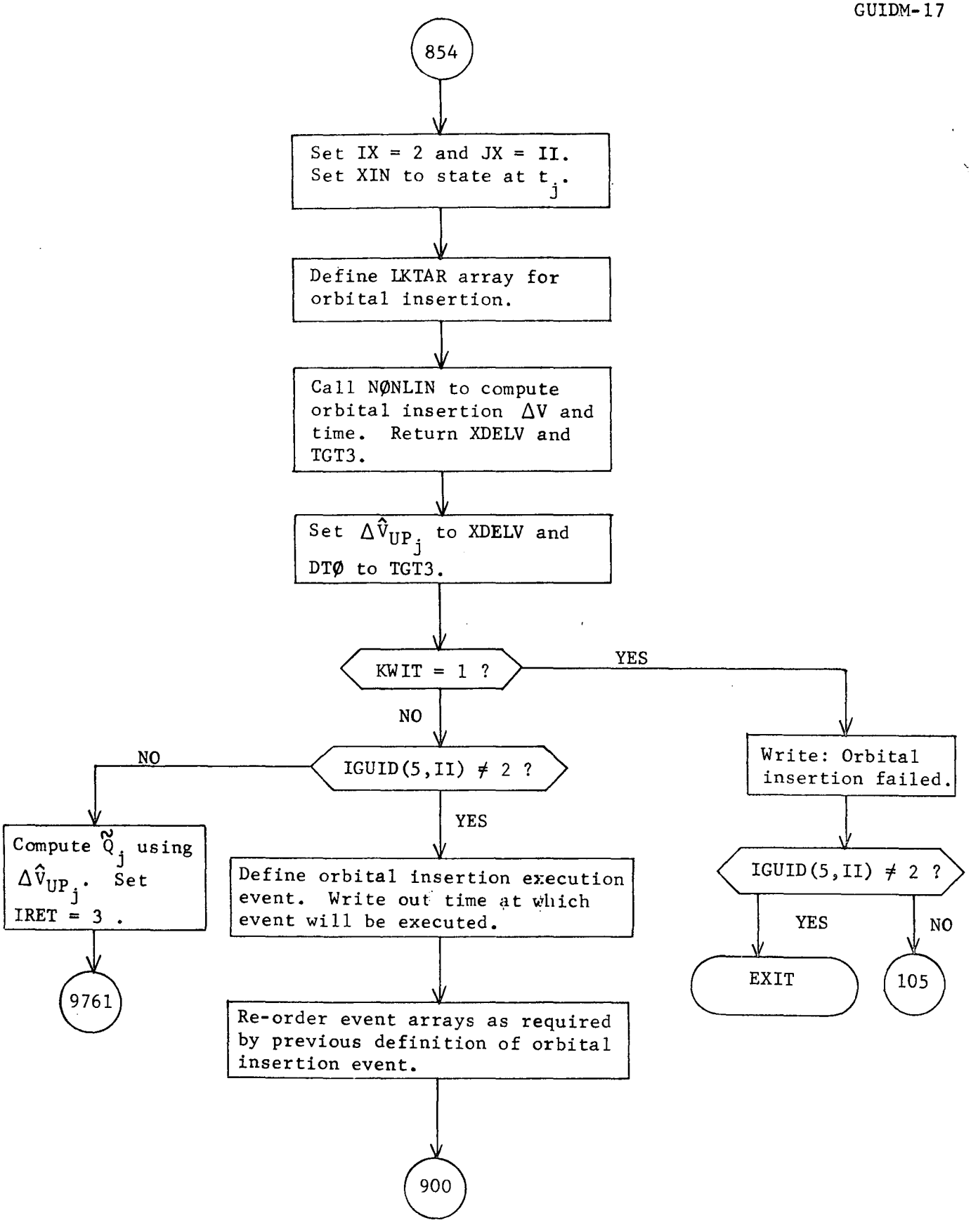

B
IGUID(3,II) = 1?
YES
900
NO
Set IX = 1 and JX = II.
Set XIN to state at t_j.
Write: Bias velocity correction will be recomputed using nonlinear guidance.
IGUID(1,II) = ?
1
FTA
2
2VBP
3
3VBP
Compute Julian date corresponding to t_{CA} and set IGØ = 1.
9974
Compute position and velocity of target planet at the specified time.
Define target variables XTAR, LKTAR, LNPAR, and TGT3 for nonlinear 2VBP guidance.
Define target variables XTAR, LKTAR, LNPAR, and TGT3 for nonlinear 3VBP guidance.
IGØ = ?
2
99731
1
Define target variables XTAR, LKTAR, LNPAR, and TGT3 for nonlinear FTA guidance.
9980

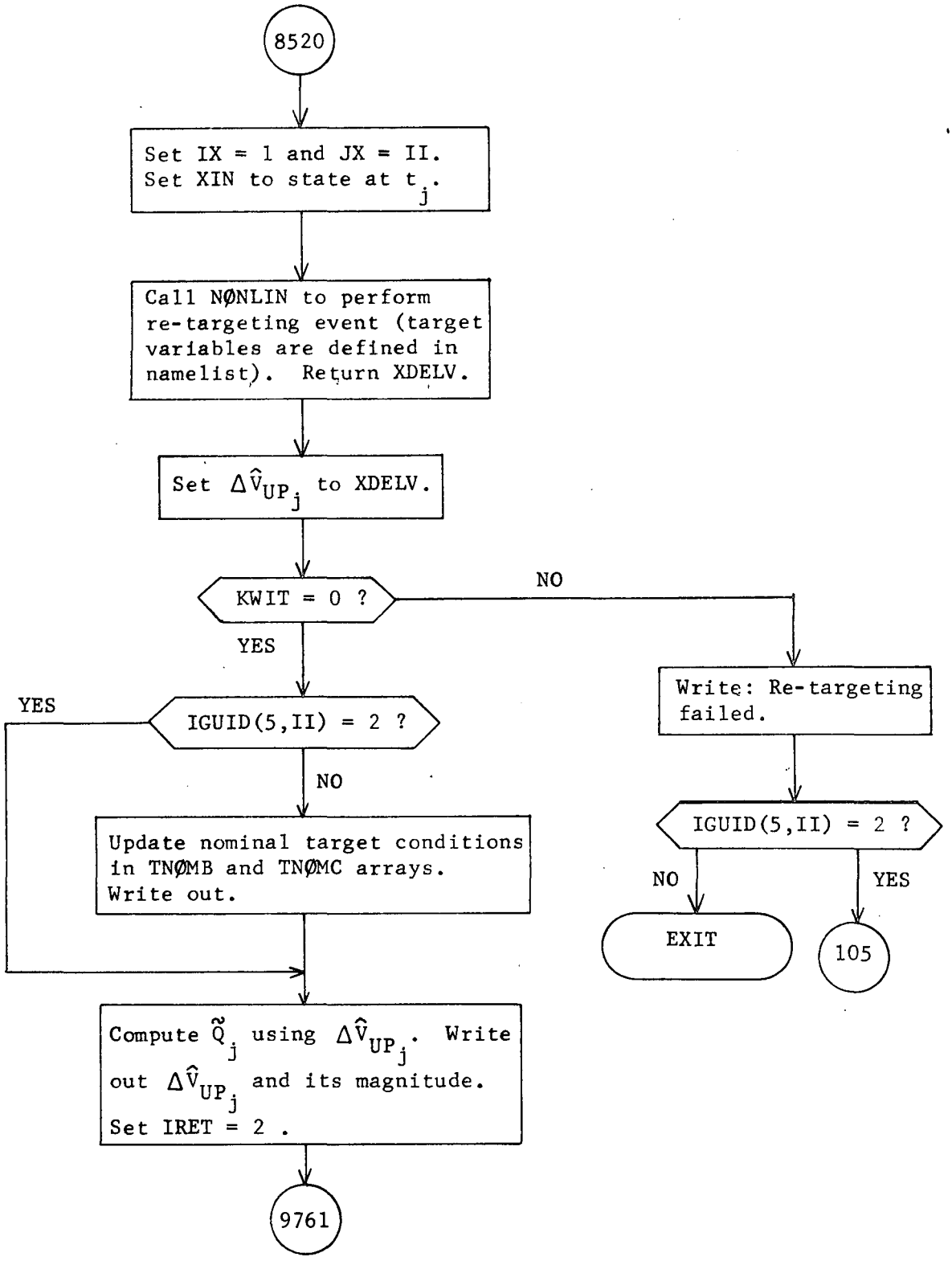

9980
Call NØNLIN to compute nonlinear bias velocity correction. Return XDELV.
KWIT = 0?
NO
YES
Set $\Delta\hat{V}_{UP_j}$ to XDELV.
Compute magnitude. Write out $\Delta\hat{V}_{UP_j}$ and its magnitude.
Write: Nonlinear guidance failed. Linear guidance will be employed.
YES
t_j = injection time?
NO
Set $\tilde{Q}_j$ = 0.
Compute $\tilde{Q}_j$ using $\Delta\hat{V}_{UP_j} + {}^{"}E\left[\Delta\hat{V}_{N_j}\right]{}^{"}$.
Set IRET = 2.
9761
Write out correlation matrix and standard deviations associated with $\tilde{Q}_j$.
2
IRET = ?
3
105
1
Set $\Delta\hat{V}_{UP_j}$ to zero.
900

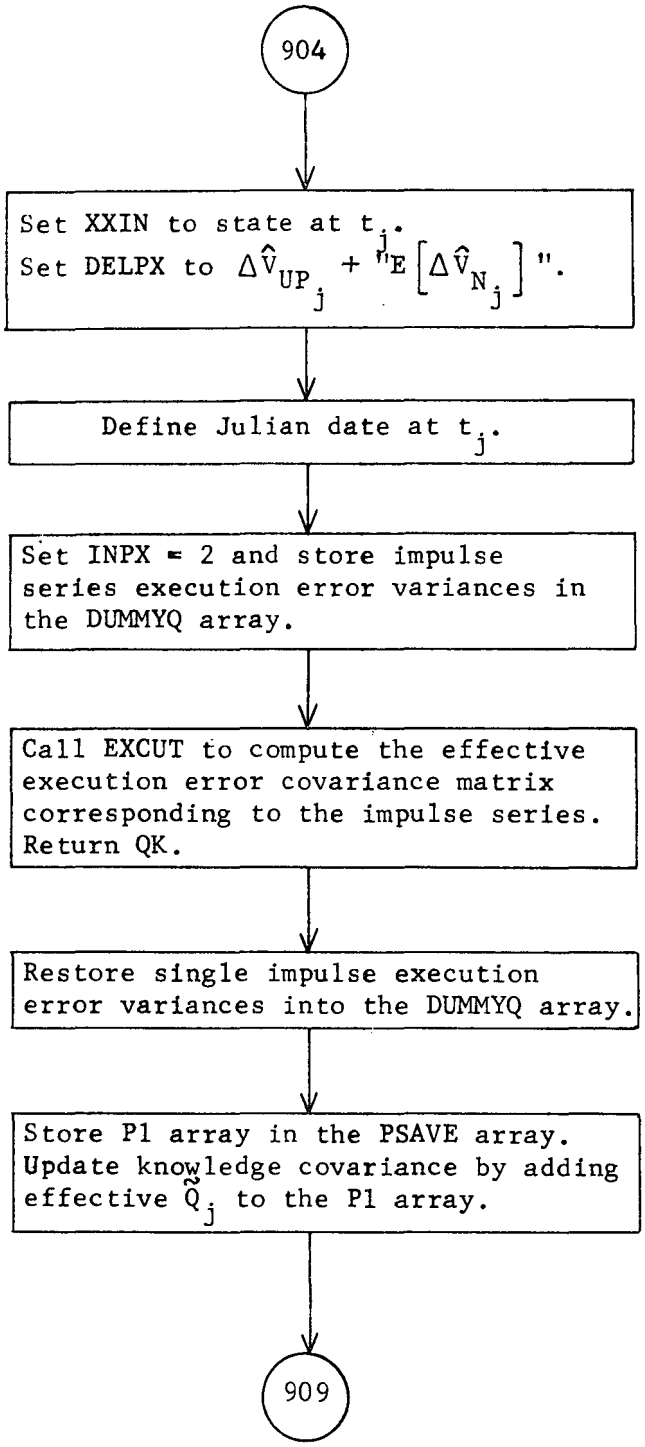

900
IGUID(5,II)=?
(Is velocity correction to be treated at t_j?)
1,2,3
(YES)
4
(NO)
IGUID(1,II)=4 or 5?
YES
NO
105
851
IGUID(4,II) = ?
(What kind of thrust model is to be employed?)
2
(impulse series)
904
3
(finite burn)
1
(single impulse)
Write: Finite burn not available.
EXIT
Store P1 array in PSAVE array. Update knowledge covariance by adding $\tilde{Q}_j$ to the velocity partition of the P1 array.
909
Call CØRREL to write out control and knowledge correlation matrix partitions and standard deviations just after the guidance correction.
ICØDE2 = 3
199


320
IGUID(1,II) ≥ 4?
YES
NO
NN = 3
IGUID(1,II) = 2?
NO
YES
NN = 2
Compute target condition covariance matrix W_j^+ following the guidance correction. Write out the associated correlation matrix and standard deviations.
Compute and write out eigenvalues, eigenvectors, and hyperellipsoid associated with W_j^+.
IGUID(5,II) = ?
(Is velocity correction to be executed?)
2
(NO)
4
851
1,3
(YES)
Store PSAVE array in the P1 array so that the P1 array contains the knowledge covariance just before the guidance event.
Update and write out targeted nominal immediately following the guidance correction.
Update state vectors, times, and control and knowledge covariance matrix partitions in preparation for next cycle. Restore ISPH.
105
RETURN


850
851
NO
IGUID(1,II) ≤ 3 or IGUID(1,II) = 7 ?
YES
Set $\Delta\hat{V}_{UP_j}$ to the pre-specified DELV array. Write out $\Delta\hat{V}_{UP_j}$ and its magnitude.
Write: Error in IGUID array.
EXIT
IGUID(1,II) = 4 or 5 ?
NO
YES
99731
Define Julian date at orbital insertion and set IGØ = 2.
Compute and write out nominal spacecraft state relative to the target planet immediately following orbital insertion.
9974
Call PECEQ to compute the transformation from planeto-centric ecliptic to planeto-centric equatorial coordinates.
Compute and write out planeto-centric equatorial coordinates of $\Delta\hat{V}_{UP_j}$ and the relative spacecraft state.
Compute target planet gravitational constant. Call CAREL to compute the orbital elements of the spacecraft orbit. Write out elements.
Compute $\tilde{Q}_j$ using $\Delta\hat{V}_{UP_j}$.
Set IRET = 2 .
9761


854
Set IX = 2 and JX = II. Set XIN to state at t_j.
Define LKTAR array for orbital insertion.
Call NØNLIN to compute orbital insertion ΔV and time. Return XDELV and TGT3.
Set $\Delta\hat{V}_{UP_j}$ to XDELV and DTØ to TGT3.
KWIT = 1 ?
YES
NO
Write: Orbital insertion failed.
IGUID(5,II) ≠ 2 ?
NO
YES
Compute $\tilde{Q}_j$ using $\Delta\hat{V}_{UP_j}$. Set IRET = 3 .
Define orbital insertion execution event. Write out time at which event will be executed.
IGUID(5,II) ≠ 2 ?
YES
NO
9761
Re-order event arrays as required by previous definition of orbital insertion event.
EXIT
105
900


8520
Set IX = 1 and JX = II.
Set XIN to state at t_j.
Call NØNLIN to perform re-targeting event (target variables are defined in namelist). Return XDELV.
Set $\Delta\hat{V}_{UP_j}$ to XDELV.
KWIT = 0 ?
NO
YES
Write: Re-targeting failed.
IGUID(5,II) = 2 ?
YES
IGUID(5,II) = 2 ?
NO
Update nominal target conditions in TNØMB and TNØMC arrays. Write out.
NO
YES
EXIT
105
Compute $\tilde{Q}_j$ using $\Delta\hat{V}_{UP_j}$. Write out $\Delta\hat{V}_{UP_j}$ and its magnitude.
Set IRET = 2 .
9761


904
Set XXIN to state at t_j.
Set DELPX to $\Delta\hat{V}_{UP_j}$ + "$E\left[\Delta\hat{V}_{N_j}\right]$".
Define Julian date at t_j.
Set INPX = 2 and store impulse series execution error variances in the DUMMYQ array.
Call EXCUT to compute the effective execution error covariance matrix corresponding to the impulse series. Return QK.
Restore single impulse execution error variances into the DUMMYQ array.
Store P1 array in the PSAVE array. Update knowledge covariance by adding effective $\tilde{Q}_j$ to the P1 array.
909

GUIS Analysis

Subroutine GUIS is called at a midcourse guidance event at t_j in the simulation mode to compute three primary quantities for the selected midcourse guidance policy. These three quantities are the variation matrix η_j, the target condition covariance matrix prior to the velocity correction $\bar{W}_j$, and the guidance matrix Γ_j. Three midcourse guidance policies are available: fixed-time-of-arrival (FTA), two-variable B-plane (2VBP), and three-variable B-plane (3VBP). All are linear impulsive guidance policies having form

$$\Delta\hat{V}_j = \Gamma_j \delta\hat{X}_j$$

where $\Delta\hat{V}_j$ is the commanded velocity correction, and $\delta\hat{X}_j$ is the estimate of the spacecraft position/velocity deviation from the targeted nominal. The relevant equations for each guidance policy will be summarized below.

The variation matrix η_j for FTA guidance relates deviations in spacecraft state at t_j to position deviations at time of closest approach t_{CA}, and is given by

$$\eta_j = \left[\phi_1 \mid \phi_2\right]$$

where $\left[\phi_1 \mid \phi_2\right]$ is the upper half of the state transition matrix $\Phi(t_{CA}, t_j)$. The guidance matrix for FTA guidance is given by

$$\Gamma_j = \left[-\phi_2^{-1}\phi_1 \mid -I\right] .$$

The variation matrix for 3VBP guidance relates deviations in spacecraft state at t_j to deviations in B•T, B•R, and t_{SI}, where t_{SI} is the time at which the sphere of influence is pierced. Unlike the variation matrix for FTA guidance, which can be computed analytically or by numerical differencing, the 3VBP variation matrix must always be computed using numerical differencing since no good analytical formulas are available which relate deviations in spacecraft state at t_j to deviations in t_{SI}. If the variation matrix is written as

$$\eta_j = \left[\eta_1 \mid \eta_2\right]$$

then the guidance matrix for 3VBP guidance is given by

$$\Gamma_j = \left[-\eta_2^{-1}\eta_1 \mid -I\right] .$$

The variation matrix for 2VBP guidance relates deviations in spacecraft state at t_j to deviations in B•T and B•R and is given by

$$\eta_j = M \Phi (t_{SI}, t_j)$$

where M is an analytically computed matrix relating B•T and B•R deviations to spacecraft state deviations at t_{SI}, and $\Phi(t_{SI}, t_j)$ is the state transition matrix over $[t_j, t_{SI}]$. If η_j is written as

$$\eta_j = \left[\begin{array}{c|c} A & B \end{array}\right]$$

then the guidance matrix for 2VBP guidance is given by

$$\Gamma_j = \left[\begin{array}{c|c} -B^T(BB^T)^{-1}A & -B^T(BB^T)^{-1}B \end{array}\right]$$

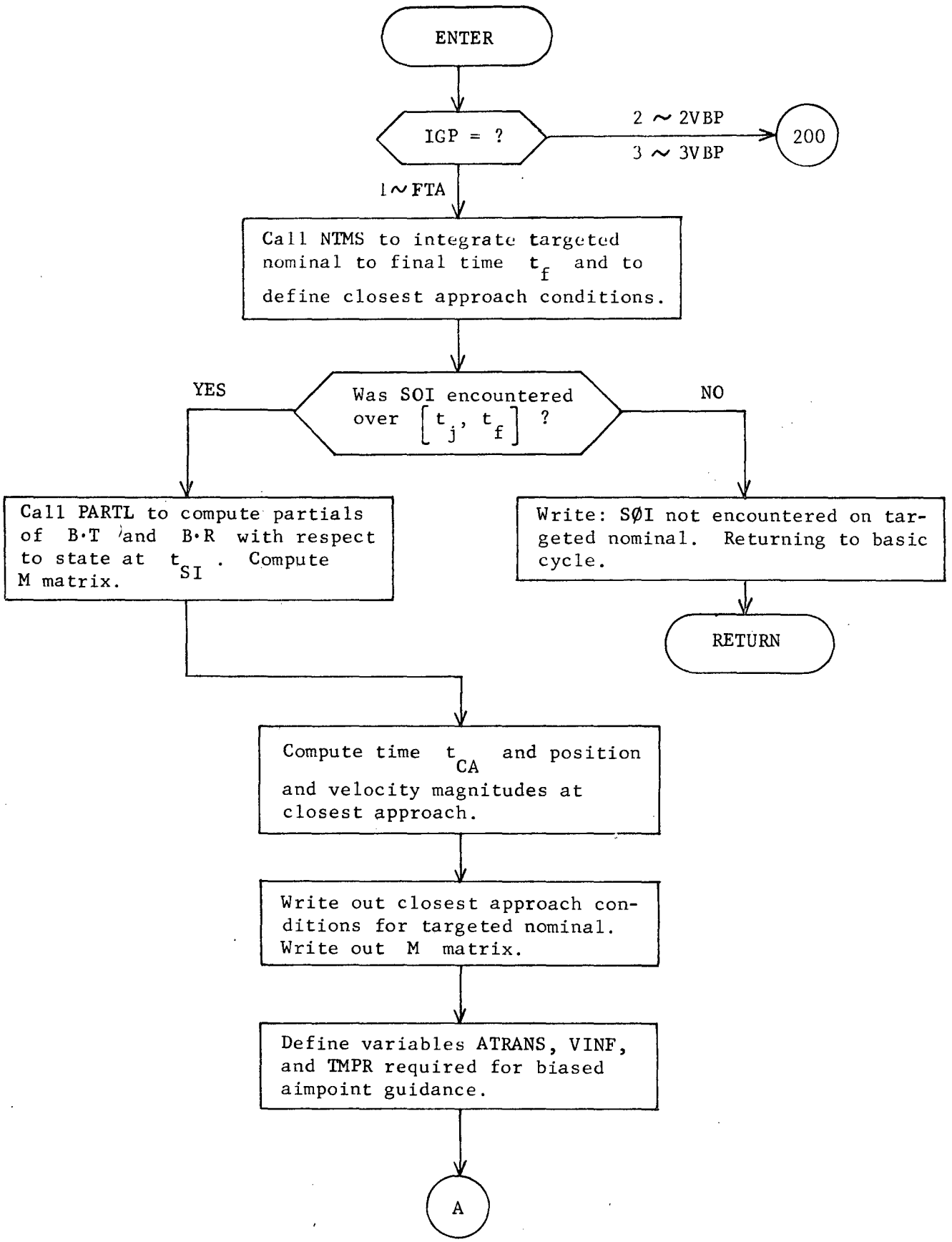
All state transition matrices and, hence, all variation matrices used by the above three guidance policies are referenced to the most recent nominal trajectory for improved numerical accuracy.

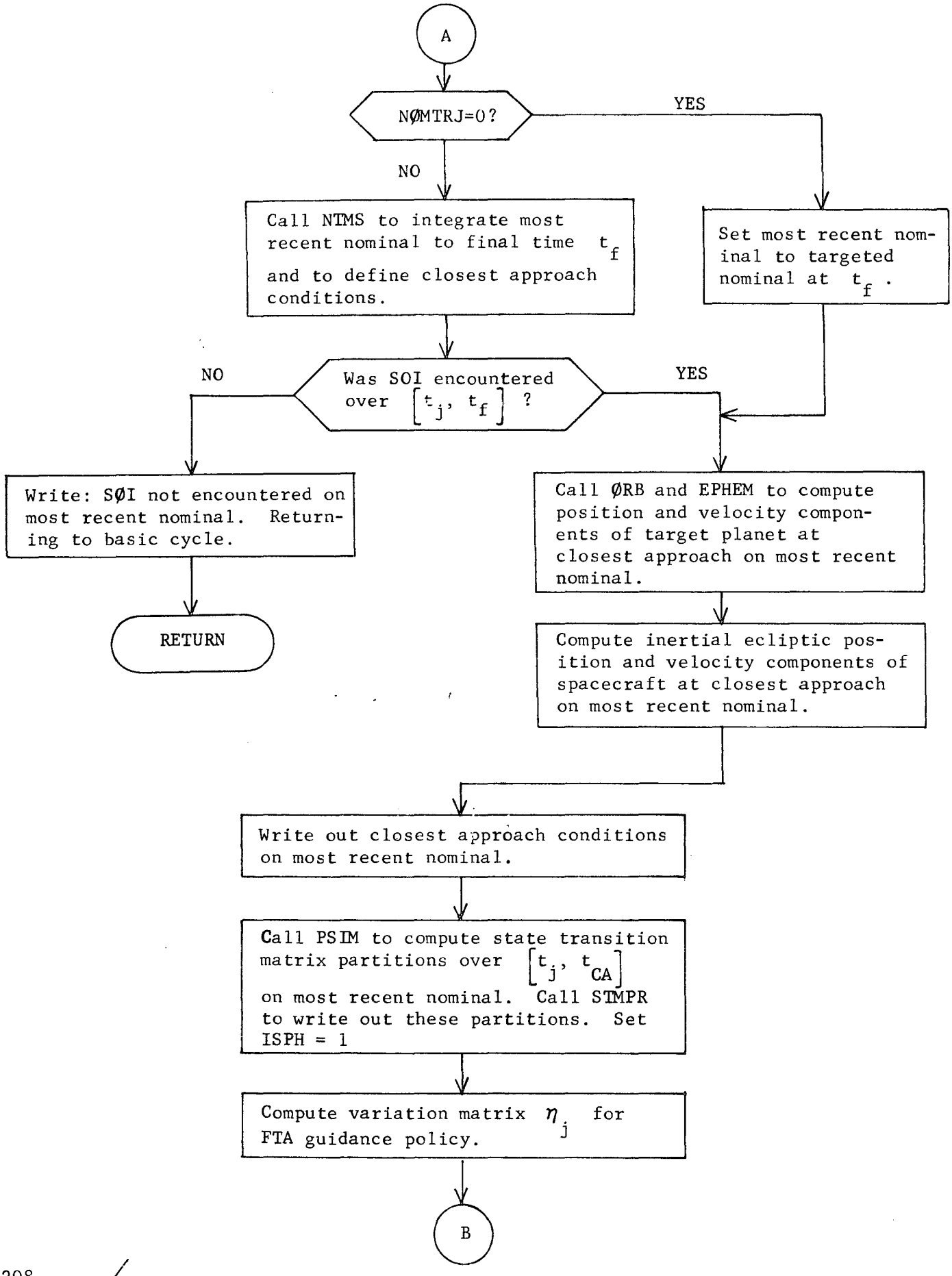
Once the variation matrix η_j is available for any of the above guidance policies, the target condition covariance matrix can be computed using

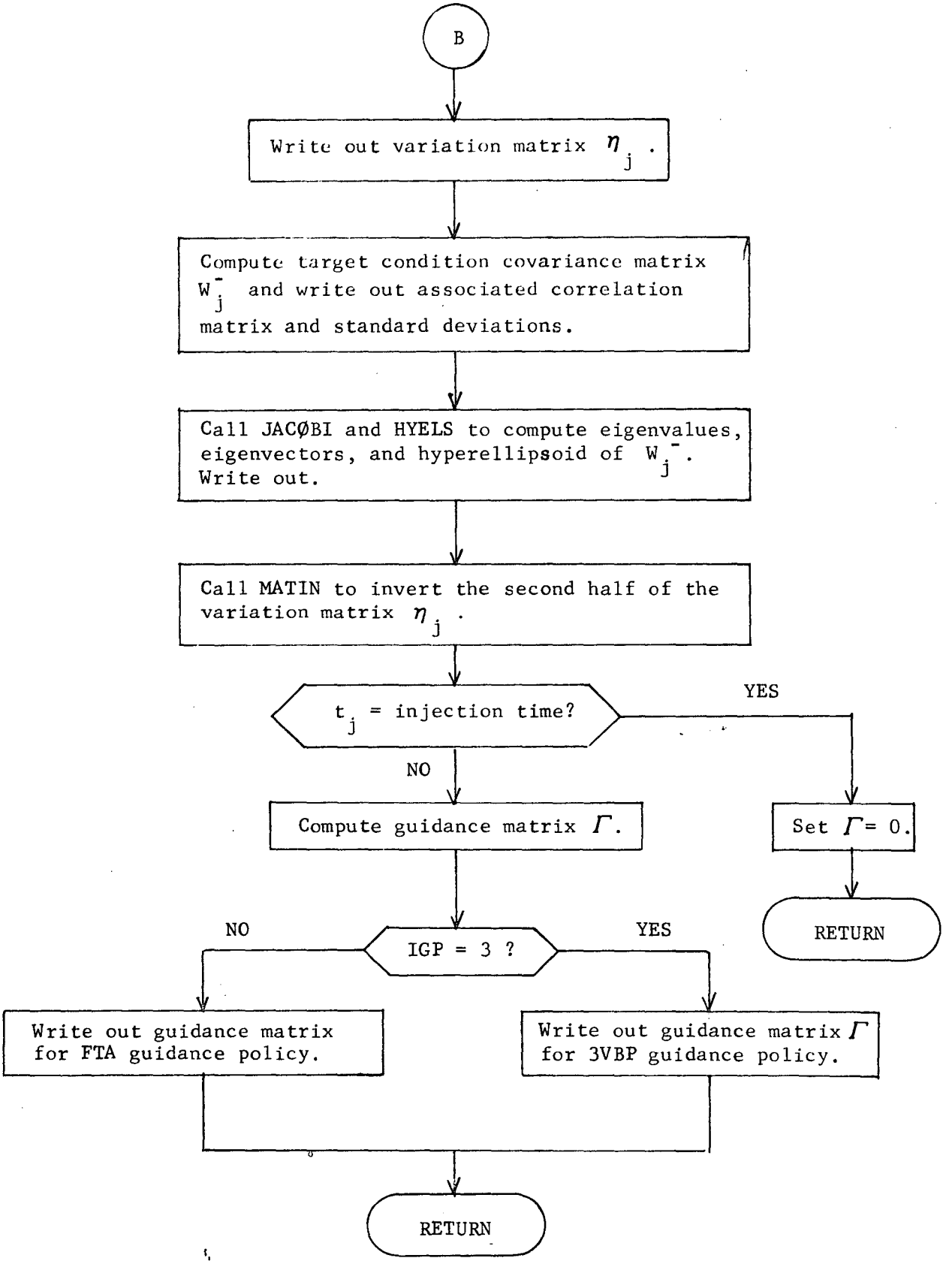
$$\bar{W}_j = \eta_j \bar{P}_{c_j} \eta_j^T$$

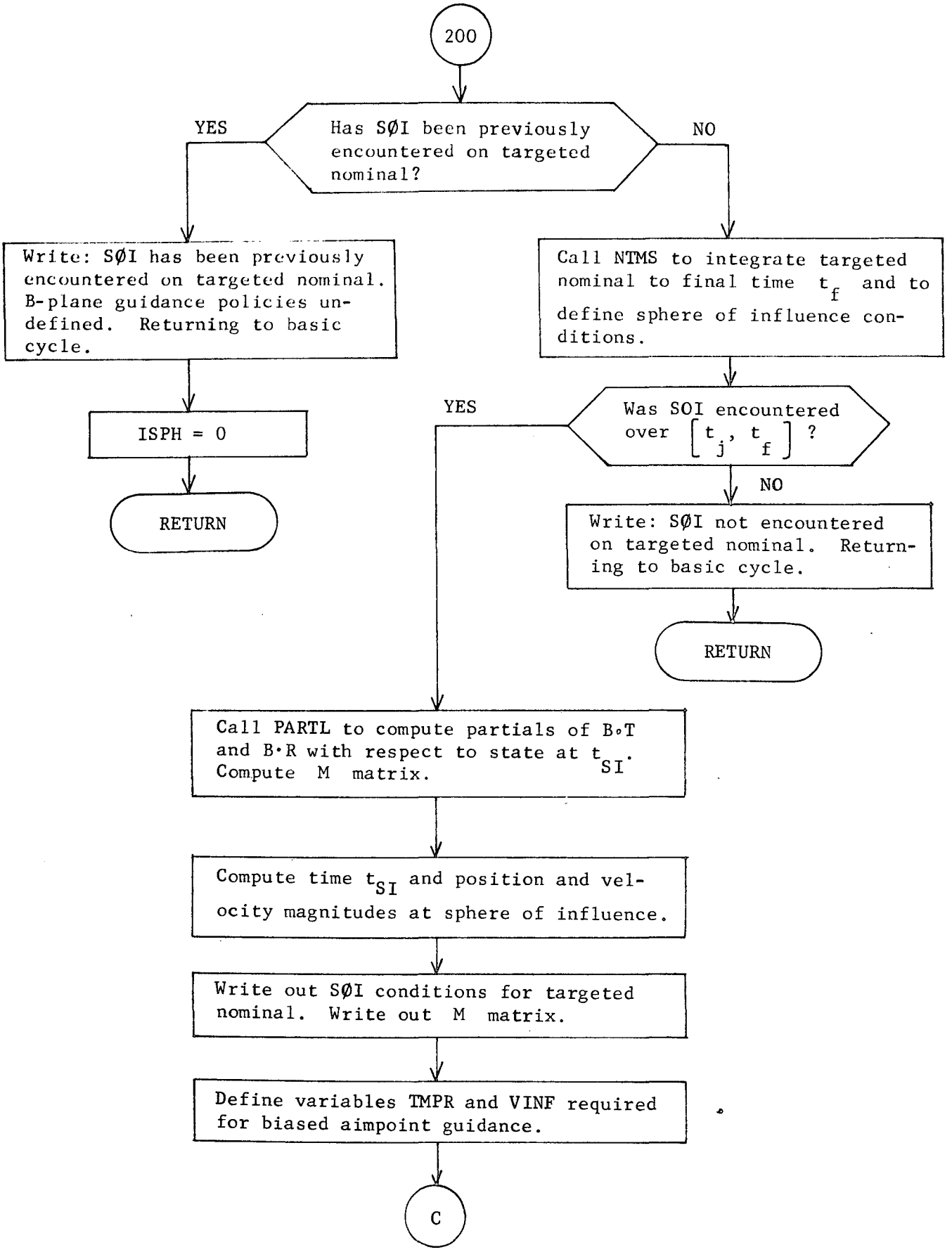
where $\bar{P}_{c_j}$ is the control covariance matrix immediately prior to the guidance event.

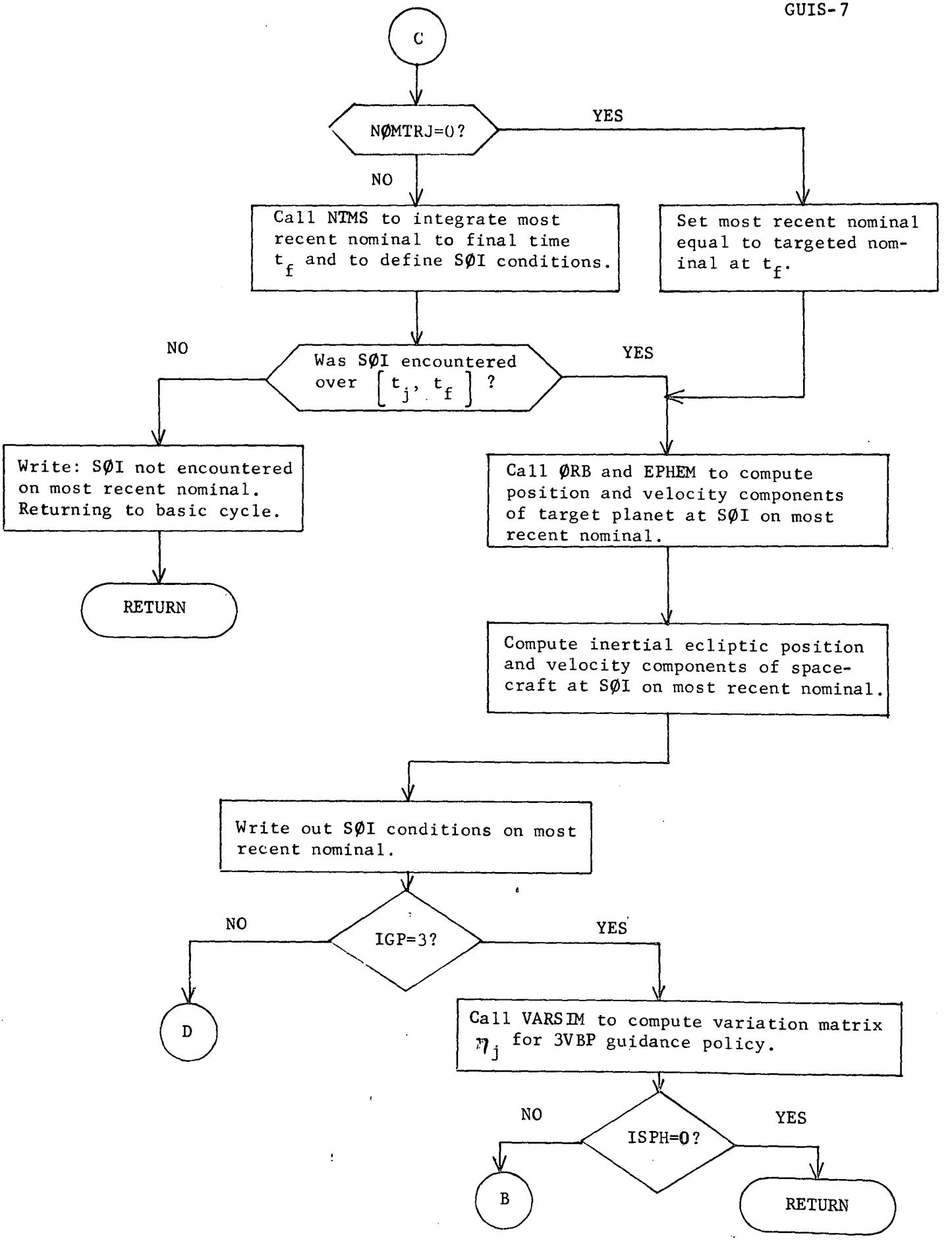
GUIS Flow Chart

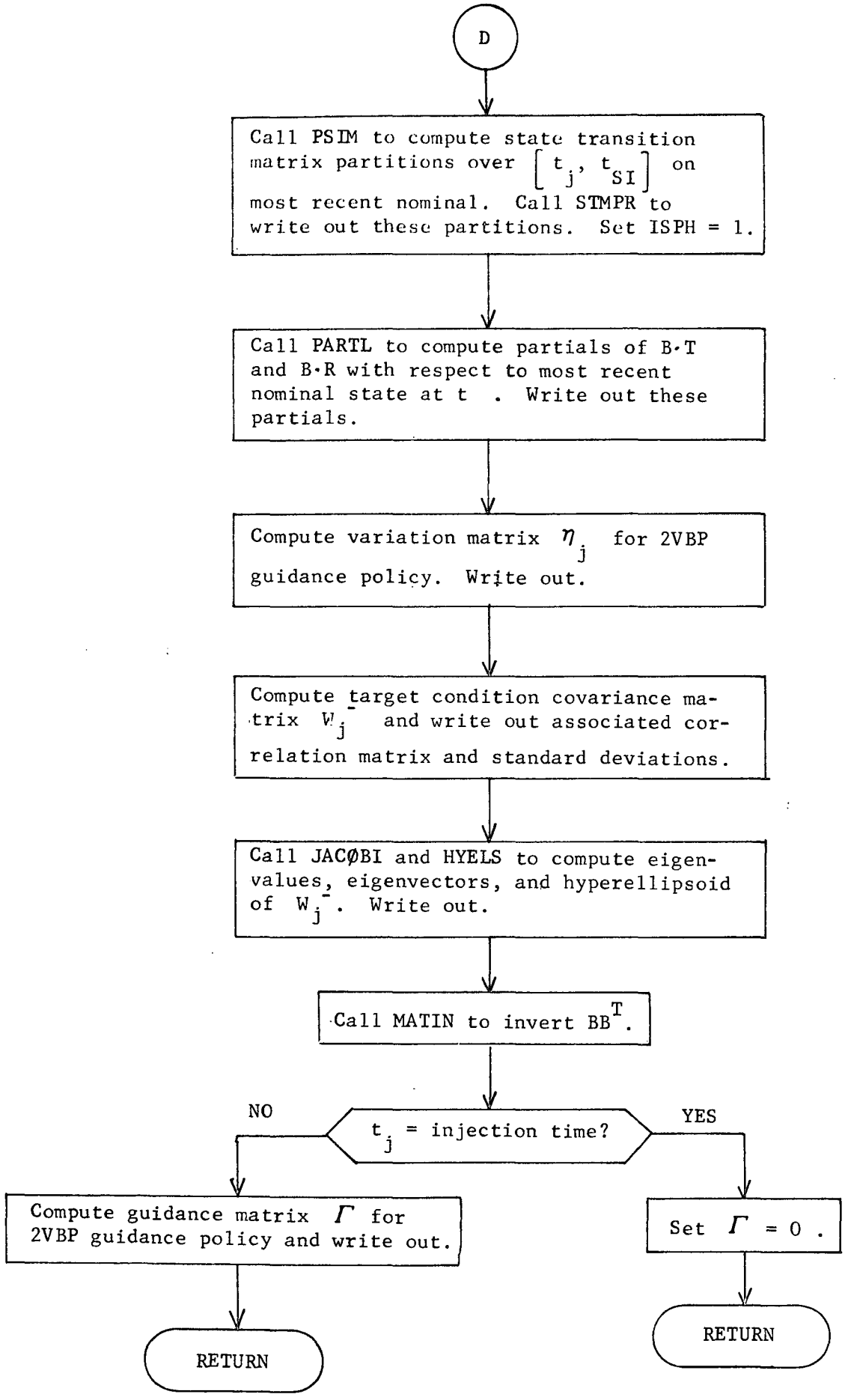



A
NØMTRJ=0?
YES
NO
Call NTMS to integrate most recent nominal to final time t_f and to define closest approach conditions.
Set most recent nominal to targeted nominal at t_f.
Was SOI encountered over $[t_j, t_f]$?
NO
YES
Write: SØI not encountered on most recent nominal. Returning to basic cycle.
RETURN
Call ØRB and EPHEM to compute position and velocity components of target planet at closest approach on most recent nominal.
Compute inertial ecliptic position and velocity components of spacecraft at closest approach on most recent nominal.
Write out closest approach conditions on most recent nominal.
Call PSIM to compute state transition matrix partitions over $[t_j, t_{CA}]$ on most recent nominal. Call STMPR to write out these partitions. Set ISPH = 1
Compute variation matrix η_j for FTA guidance policy.
B


B
Write out variation matrix η_j .
Compute target condition covariance matrix $\bar{W}_j$ and write out associated correlation matrix and standard deviations.
Call JACØBI and HYELS to compute eigenvalues, eigenvectors, and hyperellipsoid of $\bar{W}_j$. Write out.
Call MATIN to invert the second half of the variation matrix η_j .
t_j = injection time?
YES
NO
Compute guidance matrix Γ.
Set Γ = 0.
RETURN
NO
IGP = 3 ?
YES
Write out guidance matrix for FTA guidance policy.
Write out guidance matrix Γ for 3VBP guidance policy.
RETURN


200
Has SØI been previously encountered on targeted nominal?
YES
NO
Write: SØI has been previously encountered on targeted nominal. B-plane guidance policies undefined. Returning to basic cycle.
ISPH = 0
RETURN
Call NTMS to integrate targeted nominal to final time t_f and to define sphere of influence conditions.
Was SOI encountered over $[t_j, t_f]$?
YES
NO
Write: SØI not encountered on targeted nominal. Returning to basic cycle.
RETURN
Call PARTL to compute partials of B•T and B•R with respect to state at t_{SI}. Compute M matrix.
Compute time t_{SI} and position and velocity magnitudes at sphere of influence.
Write out SØI conditions for targeted nominal. Write out M matrix.
Define variables TMPR and VINF required for biased aimpoint guidance.
C


C
NØMTRJ=0?
YES
NO
Call NTMS to integrate most recent nominal to final time t_f and to define SØI conditions.
Set most recent nominal equal to targeted nominal at t_f.
NO
Was SØI encountered over $[t_j, t_f]$?
YES
Write: SØI not encountered on most recent nominal. Returning to basic cycle.
RETURN
Call ØRB and EPHEM to compute position and velocity components of target planet at SØI on most recent nominal.
Compute inertial ecliptic position and velocity components of spacecraft at SØI on most recent nominal.
Write out SØI conditions on most recent nominal.
NO
IGP=3?
YES
D
Call VARSIM to compute variation matrix η_j for 3VBP guidance policy.
NO
ISPH=0?
YES
B
RETURN


D
Call PSIM to compute state transition matrix partitions over $[t_j, t_{SI}]$ on most recent nominal. Call STMPR to write out these partitions. Set ISPH = 1.
Call PARTL to compute partials of B·T and B·R with respect to most recent nominal state at t_j. Write out these partials.
Compute variation matrix η_j for 2VBP guidance policy. Write out.
Compute target condition covariance matrix W_j and write out associated correlation matrix and standard deviations.
Call JACØBI and HYELS to compute eigenvalues, eigenvectors, and hyperellipsoid of W_j. Write out.
Call MATIN to invert BB^T.
t_j = injection time?
NO
YES
Compute guidance matrix Γ for 2VBP guidance policy and write out.
Set $\Gamma = 0$.
RETURN
RETURN

GUISIM Analysis

Subroutine GUISIM is the executive guidance subroutine in the simulation program. In addition to controlling the computational flow for all types of guidance events, GUISIM also performs many of the required guidance computations itself.

Before considering each type of guidance event, the treatment of a general guidance event will be discussed. Let t_j be the time at which the guidance event occurs. Before any guidance event can be executed the targeted nominal state $\bar{X}_j^-$, most recent nominal state $\tilde{X}_j^-$, estimated state deviation $\delta\hat{\tilde{X}}_j^-$ from most recent nominal, actual state deviation $\delta\tilde{X}_j^-$ from most recent nominal, knowledge covariance $P_{K_j}^-$, and control covariance $P_{c_j}^-$ must all be available, where $(\,)^-$ indicates values immediately before the event. Only the control covariance is not available prior to entering GUISIM. The propagation of the control covariance over the interval $[t_{j-1}, t_j]$, where t_{j-1} denotes the time of the previous guidance event, is performed within GUISIM.

The next step in the treatment of a general guidance event is concerned with the computation of the commanded velocity correction, execution error covariance, actual execution error, and actual velocity correction. In the simulation program a non-statistical commanded velocity correction can always be computed. This commanded velocity correction $\Delta\hat{V}_j$ is used to compute the execution error covariance matrix $\tilde{Q}_j$ and the actual execution error $\delta\Delta V_j$. A summary of the execution error model and the equations used to compute $\tilde{Q}_j$ and $\delta\Delta V_j$ can be found in the subroutine QCØMP analysis section. The actual velocity correction is then computed using the equation

$$\Delta V_j = \Delta\hat{V}_j + \delta\Delta V_j$$

The last step is concerned with the updating of required quantities prior to returning to the basic cycle. An assumption underlying the modeled guidance process is that the targeted nominal is always updated by the commanded velocity correction. In the simulation program the update velocity correction $\Delta\hat{V}_{UP_j}$ is always identical to the commanded velocity correction $\Delta\hat{V}_j$. This is in contrast to the error analysis program where $\Delta\hat{V}_{UP_j}$ is equated with the non-statistical component of $\Delta\hat{V}_j$. The

most recent and targeted nominal states immediately following the guidance event are updated using the equations

$$\tilde{X}_j^+ = \tilde{X}_j^- + \delta\tilde{\hat{X}}_j^- + \begin{bmatrix} - & 0 & - \\ \delta\Delta\hat{V}_{UP_j} \end{bmatrix}$$

$$\bar{X}_j^+ = \tilde{X}_j^+$$

The actual and estimated state deviations from the most recent nominal are given by

$$\delta\tilde{X}_j^+ = \delta\tilde{X}_j^- - \delta\tilde{\hat{X}}_j^- + \begin{bmatrix} - & 0 \\ \delta\Delta V_j^- \end{bmatrix}$$

$$\delta\tilde{\hat{X}}_j^+ = 0$$

The previous 4 equations assume an impulsive thrust model. If, instead, the thrust is modeled as an impulse series, then an effective estimated state $\hat{X}_{eff}$ and an effective actual state X_{eff} are computed. The equations used to compute these effective states are summarized in the subroutine PULSEX analysis section. The previous update equations are then replaced by the following equations

$$\tilde{X}_j^+ = \hat{X}_{eff}$$

$$\bar{X}_j^+ = \tilde{X}_j^+$$

$$\delta\tilde{X}_j^+ = X_{eff} - \hat{X}_{eff}$$

$$\delta\hat{X}_j^+ = 0$$

The knowledge covariance is updated using the equation

$$P_{K_j}^+ = P_{K_j}^- + \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \tilde{Q}_j \end{array}\right]$$

if an impulsive thrust model is assumed. If the thrust is modeled as a series of impulses, then an effective execution error covariance $\tilde{Q}_{eff}$ is computed and the knowledge covariance is updated using the equation

$$P_{K_j}^{+} = P_{K_j}^{-} + \tilde{Q}_{eff}$$

In either case the control covariance is updated simply by setting

$$P_{c_j}^{+} = P_{K_j}^{+}$$

This equation is a direct consequence of the assumption that the targeted nominal is always updated at a guidance event.

A "compute only" option is available in GUISIM in which all of the $( )^{+}$ quantities will still be computed and printed. However, states, deviations, and covariances are then reset to their former $( )^{-}$ values prior to returning to the basic cycle.

Each specific type of guidance event involves the computation of other quantities not discussed above. These will be covered in the following discussion of specific guidance events.

1. Midcourse and biased aimpoint guidance.

   Linear midcourse guidance policies have form

$$\Delta\hat{V}_{N_j} = \Gamma_j \, \delta\hat{X}_j$$

where the subscript N indicates that this is the velocity correction required to null out deviations from the nominal target state. This notation is required to differentiate between this type of velocity correction and velocity corrections required to achieve an altered target state. Linear midcourse guidance policies are discussed in more detail in the subroutine GUIS analysis section.

Subroutine GUISIM calls GUIS to compute the guidance matrix, Γ_j, and the target condition covariance immediately prior to the guidance event, W_j^{-}, and then uses Γ_j to compute the velocity correction covariance S_j, which is defined as

$$S_j = E\left[\Delta\hat{V}_{N_j}\,\Delta\hat{V}_{N_j}^T\right],$$

and is given by the equation

$$S_j = \Gamma_j (P_{c_j}^- - P_{K_j}^-)\ \Gamma_j^T$$

This equation assumes that an optimal estimation algorithm is employed in the navigation process, since the derivation of this equation requires the orthogonality of the estimate and the estimation error.

Since state estimates $\delta\hat{X}_j$ are generated in the simulation program, an actual $\Delta\hat{V}_{N_j}$ can always be computed. This is in contrast to the error analysis program where only a statistical or effective $\Delta\hat{V}_{N_j}$ can be computed. The perfect velocity correction $\underline{\Delta V}_j$, defined as the velocity correction required to null out actual deviations from the nominal target state, is also computed for midcourse guidance events. Assuming linear guidance theory, the perfect velocity correction is given by

$$\underline{\Delta V}_j = \Gamma_j\ \delta X_j$$

where δX_j is the actual deviation from the targeted nominal. An option is also available in GUISIM for re-computing $\Delta\hat{V}_{N_j}$ using nonlinear techniques. However, it should be noted that the nonlinear two-variable B-plane guidance policy, unlike the corresponding linear policy, constrains the z-component of ΔV_{N_j} to be zero.

If planetary quarantine constraints must be satisfied at a midcourse correction, GUISIM calls BIAIM to compute the new aimpoint μ_j and the bias velocity correction $\Delta\hat{V}_{B_j}$. All computations in BIAIM are based on linear guidance theory. However, an option is available in GUISIM to re-compute the total velocity correction $\Delta\hat{V}_{B_j} + \Delta\hat{V}_{N_j}$, but not μ_j, using non-linear techniques. This option is recommended if a biased aimpoint guidance event occurs at t_j = injection time. It should also be noted that $\tilde{Q}_j$ is set to zero if t_j = injection time since it is assumed that the injection

covariance does not change for small changes in injection velocity.

After the updated control covariance $P_{c_j}^+$ has been computed, the target condition covariance matrix W_j^+ following the guidance correction is computed using the equation

$$W_j^+ = \eta_j \, P_{c_j}^+ \, \eta_j^T$$

where variation matrix η_j has been previously computed in subroutine GUIS.

2. Re-targeting.

In the simulation (and error analysis) program a re-targeting event is defined to be the computation of a velocity correction $\Delta\hat{V}_{RT}$ required to achieve a new set of target conditions using nonlinear techniques. Since the state estimate $\tilde{X}_j^- + \delta\tilde{\hat{X}}_j^-$ is used as the zero-th iterate in the re-targeting process, the new target conditions must be close enough to the original nominal target conditions to ensure a convergent process.

It should be noted that after a re-targeting event the new target conditions are henceforth treated as the nominal target conditions.

3. Orbital insertion.

An orbital insertion event is divided into a decision event and an execution event. At a decision event the orbital insertion velocity correction $\Delta\hat{V}_{\emptyset I}$ and the time interval Δt separating decision and execution are computed based on the state estimate $\tilde{X}_j^- + \delta\tilde{\hat{X}}_j^-$. The relevant equations can be found in the subroutine CØPINS analysis section for coplanar orbital insertion; in NØPINS, for non-planar orbital insertion. Before returning to the basic cycle, GUISIM schedules the orbital insertion execution event to occur at $t_j + \Delta t$ and re-orders the necessary event arrays accordingly.

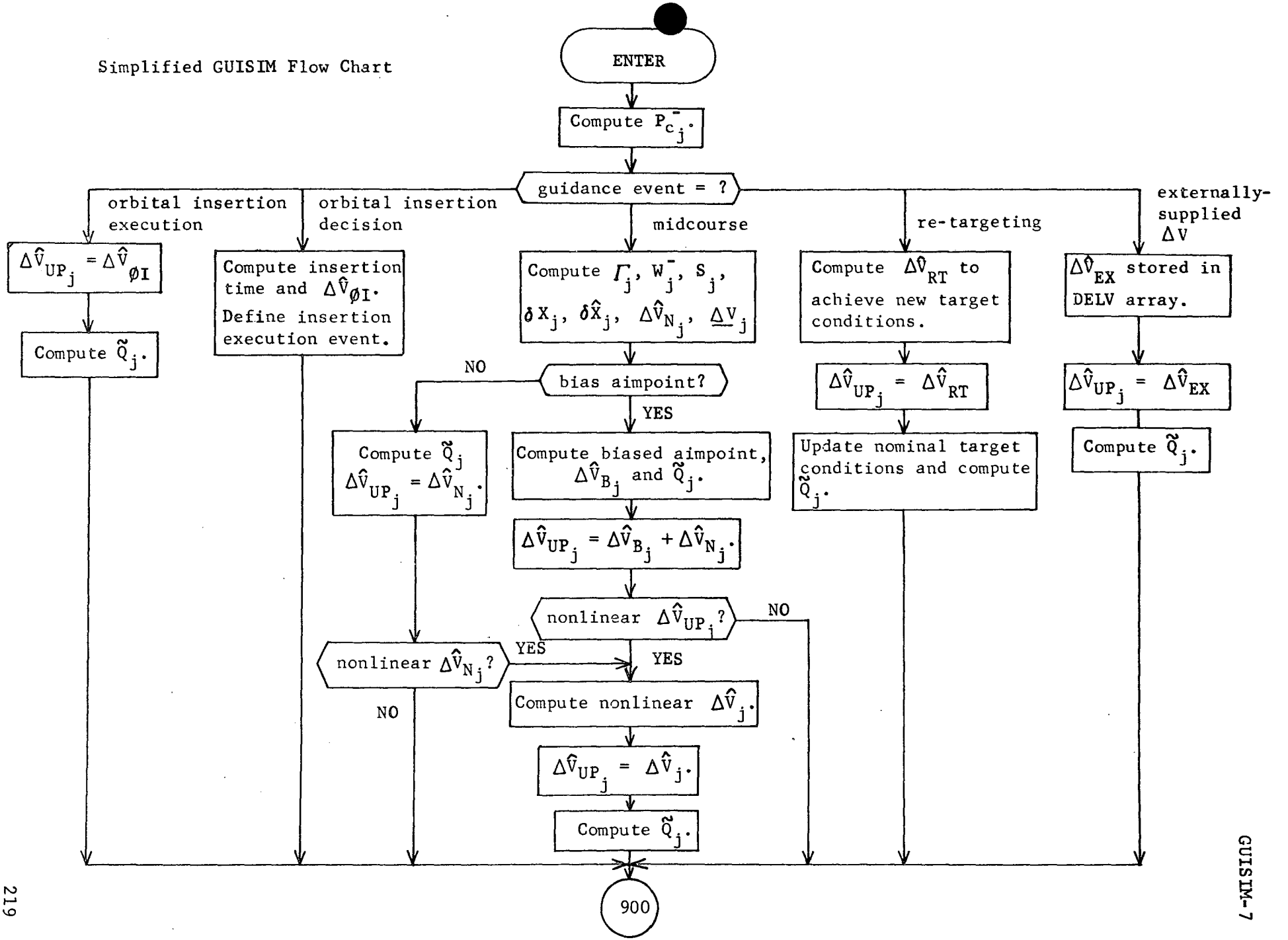
At an orbital insertion execution event the previously computed $\Delta\hat{V}_{\emptyset I}$ is used to update the targeted nominal state. In addition, the planeto-centric equatorial components of $\Delta\hat{V}_{\emptyset I}$ and the actual spacecraft cartesian and orbital element states following the insertion maneuver are computed.

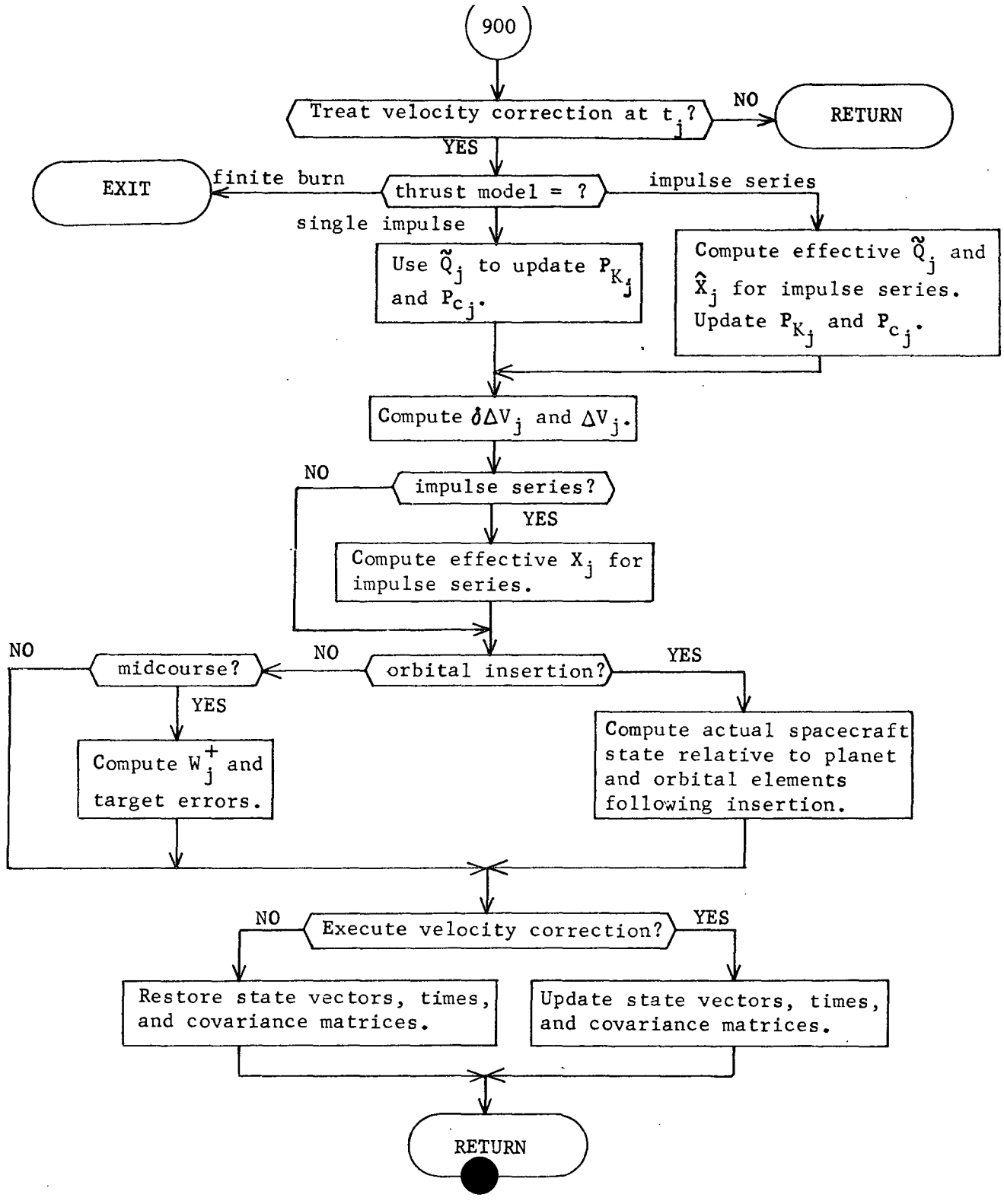
4. Externally-supplied velocity correction.

At this type of guidance event the state estimate $\tilde{X}_j^- + \hat{\tilde{X}}_j^-$ is simply updated using the externally-supplied velocity correction $\Delta\hat{V}_{EX}$.

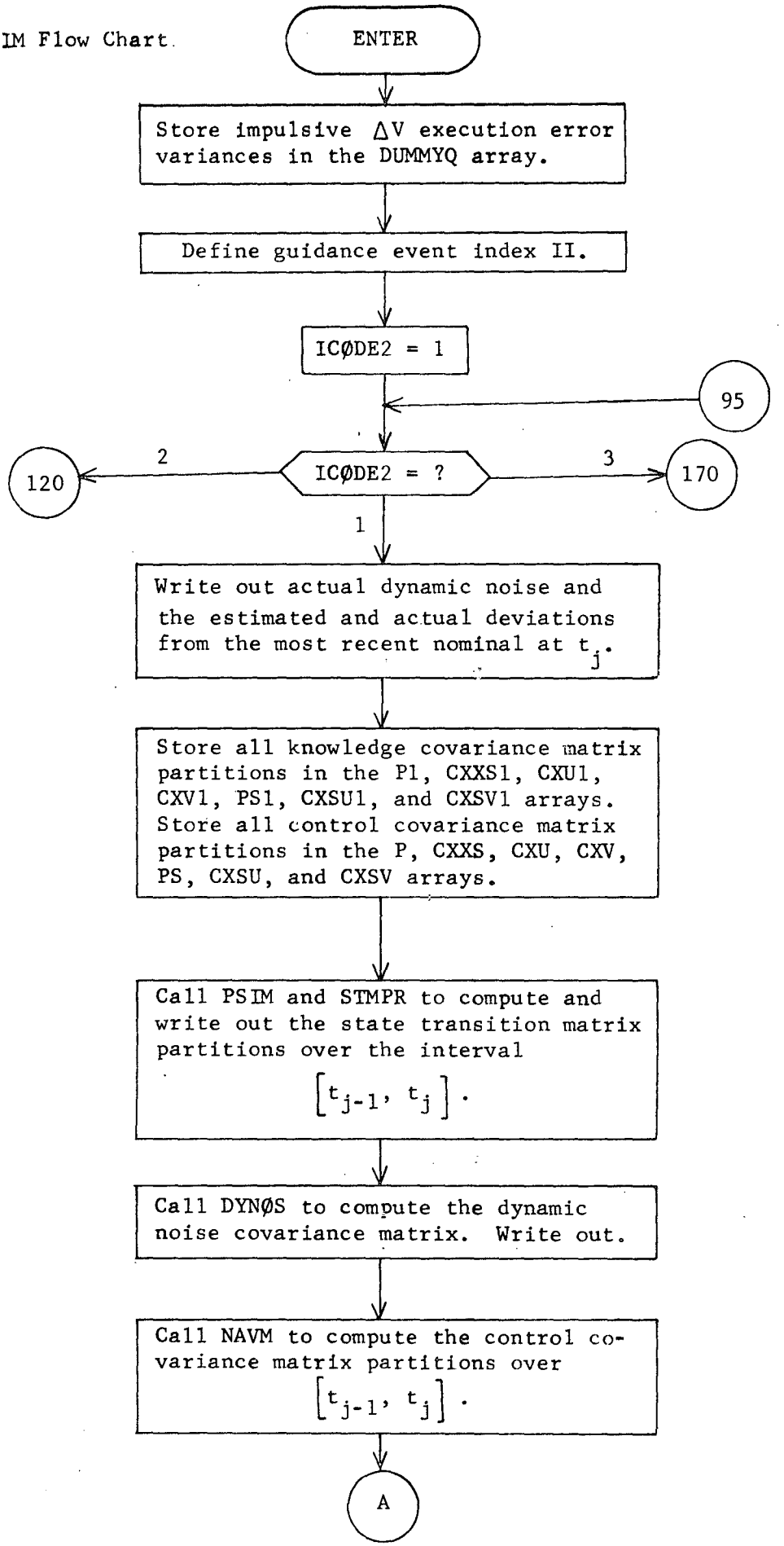
Because of the complexity of the GUISIM flow chart, a simplified flow chart depicting the main elements of the GUISIM structure precedes the complete GUISIM flow chart.

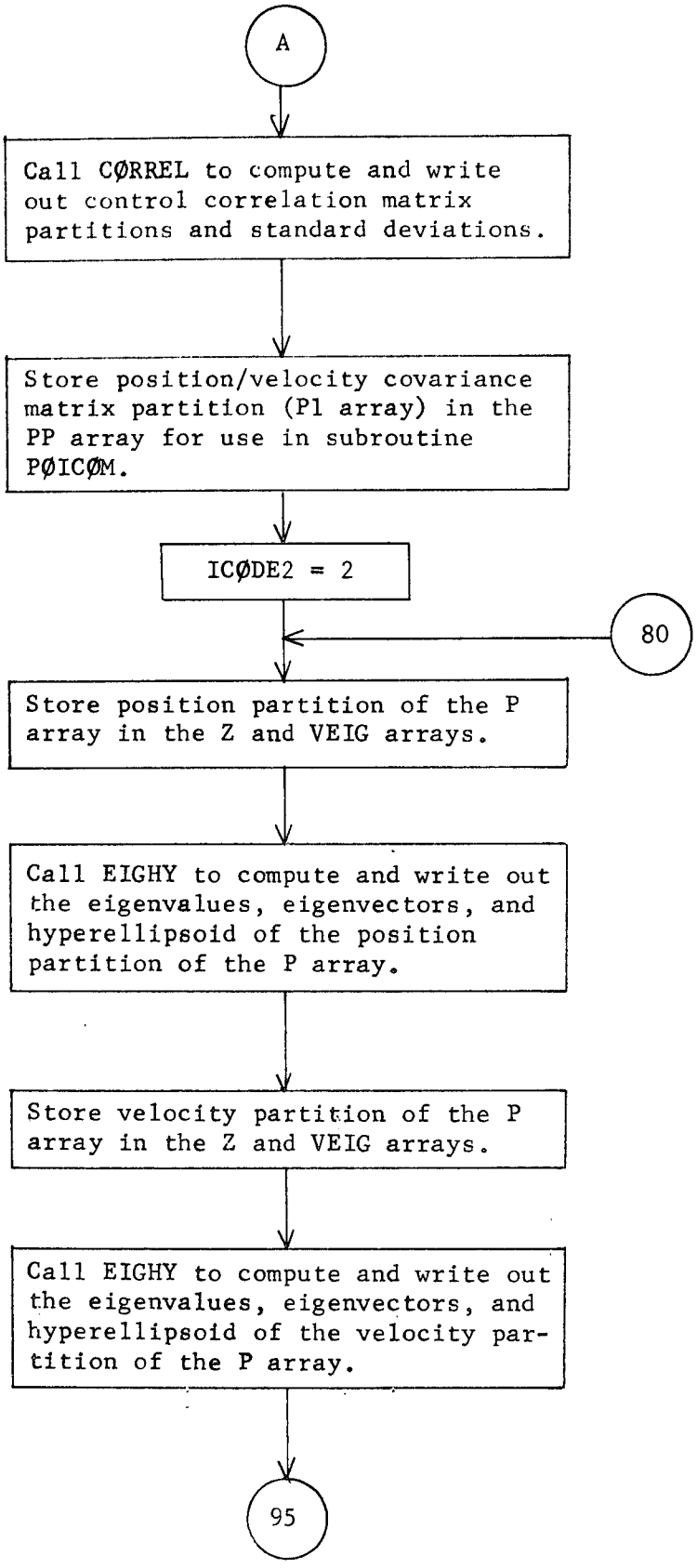
Simplified GUISIM Flow Chart

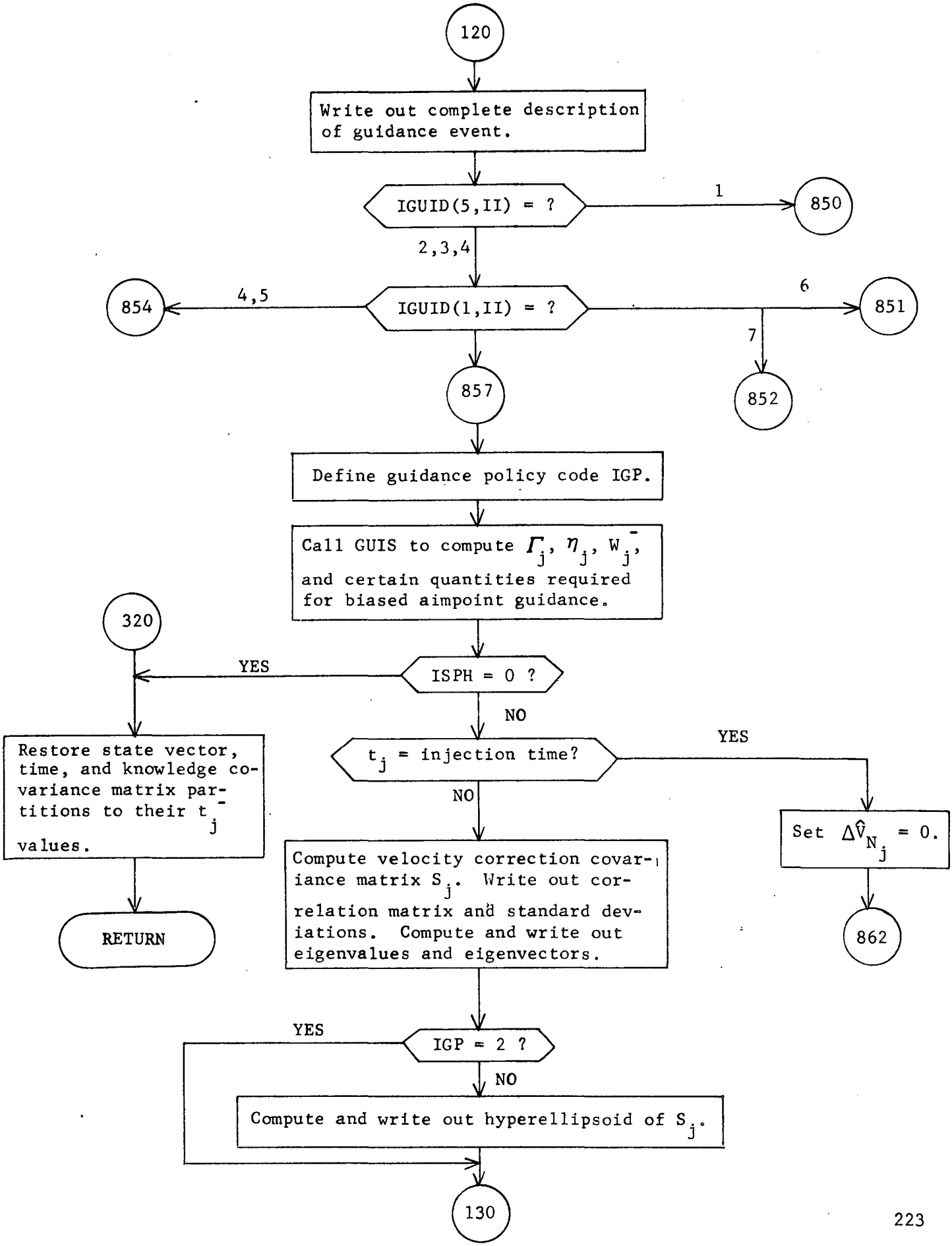


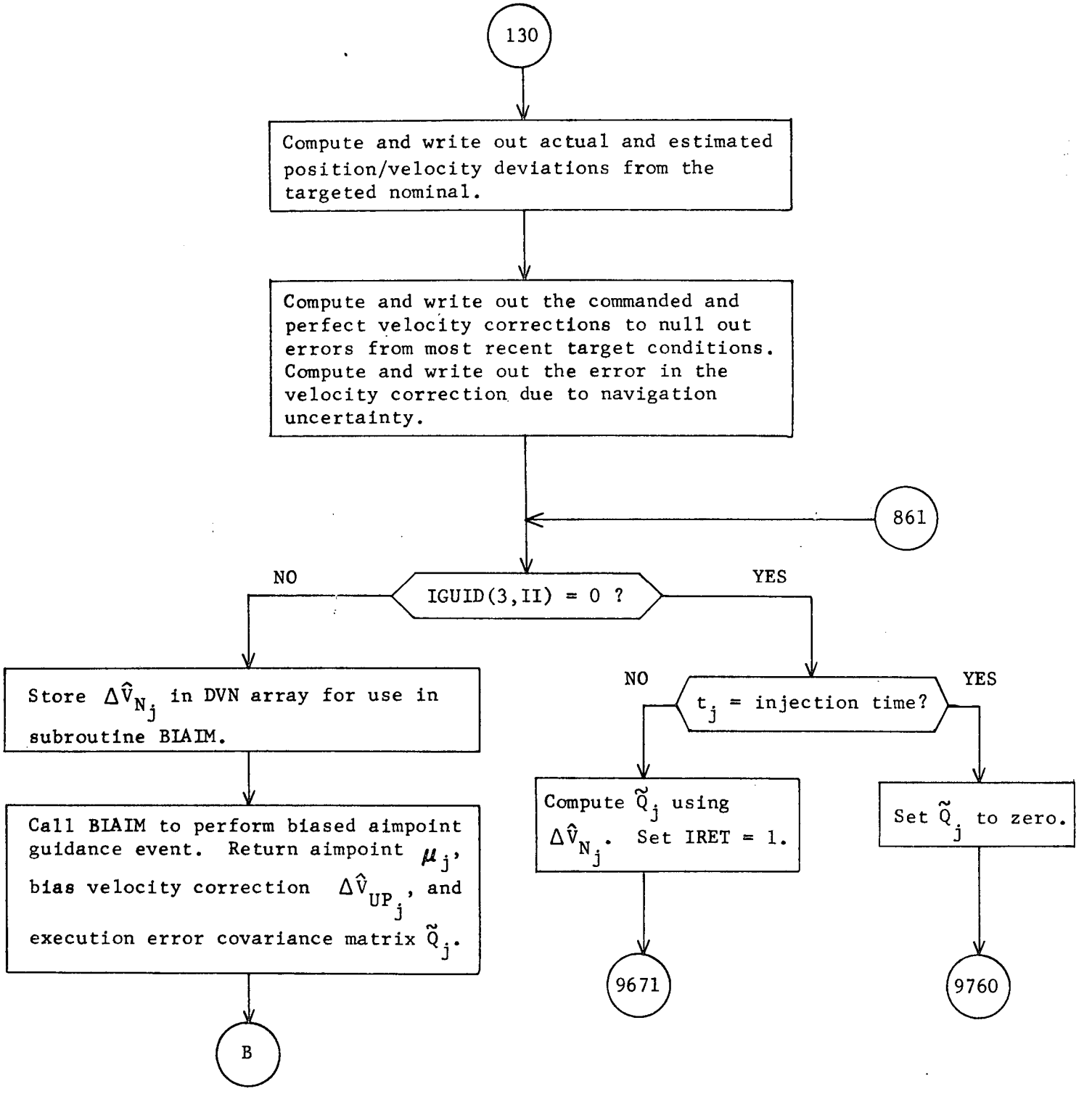

900
Treat velocity correction at t_j?
NO
RETURN
YES
EXIT
finite burn
thrust model = ?
impulse series
single impulse
Use $\tilde{Q}_j$ to update P_{K_j} and P_{c_j}.
Compute effective $\tilde{Q}_j$ and $\hat{X}_j$ for impulse series. Update P_{K_j} and P_{c_j}.
Compute $\delta\Delta V_j$ and ΔV_j.
NO
impulse series?
YES
Compute effective X_j for impulse series.
NO
midcourse?
NO
orbital insertion?
YES
YES
Compute W_j^+ and target errors.
Compute actual spacecraft state relative to planet and orbital elements following insertion.
NO
Execute velocity correction?
YES
Restore state vectors, times, and covariance matrices.
Update state vectors, times, and covariance matrices.
RETURN

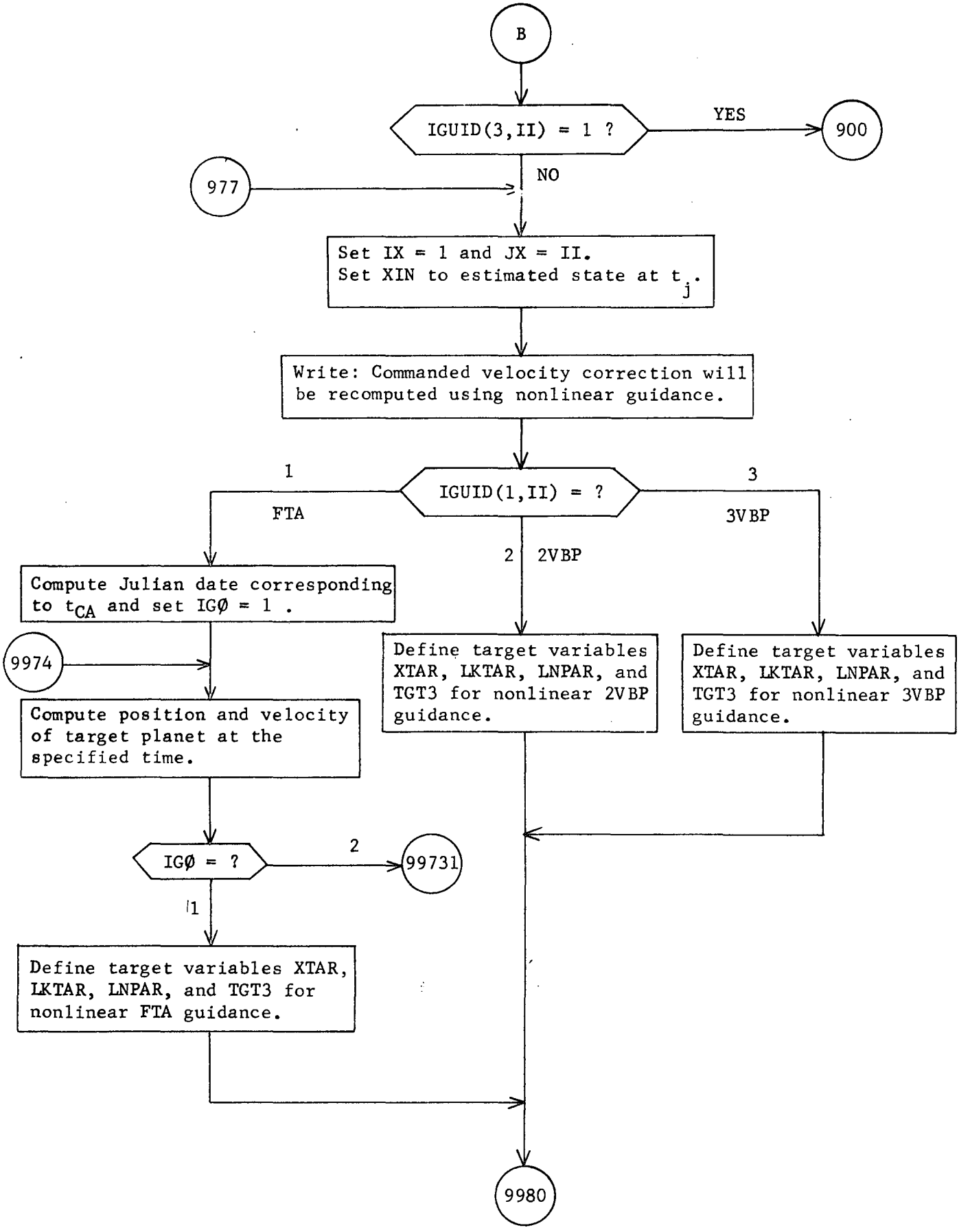
GUISIM Flow Chart.

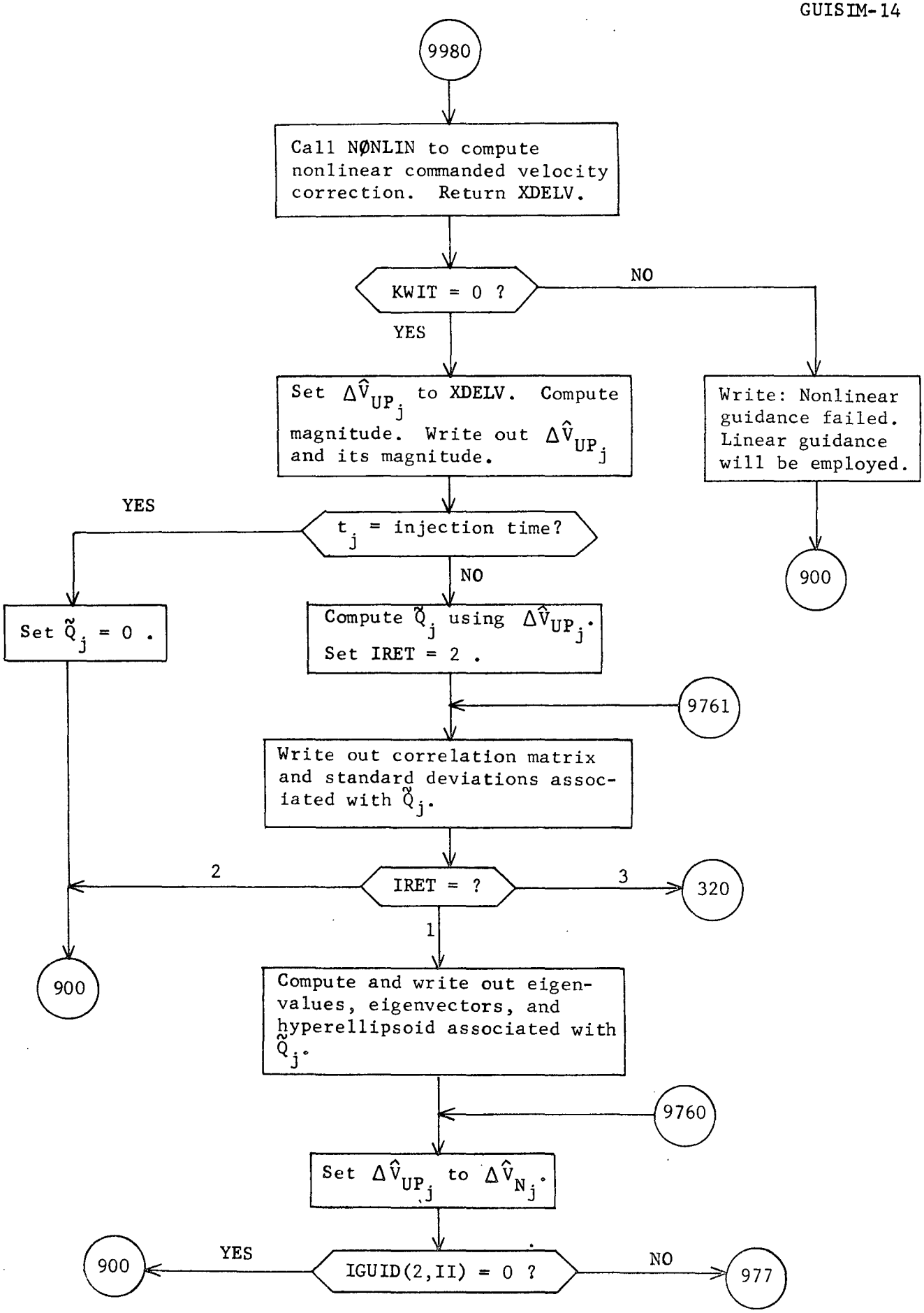


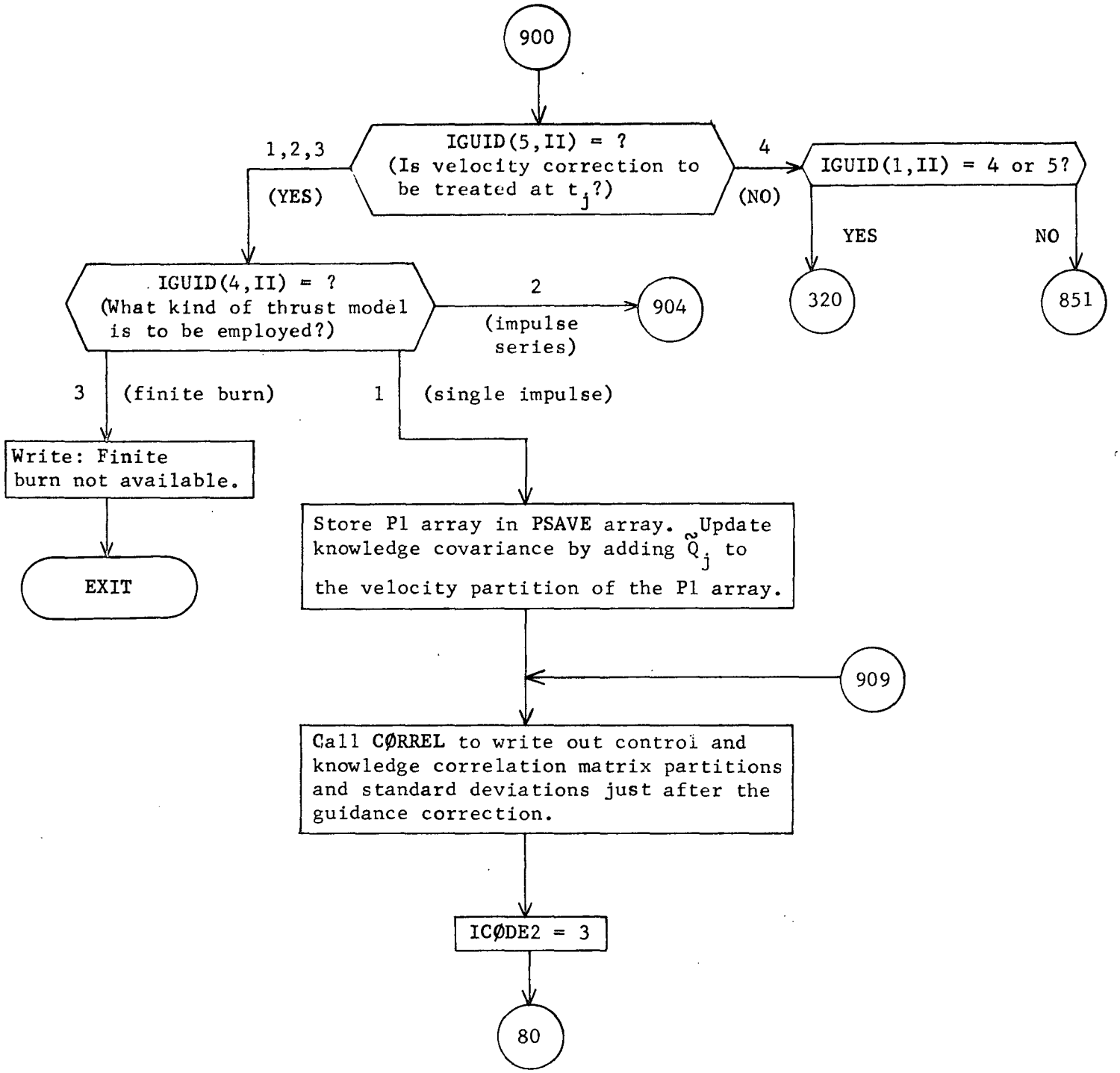

A
Call CØRREL to compute and write out control correlation matrix partitions and standard deviations.
Store position/velocity covariance matrix partition (P1 array) in the PP array for use in subroutine PØICØM.
ICØDE2 = 2
80
Store position partition of the P array in the Z and VEIG arrays.
Call EIGHY to compute and write out the eigenvalues, eigenvectors, and hyperellipsoid of the position partition of the P array.
Store velocity partition of the P array in the Z and VEIG arrays.
Call EIGHY to compute and write out the eigenvalues, eigenvectors, and hyperellipsoid of the velocity partition of the P array.
95

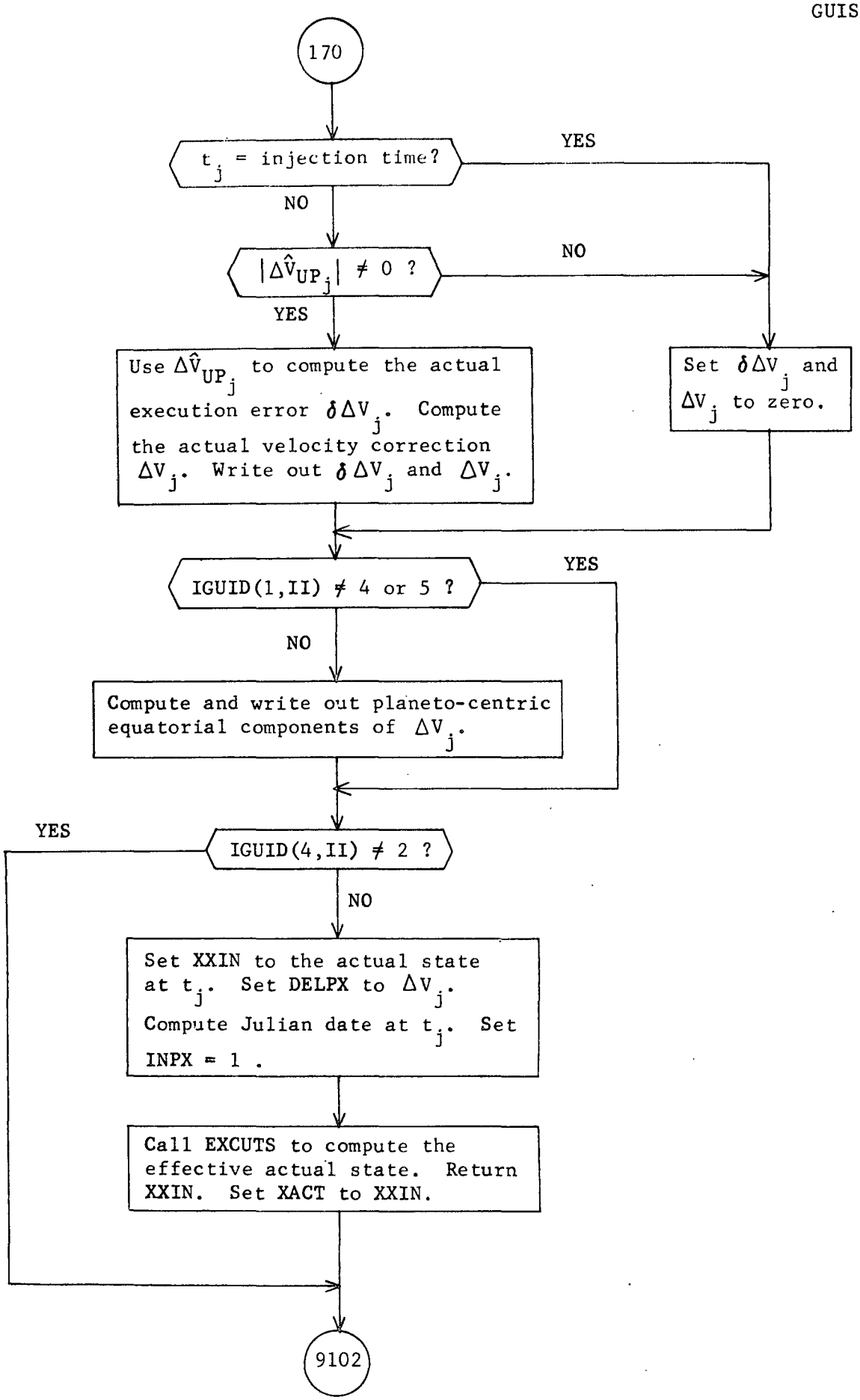

120
Write out complete description of guidance event.
IGUID(5,II) = ?
1
850
2,3,4
854
4,5
IGUID(1,II) = ?
6
851
7
852
857
Define guidance policy code IGP.
Call GUIS to compute Γ_j, η_j, $\bar{W}_j$, and certain quantities required for biased aimpoint guidance.
320
YES
ISPH = 0 ?
NO
Restore state vector, time, and knowledge covariance matrix partitions to their $\bar{t}_j$ values.
t_j = injection time?
YES
NO
Set $\Delta\hat{V}_{N_j} = 0.$
RETURN
Compute velocity correction covariance matrix S_j. Write out correlation matrix and standard deviations. Compute and write out eigenvalues and eigenvectors.
862
YES
IGP = 2 ?
NO
Compute and write out hyperellipsoid of S_j.
130

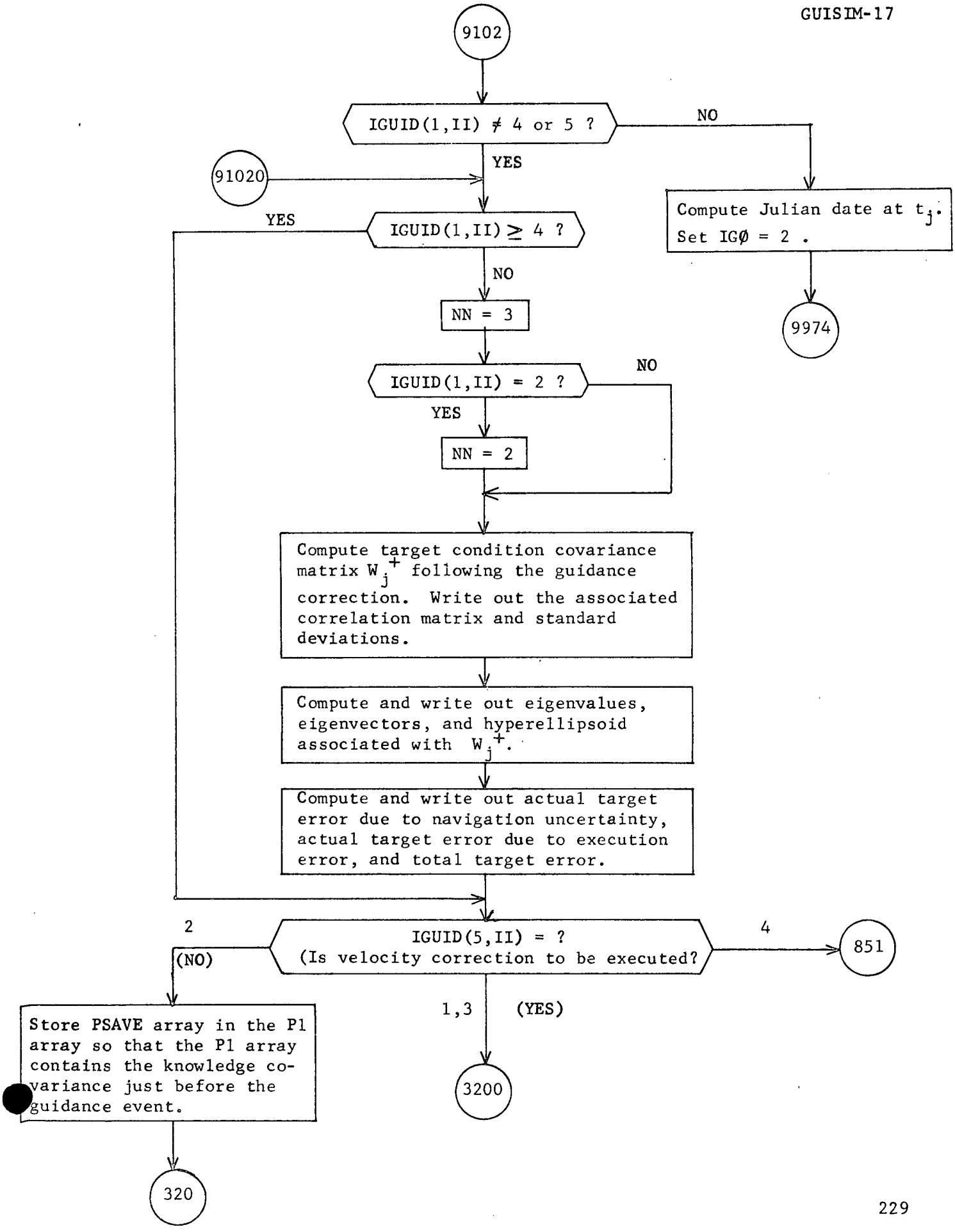

130
Compute and write out actual and estimated position/velocity deviations from the targeted nominal.
Compute and write out the commanded and perfect velocity corrections to null out errors from most recent target conditions. Compute and write out the error in the velocity correction due to navigation uncertainty.
861
NO
IGUID(3,II) = 0 ?
YES
Store $\Delta\hat{V}_{N_j}$ in DVN array for use in subroutine BIAIM.
Call BIAIM to perform biased aimpoint guidance event. Return aimpoint μ_j, bias velocity correction $\Delta\hat{V}_{UP_j}$, and execution error covariance matrix $\tilde{Q}_j$.
B
NO
t_j = injection time?
YES
Compute $\tilde{Q}_j$ using $\Delta\hat{V}_{N_j}$. Set IRET = 1.
Set $\tilde{Q}_j$ to zero.
9671
9760

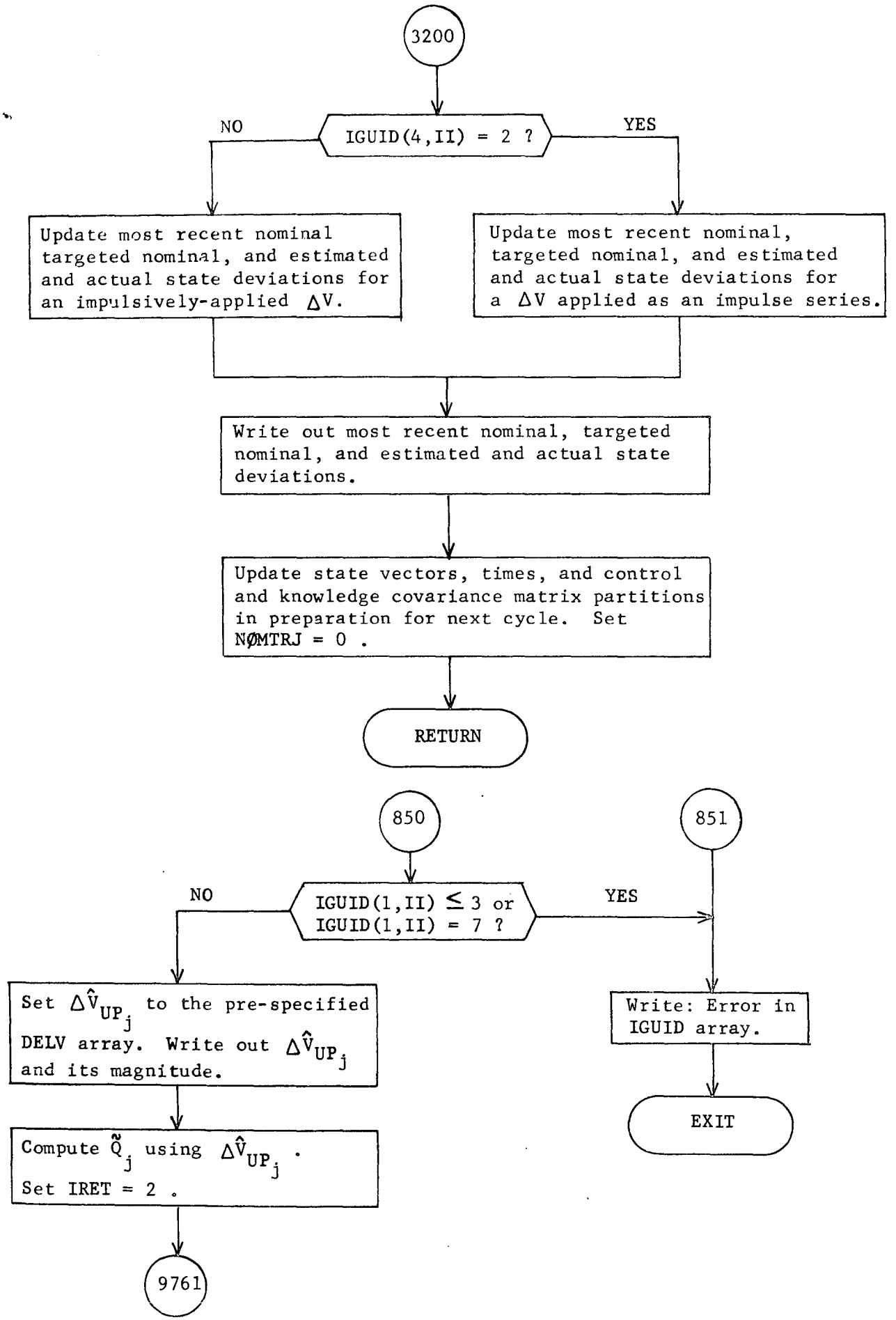

B
IGUID(3,II) = 1 ?
YES
900
977
NO
Set IX = 1 and JX = II.
Set XIN to estimated state at t_j.
Write: Commanded velocity correction will be recomputed using nonlinear guidance.
1
IGUID(1,II) = ?
3
FTA
3VBP
2
2VBP
Compute Julian date corresponding to t_{CA} and set IGØ = 1 .
9974
Define target variables XTAR, LKTAR, LNPAR, and TGT3 for nonlinear 2VBP guidance.
Define target variables XTAR, LKTAR, LNPAR, and TGT3 for nonlinear 3VBP guidance.
Compute position and velocity of target planet at the specified time.
IGØ = ?
2
99731
1
Define target variables XTAR, LKTAR, LNPAR, and TGT3 for nonlinear FTA guidance.
9980

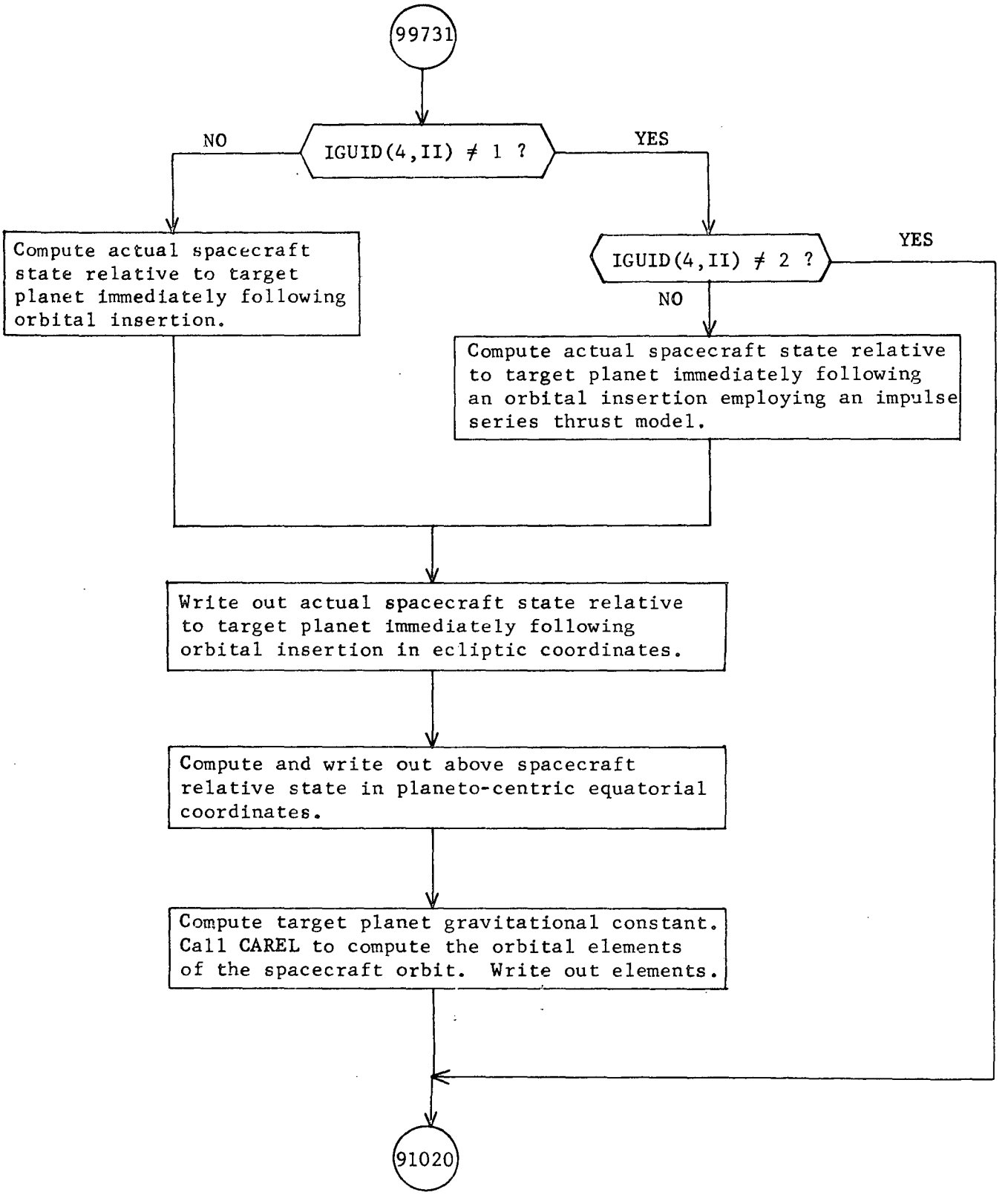

9980
Call NØNLIN to compute nonlinear commanded velocity correction. Return XDELV.
KWIT = 0 ?
NO
YES
Set $\Delta\hat{V}_{UP_j}$ to XDELV. Compute magnitude. Write out $\Delta\hat{V}_{UP_j}$ and its magnitude.
Write: Nonlinear guidance failed. Linear guidance will be employed.
900
YES
t_j = injection time?
NO
Set $\tilde{Q}_j = 0$.
Compute $\tilde{Q}_j$ using $\Delta\hat{V}_{UP_j}$.
Set IRET = 2 .
9761
Write out correlation matrix and standard deviations associated with $\tilde{Q}_j$.
2
IRET = ?
3
320
1
900
Compute and write out eigenvalues, eigenvectors, and hyperellipsoid associated with $\tilde{Q}_j$.
9760
Set $\Delta\hat{V}_{UP_j}$ to $\Delta\hat{V}_{N_j}$.
900
YES
IGUID(2,II) = 0 ?
NO
977

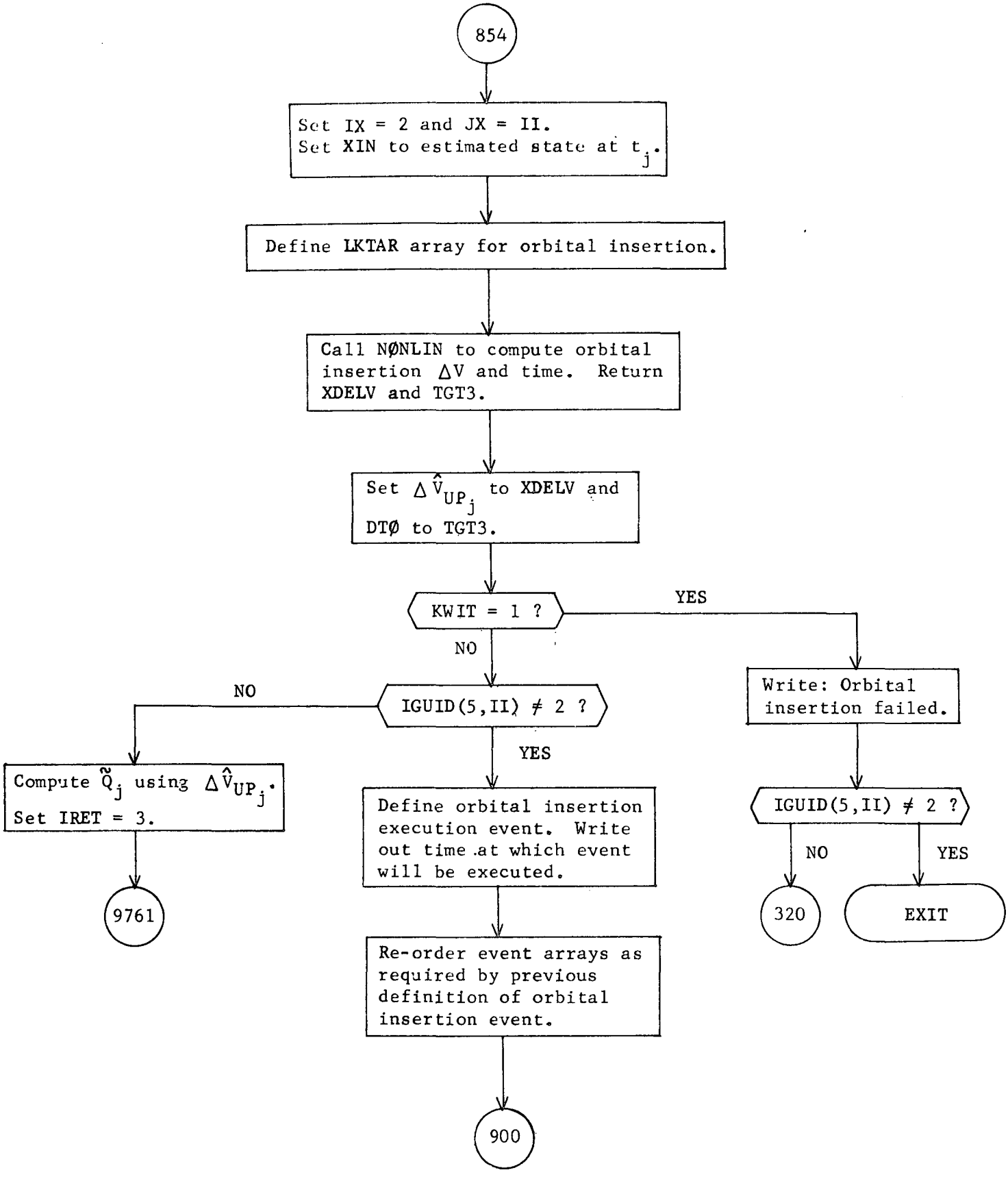

900
IGUID(5,II) = ?
(Is velocity correction to be treated at t_j?)
1,2,3
(YES)
4
(NO)
IGUID(1,II) = 4 or 5?
YES
NO
320
851
IGUID(4,II) = ?
(What kind of thrust model is to be employed?)
2
(impulse series)
904
3
(finite burn)
1
(single impulse)
Write: Finite burn not available.
EXIT
Store P1 array in PSAVE array. Update knowledge covariance by adding $\tilde{Q}_j$ to the velocity partition of the P1 array.
909
Call CØRREL to write out control and knowledge correlation matrix partitions and standard deviations just after the guidance correction.
ICØDE2 = 3
80

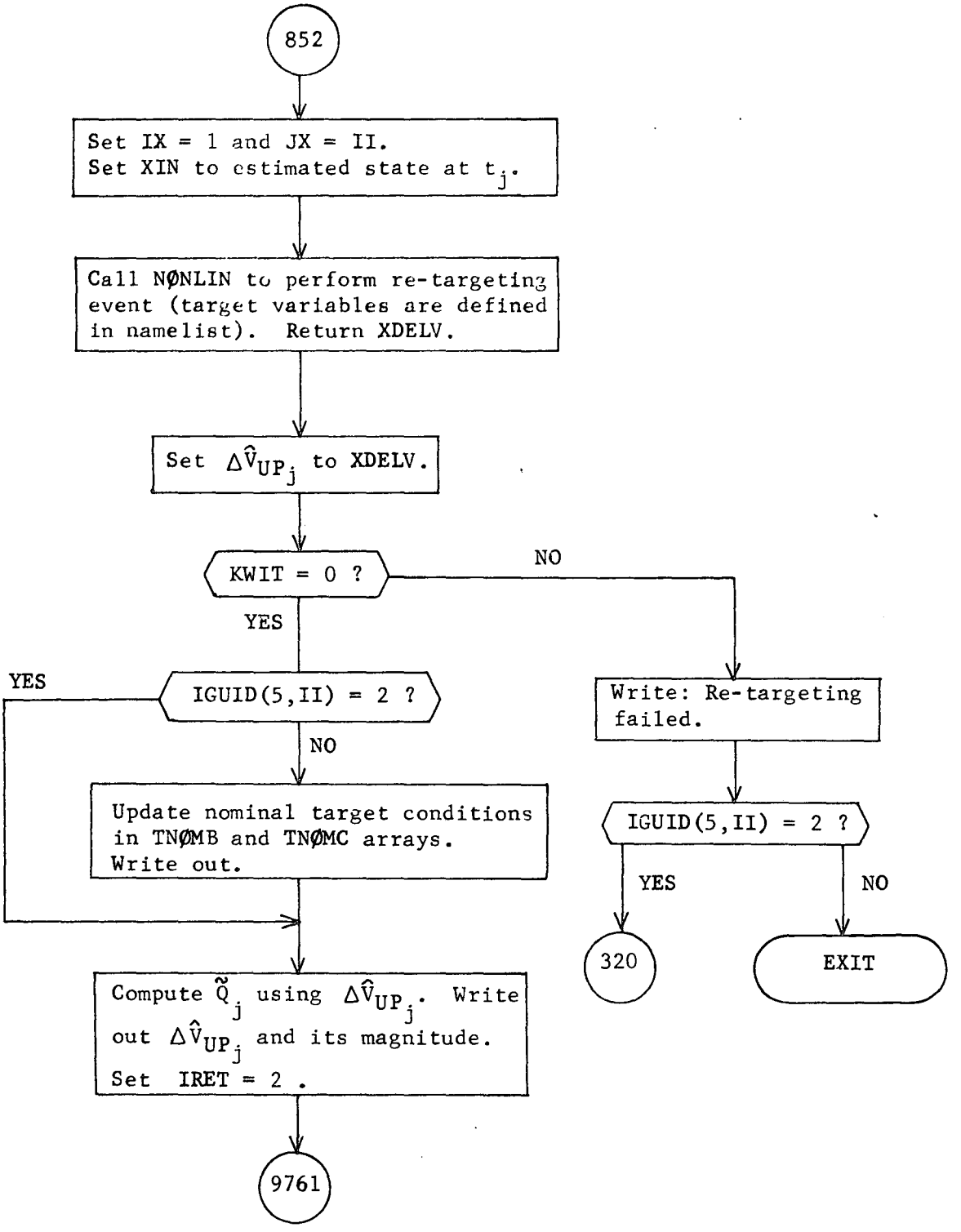

170
t_j = injection time?
YES
NO
$|\Delta\hat{V}_{UP_j}| \neq 0$?
NO
YES
Use $\Delta\hat{V}_{UP_j}$ to compute the actual execution error $\delta\Delta V_j$. Compute the actual velocity correction ΔV_j. Write out $\delta\Delta V_j$ and ΔV_j.
Set $\delta\Delta V_j$ and ΔV_j to zero.
IGUID(1,II) ≠ 4 or 5 ?
YES
NO
Compute and write out planeto-centric equatorial components of ΔV_j.
YES
IGUID(4,II) ≠ 2 ?
NO
Set XXIN to the actual state at t_j. Set DELPX to ΔV_j. Compute Julian date at t_j. Set INPX = 1 .
Call EXCUTS to compute the effective actual state. Return XXIN. Set XACT to XXIN.
9102

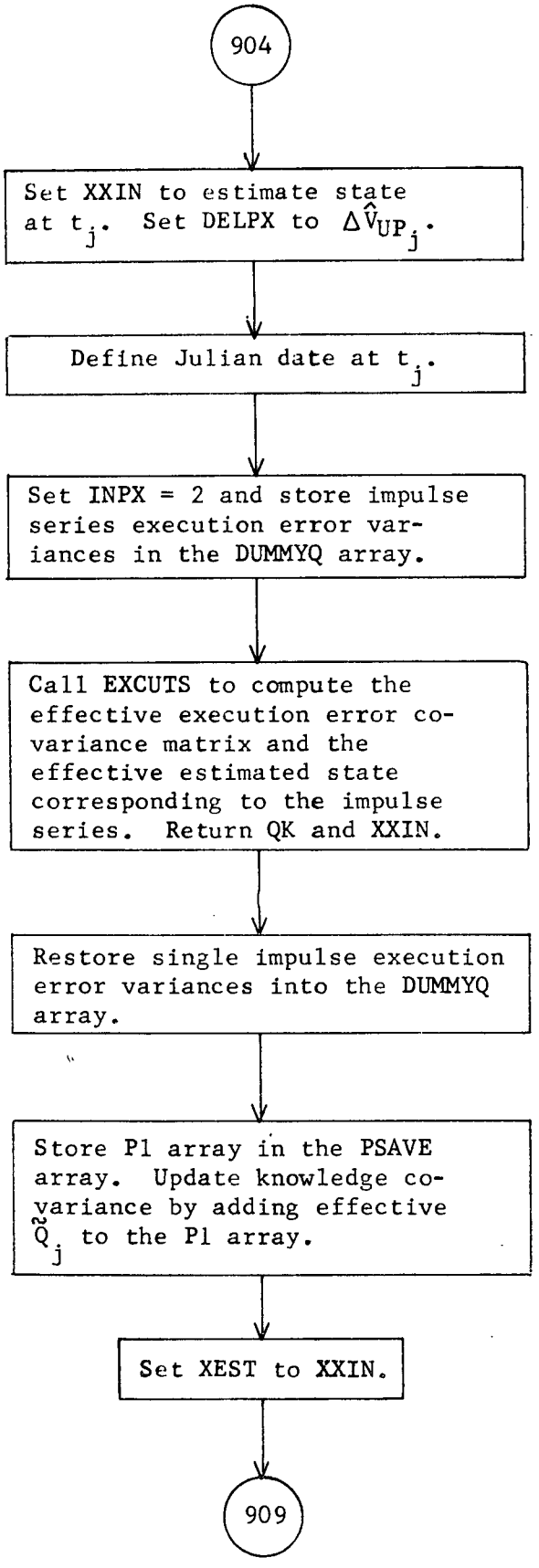

9102
IGUID(1,II) ≠ 4 or 5 ?
NO
YES
91020
Compute Julian date at t_j.
Set IGØ = 2 .
9974
YES
IGUID(1,II) ≥ 4 ?
NO
NN = 3
IGUID(1,II) = 2 ?
NO
YES
NN = 2
Compute target condition covariance matrix W_j^+ following the guidance correction. Write out the associated correlation matrix and standard deviations.
Compute and write out eigenvalues, eigenvectors, and hyperellipsoid associated with W_j^+.
Compute and write out actual target error due to navigation uncertainty, actual target error due to execution error, and total target error.
2
IGUID(5,II) = ?
(Is velocity correction to be executed?
4
851
(NO)
1,3
(YES)
Store PSAVE array in the P1 array so that the P1 array contains the knowledge covariance just before the guidance event.
3200
320


3200
IGUID(4,II) = 2 ?
NO
YES
Update most recent nominal targeted nominal, and estimated and actual state deviations for an impulsively-applied ΔV.
Update most recent nominal, targeted nominal, and estimated and actual state deviations for a ΔV applied as an impulse series.
Write out most recent nominal, targeted nominal, and estimated and actual state deviations.
Update state vectors, times, and control and knowledge covariance matrix partitions in preparation for next cycle. Set NØMTRJ = 0 .
RETURN
850
851
IGUID(1,II) ≤ 3 or IGUID(1,II) = 7 ?
NO
YES
Set ΔV̂UP_j to the pre-specified DELV array. Write out ΔV̂UP_j and its magnitude.
Write: Error in IGUID array.
Compute Q̃_j using ΔV̂UP_j .
Set IRET = 2 .
EXIT
9761


99731
IGUID(4,II) ≠ 1 ?
NO
YES
Compute actual spacecraft state relative to target planet immediately following orbital insertion.
IGUID(4,II) ≠ 2 ?
YES
NO
Compute actual spacecraft state relative to target planet immediately following an orbital insertion employing an impulse series thrust model.
Write out actual spacecraft state relative to target planet immediately following orbital insertion in ecliptic coordinates.
Compute and write out above spacecraft relative state in planeto-centric equatorial coordinates.
Compute target planet gravitational constant. Call CAREL to compute the orbital elements of the spacecraft orbit. Write out elements.
91020


854
Set IX = 2 and JX = II.
Set XIN to estimated state at t_j.
Define LKTAR array for orbital insertion.
Call NØNLIN to compute orbital insertion ΔV and time. Return XDELV and TGT3.
Set $\Delta \hat{V}_{UP_j}$ to XDELV and DTØ to TGT3.
KWIT = 1 ?
YES
NO
Write: Orbital insertion failed.
IGUID(5,II) ≠ 2 ?
NO
YES
Compute $\tilde{Q}_j$ using $\Delta \hat{V}_{UP_j}$.
Set IRET = 3.
Define orbital insertion execution event. Write out time at which event will be executed.
IGUID(5,II) ≠ 2 ?
NO
YES
9761
Re-order event arrays as required by previous definition of orbital insertion event.
320
EXIT
900


852
Set IX = 1 and JX = II.
Set XIN to estimated state at t_j.
Call NØNLIN to perform re-targeting event (target variables are defined in namelist). Return XDELV.
Set $\Delta\hat{V}_{UP_j}$ to XDELV.
KWIT = 0 ?
NO
YES
YES
IGUID(5,II) = 2 ?
NO
Update nominal target conditions in TNØMB and TNØMC arrays. Write out.
Write: Re-targeting failed.
IGUID(5,II) = 2 ?
YES
NO
320
EXIT
Compute $\tilde{Q}_j$ using $\Delta\hat{V}_{UP_j}$. Write out $\Delta\hat{V}_{UP_j}$ and its magnitude.
Set IRET = 2 .
9761


904
Set XXIN to estimate state at t_j. Set DELPX to $\Delta\hat{V}_{UP_j}$.
Define Julian date at t_j.
Set INPX = 2 and store impulse series execution error variances in the DUMMYQ array.
Call EXCUTS to compute the effective execution error covariance matrix and the effective estimated state corresponding to the impulse series. Return QK and XXIN.
Restore single impulse execution error variances into the DUMMYQ array.
Store P1 array in the PSAVE array. Update knowledge covariance by adding effective $\tilde{Q}_j$ to the P1 array.
Set XEST to XXIN.
909

HELIO Analysis

HELIO computes the zero iterate initial state for interplanetary trajectories. The initial and final states are determined either by an arbitrary position vector or by the location of a specified planet at a prescribed time according to

IZERO = 1 planet to planet
2 planet to arbitrary final point
3 arbitrary initial point to planet
4 arbitrary initial point to final point

The final time used in locating a planet must correspond to the closest approach (CA) to the planet. Therefore if the target time is read in as a sphere of influence (SOI) time, a modification is required. The heliocentric conic is computed (as described below) using the t_{SI} time to determine the final position. The approach asymptote $\vec{v}_{HP}$ corresponding to that trajectory is used with the desired r_{CA} to compute the time from SOI to CA. If r_{CA} is not a target variable then the target values of B·T and B·R are used to estimate the r_{CA}

$$r_{CA} = -\frac{\mu}{V_{HP}} + \frac{1}{2}\sqrt{\left(\frac{2\mu}{V_{HP}^2}\right)^2 + 4B^2} \qquad (1)$$

Then the approximate approach hyperbola is given by

$$a_h = \frac{\mu r_{SI}}{2\mu - V_{HP}^2 r_{SI}}$$

$$e_h = 1 - \frac{r_{CA}}{a} \qquad (2)$$

$$P_h = a_n(1 - e_h^2)$$

and the hyperbolic time to go from SOI to CA is given by

$$\Delta t_{SICA} = \frac{\mu}{V_{HP}^3}\ (e \sinh F - F) \qquad (3)$$

where

$$\tanh \frac{F}{2} = \sqrt{\frac{e_n - 1}{e_h + 1}} \tanh \frac{f}{2}$$

$$\cos f = \frac{1}{e}\left(\frac{P_h}{r_{SI}} - 1\right) \tag{4}$$

The final time is then given by $t_f = t_{SI} + \Delta t_{SICA}$.

The initial and final positions $\vec{r}_i$ and $\vec{r}_f$ of the heliocentric conic are either input or computed from the positions of planets determined by ORB and EPHEM. The unit normal to the heliocentric orbit plane is

$$\hat{W} = \frac{\vec{r}_i \times \vec{r}_f}{\left|\vec{r}_i \times \vec{r}_f\right|} \tag{5}$$

The inclination to that plane is

$$\cos i = \hat{W}_z \tag{6}$$

The ascending node of the plane is given by

$$\tan \Omega = \frac{\hat{W}_x}{-\hat{W}_y} \tag{7}$$

The central angle of transfer is defined by

$$\cos \Psi = \frac{\vec{r}_i \cdot \vec{r}_f}{r_i r_f} \tag{8}$$

The semi-major axis a and eccentricity e of the heliocentric conic are computed from Lambert's theorem in subroutine FLITE. The true anomaly f_i at the initial and final points are computed from

$$p = a\,(1 - e^2)$$

$$\cos f_i = \frac{p - r_i}{e\, r_i} \qquad \sin f_i = \frac{\cos f_i \cos \Psi - \frac{p - r_f}{e\, r_f}}{\sin \Psi} \tag{8}$$

$$f_f = f_i + \Psi$$

Finally, the argument of periapsis ω is computed from

$$\cos(\omega + f_i) = \frac{\vec{r}_i \cdot \hat{U}}{r_i} \tag{10}$$

where $\hat{U} = (\cos \Omega , \sin \Omega , 0)$.

Therefore the initial or final states ($\vec{r}_i$, $\vec{v}_i$) or ($\vec{r}_f$, $\vec{v}_f$) may now be computed by ELCAR. Let ($\vec{r}$, $\vec{v}$) denote either state and let ($\vec{r}$, $\vec{v}_p$) denote the state of the relevant planet. The departure (or approach) asymptote is then given by

$$\vec{v}_{HP} = \vec{v}_f - \vec{v}_p \qquad \vec{v}_{HE} = \vec{v}_i - \vec{v}_p \tag{11}$$

The latitude and longitude of the position vector are

$$\sin \phi = \frac{\vec{r}}{r} \qquad \tan \theta = \frac{r_y}{r_x} \tag{12}$$

The path angle Γ may be computed from

$$\cos \Gamma = \frac{\sqrt{\mu p}}{r v} \tag{13}$$

The azimuth of the relevant state is

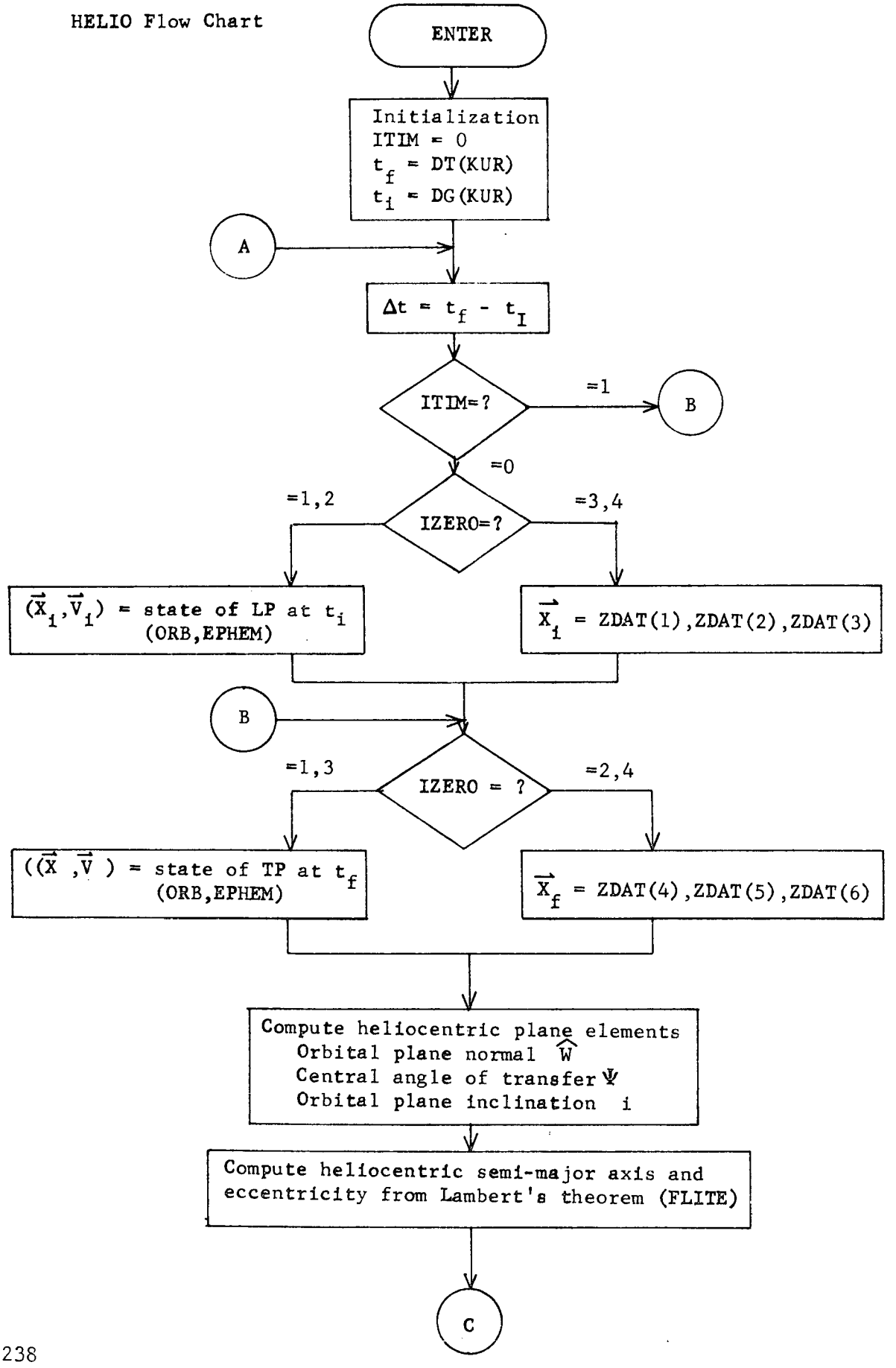
$$\sin \Sigma = \frac{(\vec{r} \times \vec{v}) \cdot \hat{U}}{|\vec{r} \times \vec{v}|} \tag{14}$$

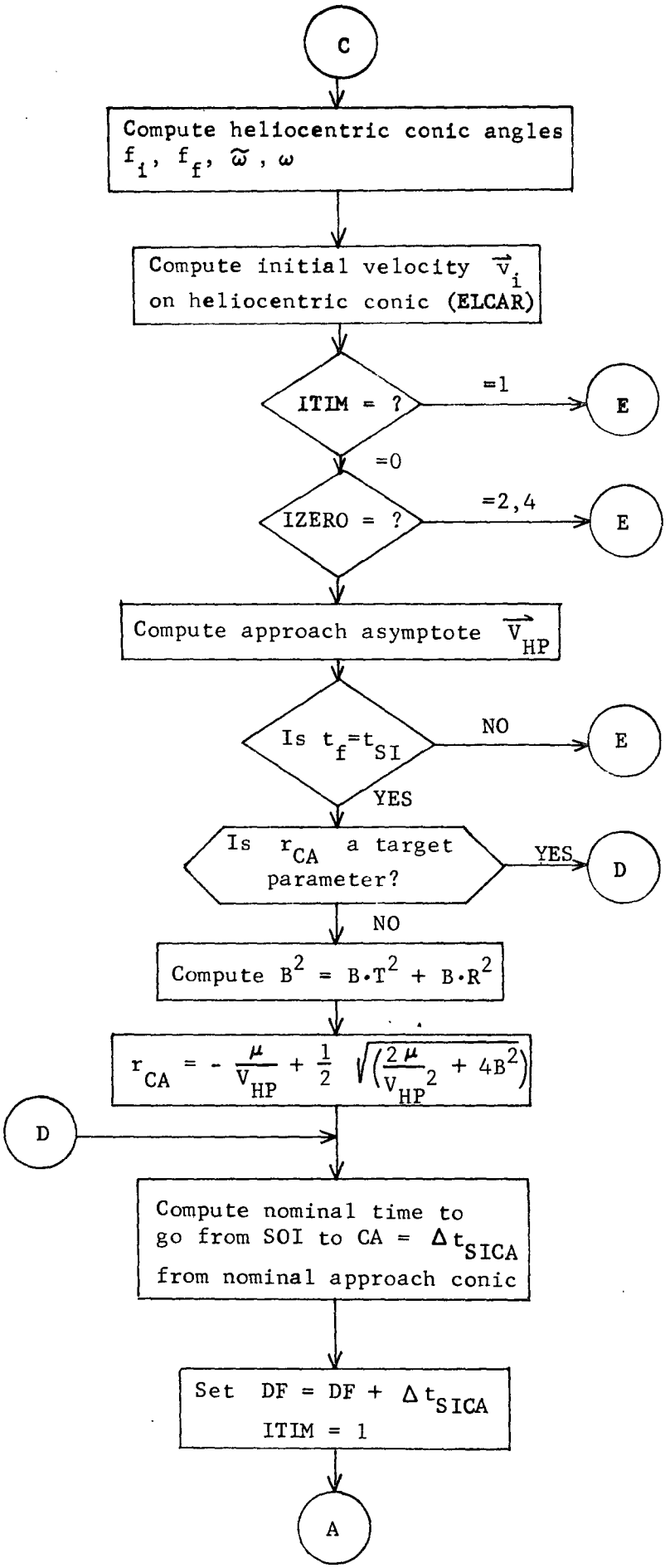
$$\cos \Sigma = \frac{\vec{V} \cdot \hat{U}}{V \cos \Gamma} \tag{15}$$

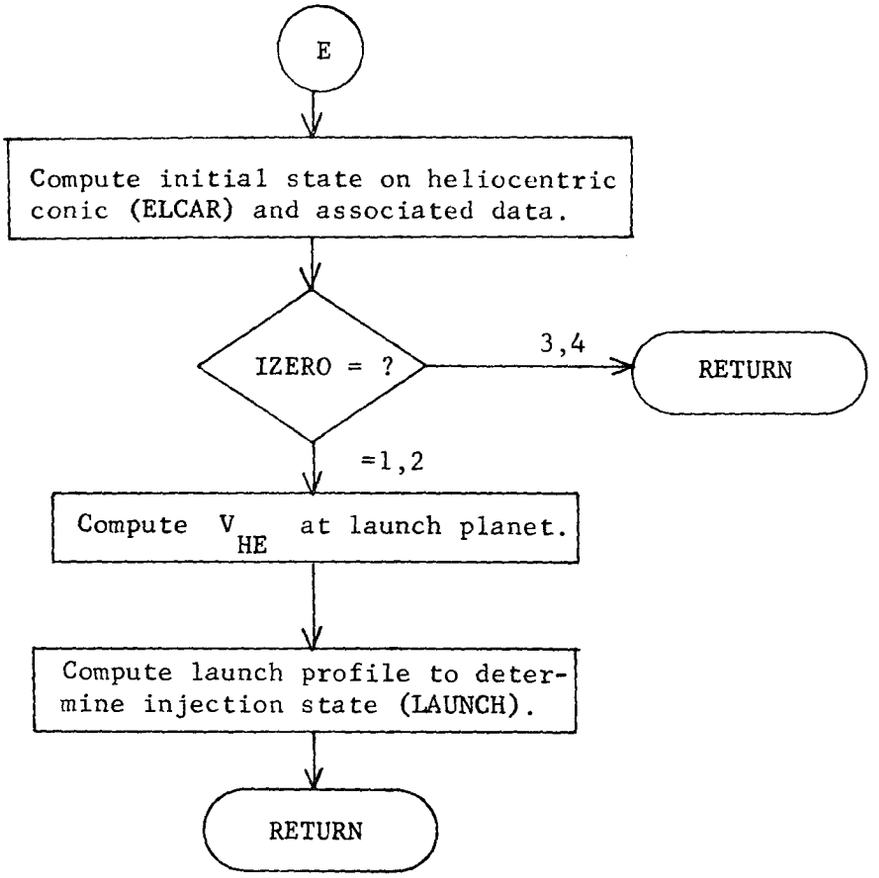
If the initial state is referenced to a planet, subroutine LAUNCH is called to convert the departure asymptote and launch profile into an injection radius, velocity, and time. Otherwise the initial state is returned as the initial state on the heliocentric conic.

Reference: Space Research Conic Program, Phase III, May 1, 1969, Jet Propulsion Laboratory, Pasadena, California.

HELIO Flow Chart




C
Compute heliocentric conic angles f_i, f_f, $\tilde{\omega}$, ω
Compute initial velocity $\vec{v}_i$ on heliocentric conic (ELCAR)
ITIM = ?
=1
E
=0
IZERO = ?
=2,4
E
Compute approach asymptote $\vec{V}_{HP}$
Is $t_f = t_{SI}$
NO
E
YES
Is r_{CA} a target parameter?
YES
D
NO
Compute $B^2 = B \cdot T^2 + B \cdot R^2$
$r_{CA} = -\frac{\mu}{V_{HP}} + \frac{1}{2}\sqrt{\left(\frac{2\mu}{V_{HP}^2} + 4B^2\right)}$
D
Compute nominal time to go from SOI to CA = Δt_{SICA} from nominal approach conic
Set DF = DF + Δt_{SICA}
ITIM = 1
A


E
Compute initial state on heliocentric conic (ELCAR) and associated data.
IZERO = ?
3,4
RETURN
=1,2
Compute V_{HE} at launch planet.
Compute launch profile to determine injection state (LAUNCH).
RETURN

HYELS Analysis

Subroutine HYELS computes and writes out hyperellipsoids associated with a 2 or 3 dimensional covariance matrix P.

If P is assumed to be the covariance matrix of an n-dimensional random variable $\vec{x}$ having a gaussian distribution with mean zero, then the probability density function is given by

$$p = \frac{1}{(2\pi)^{n/2} |P|^{1/2}} \exp\left[-\frac{1}{2} \vec{x}^T P^{-1} \vec{x}\right]$$

Re-writing this equation as

$$\vec{x}^T P^{-1} \vec{x} = 2 \ln\left[\frac{1}{(2\pi)^{n/2} p |P|^{1/2}}\right] = k^2$$

shows that the surface of constant probability density p is an m-dimensional ellipsoid, where m is the rank of P. The constant k can be shown to correspond to the sigma level of the ellipsoid.

For n = 3, the above equation has form

$$ax^2 + by^2 + cz^2 + d\,xy + e\,xz + f\,yz = k^2$$

where $a = a_{11}$ $\quad d = 2a_{12}$

$b = a_{22}$ $\quad e = 2a_{13}$

$c = a_{33}$ $\quad f = 2a_{23}$

and the a_{ij} are the elements of P^{-1}.

Subroutine HYELS uses this equation to compute a 3-dimensional hyperellipsoid, and sets the appropriate constants to zero to compute a 2-dimensional hyperellipsoid.

Reference: H. Sorenson. "Kalman Filtering", Advances in Control Systems, Vol. 3, C. T. Leades (Ed.). New York: Academic Press, 1966, p. 219.

IMPACT Analysis

The impact parameters $B\cdot T$ and $B\cdot R$ form a convenient set of variables for the description of the approach geometry for lunar and interplanetary missions. Let a reference cartesian coordinate system XYZ (ecliptic in STEAP) be established at the center of the target body. Let $\vec{V}_\infty$ denote the hyperbolic excess velocity of the spacecraft in the XYZ system. An auxiliary coordinate system R-S-T may be constructed relative to the $\vec{V}_\infty$ by the definitions

$$\hat{S} = \vec{V}_\infty / \vec{v}_\infty \qquad \hat{T} = \frac{\hat{S} \times \hat{K}}{|\hat{S} \times \hat{K}|} \qquad \hat{R} = \hat{S} \times \hat{T} \qquad (1)$$

Therefore $\hat{S}$ is in the direction of the approach asymptote, $\hat{T}$ lies along the intersection of the impact plane (the plane normal to $\hat{S}$ and passing through the center of the planet) and the reference plane (XY-plane), and R completes the right hand system. The $\vec{B}$ vector lies in the impact plane and is directed to the incoming asymptote. Then $B\cdot T$ and $B\cdot R$ have the usual vector definitions.

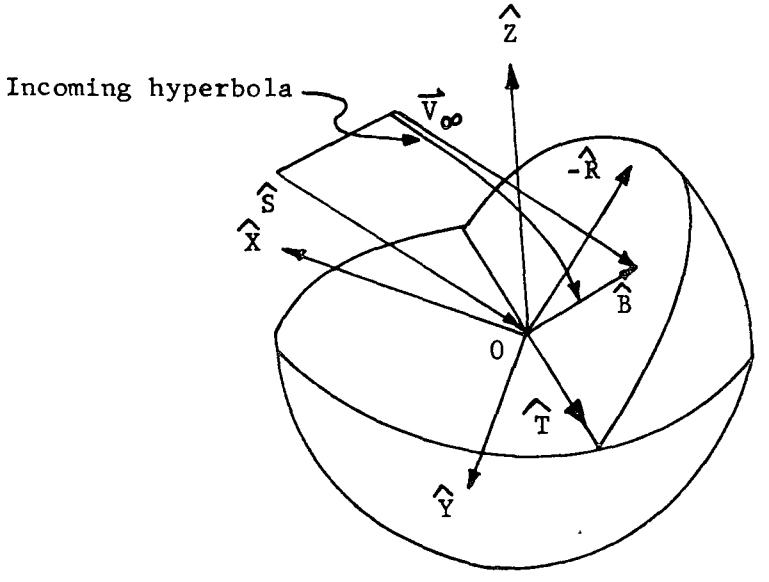


Figure 1. Impact Plane Parameters

In the optional part of the subroutine, the target impact parameter $\vec{B}^*$ associated with $\hat{S}$ and a target inclination i (relative to target planet equator) and radius of closest approach r_{CA} is computed. However given an approach asymptote $\hat{S}$ there are generally four trajectories with the same values of i and r_{CA}. Two of these trajectories are retrograde and

two are posigrade. For each type of motion there are two distinct planes that have the same inclination and include the $\hat{S}$ vector. These are distinguished by the direction of motion when the approach asymptote is crossed, i.e., whether the motion is from north to south (northern approach) or from south to north (southern approach). Let $0 \leq \alpha \leq 90^{\circ}$. Then setting the target inclinations to the following values determines the trajectory which will be specified:

i	Trajectory
α	posigrade with northern approach
$-\alpha$	posigrade with southern approach
$180+\alpha$	retrograde with northern approach
$180-\alpha$	retrograde with southern approach

The possible trajectories are illustrated in Figure 2.

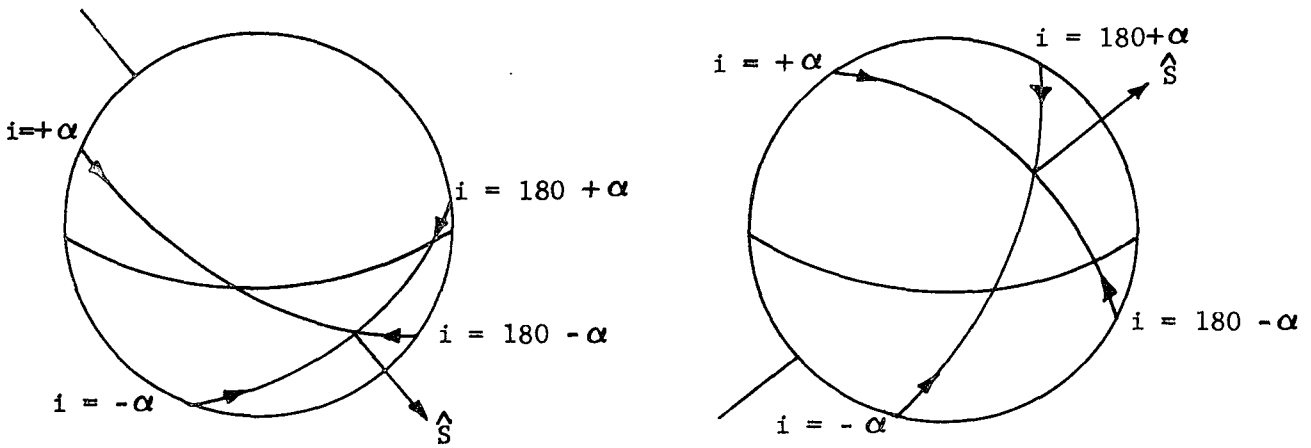


Figure 2. Possible Trajectories with Same Inclination

The detailed computations for the basic part of the program are straightforward. Using the standard conic abbreviations,

$$c = \left| \vec{r} \times \vec{v} \right| \tag{2}$$

$$\hat{w} = \frac{\vec{r} \times \vec{v}}{c} \tag{3}$$

$$p = \frac{c^2}{\mu} \tag{4}$$

$$a = \frac{r}{2 - rv^2/\mu} \tag{5}$$

$$e^2 = 1 - \frac{p}{a} \tag{6}$$

$$b = \sqrt{p\,|a|} \tag{7}$$

$$\cos f = \frac{p - r}{e\,r} \tag{8}$$

$$\sin f = \frac{\dot{r}\,c}{e\,\mu} \tag{9}$$

$$\hat{Z} = \frac{r}{c}\,v - \frac{\dot{r}}{c}\,\vec{r} \tag{10}$$

$$\hat{P} = \frac{\vec{r}}{r}\cos f - \vec{Z}\sin f \tag{11}$$

$$\hat{Q} = \frac{\vec{r}}{r}\sin f + \hat{Z}\cos f \tag{12}$$

$$\hat{S} = -\frac{a}{\sqrt{a^2+b^2}}\,\hat{P} + \frac{b}{\sqrt{a^2+b^2}}\,\hat{Q} \tag{13}$$

$$\hat{T} = \frac{\hat{S} \times \hat{K}}{\hat{S} \times \hat{K}} \tag{14}$$

$$\hat{R} = \hat{S} \times \hat{T} \tag{15}$$

$$\vec{B} = \frac{b^2}{\sqrt{a^2+b^2}}\,\hat{P} + \frac{ab}{\sqrt{a^2+b^2}}\,\hat{Q} \tag{16}$$

$$B \cdot T = \vec{B} \cdot \hat{T} \tag{17}$$

$$B \cdot R = \vec{B} \cdot \hat{R} \tag{18}$$

The computations for the optional part of the program which converts the i and r_{CA} into an equivalent $\vec{B}^*$ proceed as follows. The approach asymptote is first converted into target planet equatorial coordinates and its right ascension and declination computed

$$\hat{S}_q = \phi_{ECEQ}\, \hat{S}$$

$$\theta_S = \tan^{-1} \frac{(Sq)_y}{(Sq)_x} \tag{19}$$

$$\delta_S = \sin^{-1} (Sq)_z$$

The angle $\Delta\theta$ between the ascending node of the trajectory and the right ascension of the approach asymptote is from Napiers rule

$$\sin \theta = \frac{\tan \delta_S}{\tan i} \tag{20}$$

after assuring that $|i| \geq |\delta_S|$. The ascending node of the trajectory is then computed recalling the definitions of the angle i

$$\Omega = \theta_S + \Delta\theta \quad (+\pi) \tag{21}$$

Thus the unit vector to the ascending node is given by

$$\hat{R}_A = (\cos\Omega,\ \sin\Omega,\ 0) \tag{22}$$

The normal to the orbital plane (in target planet equatorial coordinates) is

$$\hat{W}_q = \frac{\hat{S}_q \times \hat{R}_A}{|\hat{S}_q \times \hat{R}_A|} \tag{23}$$

This is now converted to the ecliptic coordinate system

$$\hat{W}_C = \phi^{T}_{ECEQ}\, \hat{W}_q \tag{24}$$

The unit vector in the desired $\vec{B}^*$ direction is

$$\hat{B}^* = \frac{\hat{S} \times \hat{W}_C}{|\hat{S} \times \hat{W}_C|} \tag{25}$$

The magnitude of the $\vec{B^*}$ vector is given by

$$B^* = r_{CA}\sqrt{1 + \frac{2\mu}{r_{CA} V_\infty^2}} \tag{26}$$

Then the target impact parameter is $\vec{B^*} = B^* \hat{B^*}$. The target values are then given by their obvious definitions

$$\begin{aligned} B \cdot T^* &= \vec{B^*} \cdot \hat{T} \\ B \cdot R^* &= \vec{B^*} \cdot \hat{R} \end{aligned} \tag{27}$$

Finally the hyperbolic time from $(\vec{r}, \vec{v})$ to periapsis is computed from the conic formula

$$\begin{aligned} \tanh \frac{F}{2} &= \sqrt{\frac{e - 1}{e + 1}} \tan \frac{f}{2} \\ t &= \sqrt{\frac{-a^3}{\mu}} \, (e \sinh F - F) \end{aligned} \tag{28}$$

Reference: Kizner, W., A Method of Describing Miss Distances for Lunar and Interplanetary Trajectories, Ballistic Missiles and Space Technology vol III, Pergamon Press, New York, 1961.

IMPCT Analysis

The subroutine IMPCT is responsible for computing all of the target parameter data associated with auxiliary targeting. Three basic types of target information are required given a planetocentric ecliptic state. First, what are the actual B-plane pierce point coordinates, $B_A \cdot T$ and $B_A \cdot R$? Second, what are the values of the actual target parameters? These may be triples of inclination, radius and time at closest approach or right ascension, declination, and time at impact. Third, what are the B-plane pierce point coordinates, $B_D \cdot T$ and $B_D \cdot R$ on the current trajectory required to achieve the desired values of the corresponding actual target parameters. In addition to supplying all of this information, IMPCT places it in the appropriate locations for sorting by the processing routine TARØPT.

Whenever IMPCT is called, it first calculates the actual B-plane pierce point coordinates for the current state. In the process it also calculates other useful information about the osculating conic, including the parameters a, e, θ, $\underline{W}$, $\underline{S}$, $\underline{T}$ and $\underline{R}$. For the equations giving these quantities see the subroutine STIMP analysis. If option flag KØPT is 1, only this information is desired and a return to the calling program is executed.

The values of the actual target parameters are calculated if KØPT is not 1. If the targets are inclination, radius, and time at closest approach, KØPT must be 2. If TARGET is not in the second phase of a two-phase targeting case, the target parameters, i, r_{CA}, and t_{CA} are obtained by conic extrapolation from the current state (SOI):

$$r_{CA} = a\ (1 - e). \tag{1}$$

Let D denote the transformation from planetocentric ecliptic coordinates to the planetocentric equatorial frame:

$$i = \cos^{-1}\left[(D\ \underline{W})_3\right] \tag{2}$$

If TARGET is in a second phase, the virtual mass program will have integrated the trajectory all the way to closest approach rather than stopping at the SOI. Hence refined values of all three target parameters are available from VMP.

Suppose, on the other hand, that the targets are right ascension α, declination δ, and time t_I at impact. Then KØPT must be 3.

Again if TARGET is not involved in a second phase, these target parameters are calculated by extrapolating conically from the SOI. Let $\underline{r}_I$ and $\underline{\rho}_I$ denote the position vectors of the vehicle at impact in the planetocentric ecliptic and probe-sphere frames, respectively. Let C denote the transformation from the former frame to the latter. Obviously

$$\underline{\rho}_I = C\underline{r}_I. \tag{4}$$

The right ascension and declination at impact are then

$$\alpha = \tan^{-1}\left[\left(\rho_I\right)_2 \Big/ \left(\rho_I\right)_1\right], \tag{5}$$

and

$$\delta = \sin^{-1}\left[\left(\rho_I\right)_3 \Big/ \rho_I\right]. \tag{6}$$

Let Δt_{SC} and Δt_{IC} be the times from the sphere of influence and from the probe sphere to closest approach, respectively. Let θ_I denote the true anomaly on the osculating conic at the probe sphere:

$$\cos\theta_I = \left(p/r_I - 1/a\right)/e \tag{7}$$

where

$$p = a\left(1 - e^2\right) \tag{8}$$

is the semilatus rectum of the conic. Since impact occurs before peripsis,

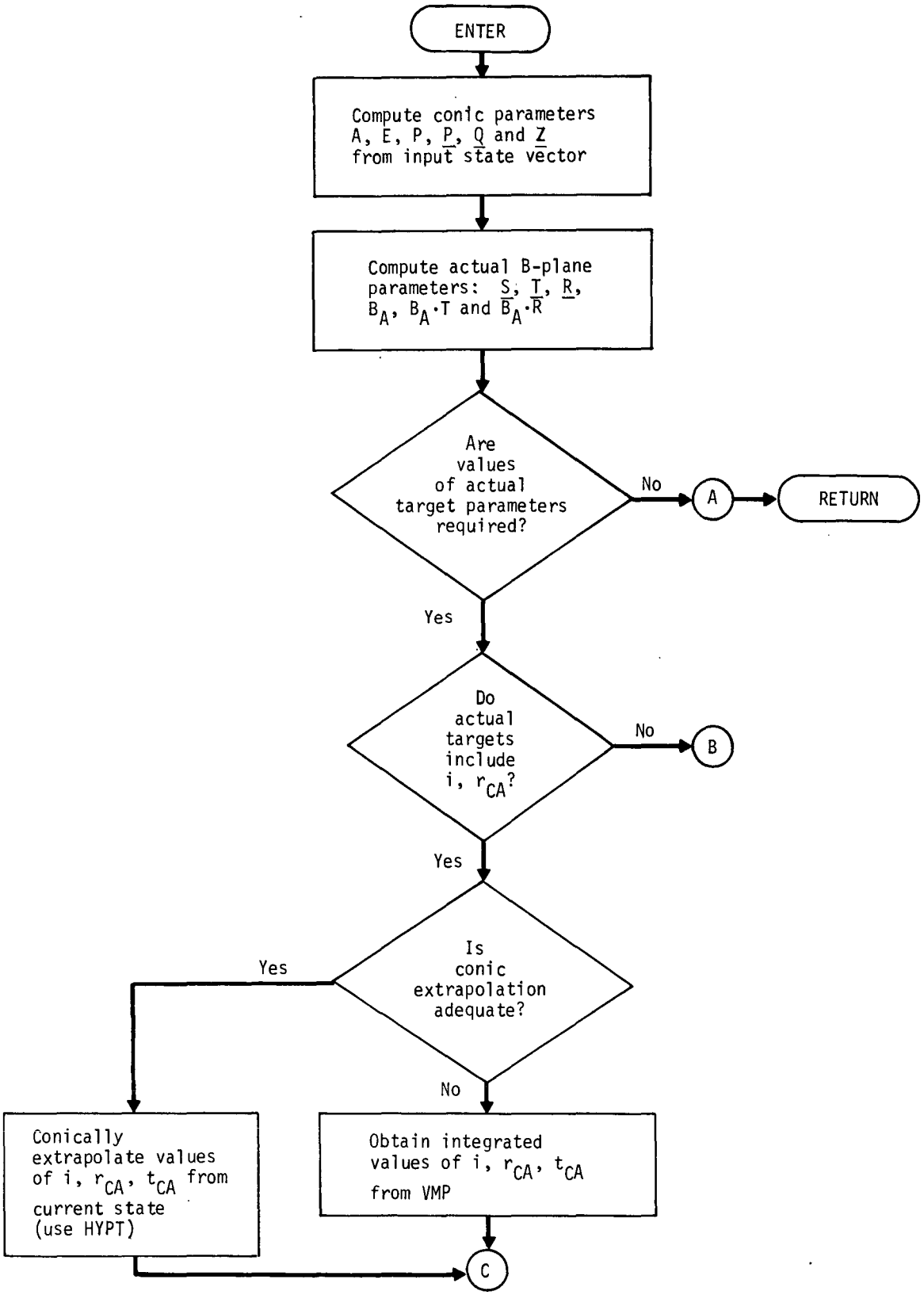
$$\sin\theta_I = -\sqrt{1 - \cos^2\theta_I}. \tag{9}$$

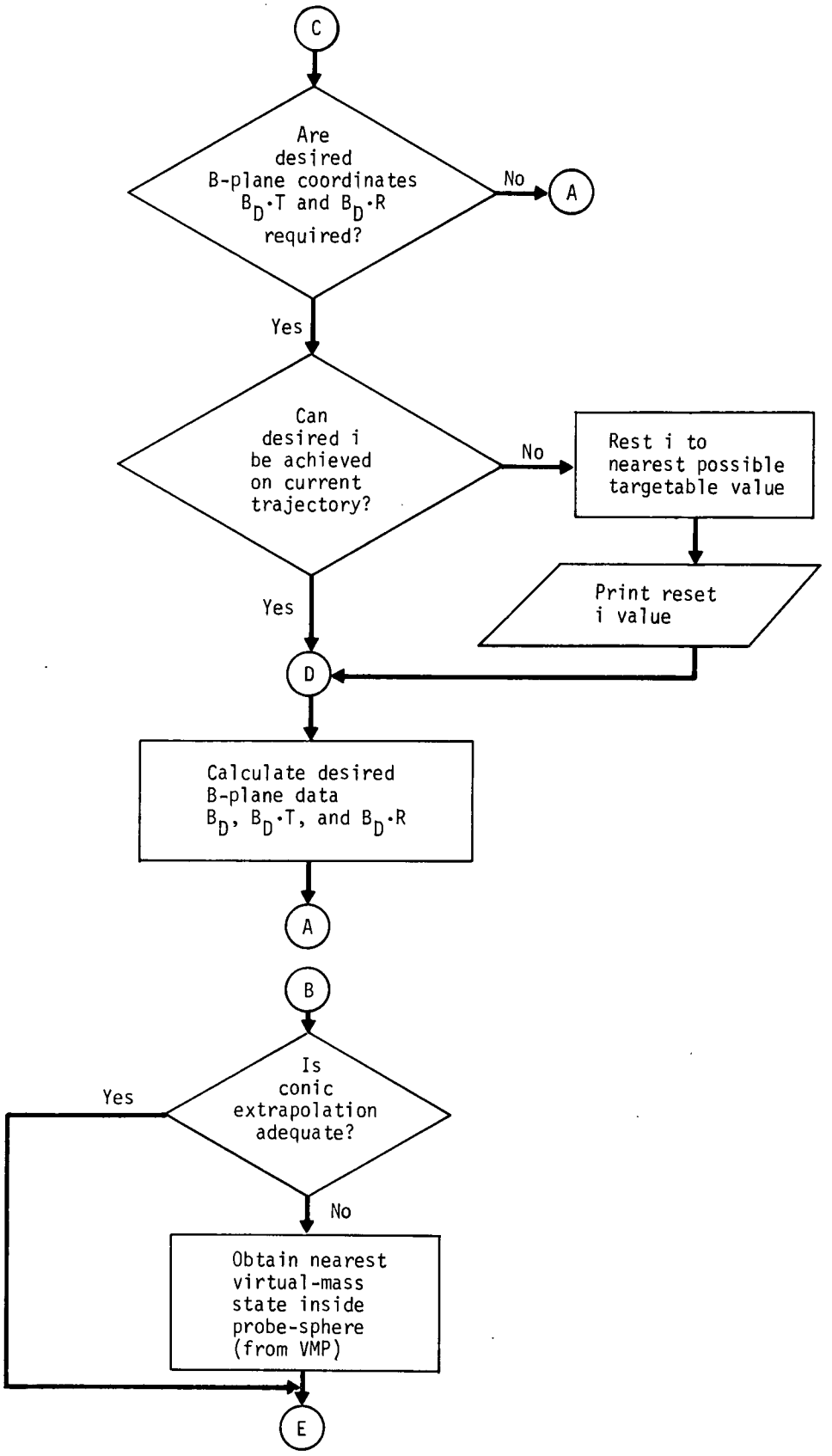
With the true anomalies on the conic at the current state and at the probe sphere available, IMPCT cals HYPT to determine Δt_{SC} and Δt_{IC}. If, as above, t_{SOI} denotes the time from the sphere of influence to periapsis,

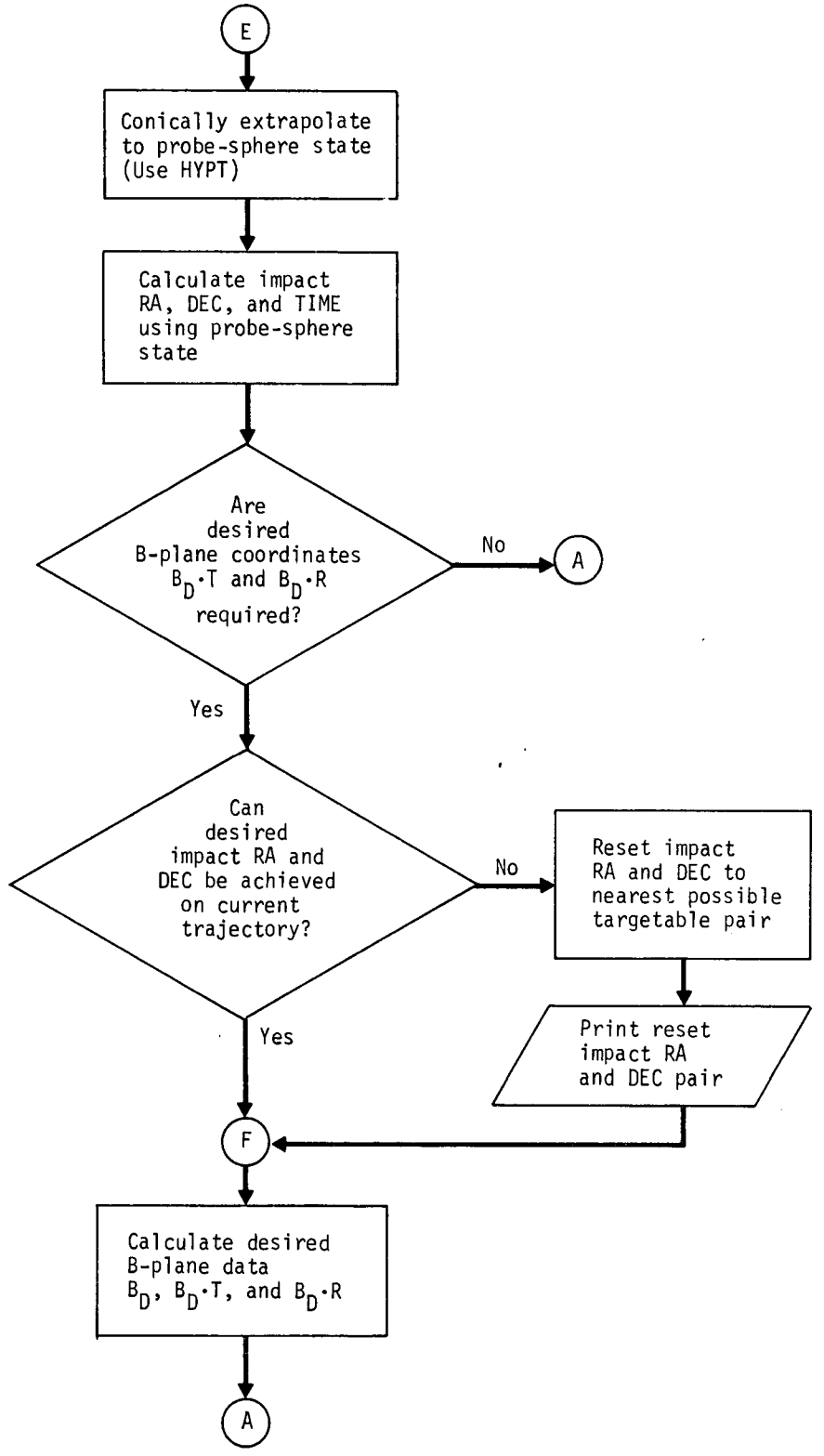
$$t_I = t_{SOI} + \left(\Delta t_{SC} - \Delta t_{IC}\right). \tag{10}$$

Finally, if TARGET is involved in the second phase of targeting, the virtual-mass algorithm integrates the trajectory all the way to the first integration increment inside the probe sphere. Hence the preceding conic extrapolation formulae can be used to obtain accurate impact target parameters by replacing the state at the SOI with the first state inside the probe sphere.

IMPCT calculates the desired B-plane pierce point coordinates $B_D \cdot T$ and $B_D \cdot R$ for either the i and r_{CA} or α and β target options if the flag ITARR is 2 indicating that a new control iteration is being made. The equations and logical flow of this calculation for the former target option is given in the subroutine IMPACT while tose for the latter are presented in DIMPCP.


ENTER
Compute conic parameters A, E, P, P, Q and Z from input state vector
Compute actual B-plane parameters: S, T, R, B_A, $B_A \cdot T$ and $B_A \cdot R$
Are values of actual target parameters required?
No
A
RETURN
Yes
Do actual targets include i, r_{CA}?
No
B
Yes
Is conic extrapolation adequate?
Yes
No
Conically extrapolate values of i, r_{CA}, t_{CA} from current state (use HYPT)
Obtain integrated values of i, r_{CA}, t_{CA} from VMP
C


C
Are desired B-plane coordinates $B_D \cdot T$ and $B_D \cdot R$ required?
No
A
Yes
Can desired i be achieved on current trajectory?
No
Rest i to nearest possible targetable value
Print reset i value
Yes
D
Calculate desired B-plane data B_D, $B_D \cdot T$, and $B_D \cdot R$
A
B
Is conic extrapolation adequate?
Yes
No
Obtain nearest virtual-mass state inside probe-sphere (from VMP)
E


E
Conically extrapolate to probe-sphere state (Use HYPT)
Calculate impact RA, DEC, and TIME using probe-sphere state
Are desired B-plane coordinates $B_D \cdot T$ and $B_D \cdot R$ required?
No
A
Yes
Can desired impact RA and DEC be achieved on current trajectory?
No
Reset impact RA and DEC to nearest possible targetable pair
Print reset impact RA and DEC pair
Yes
F
Calculate desired B-plane data B_D, $B_D \cdot T$, and $B_D \cdot R$
A

INSERS Analysis

INSERS controls the processing of an orbital insertion event. The subroutine COPINS and NONINS perform the actual computations for the coplanar and non-planar options respectively.

INSERS first records the specific parameter values for the current orbit insertion event.

It then computes the current state ($\vec{r}$, $\vec{v}$) of the spacecraft in target-planet centered ecliptic coordinates. Subroutine PECEQ is called to compute the transformation matrix Φ_{ECEQ} from ecliptic to equatorial coordinates. The planet centered equatorial coordinates are then

$$\vec{r}_q = \Phi_{ECEQ}\, \vec{r}$$

$$\vec{v}_q = \Phi_{ECEQ}\, \vec{v}$$

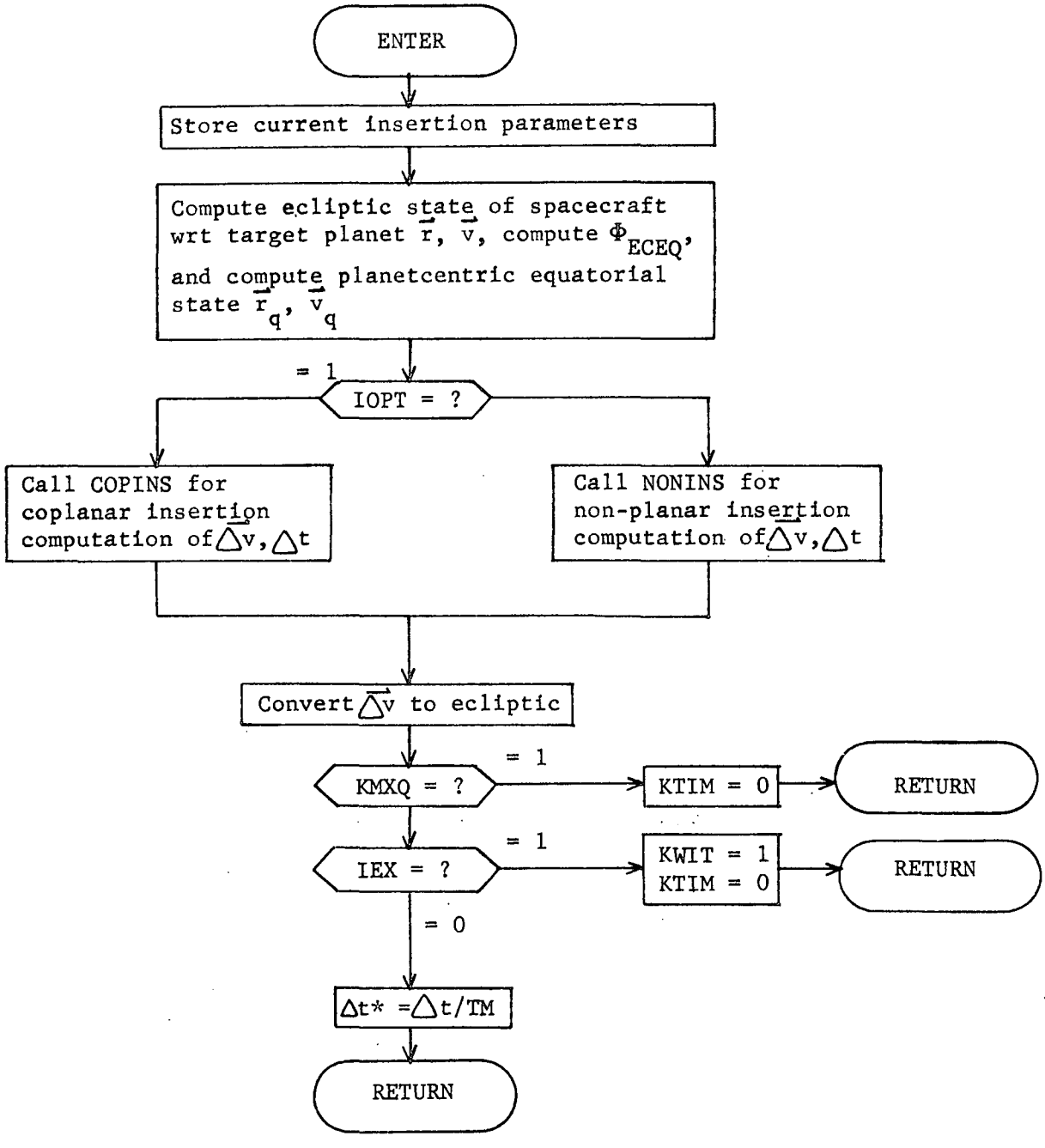
This state is then sent to COPINS or NONINS for the computation of the insertion velocity $\vec{\Delta v}_q$ and the time interval t between the current time and the time at which the insertion should take place (based on conic propagation about the target body). The correction $\vec{\Delta v}_q$ is then converted to ecliptic coordinates

$$\vec{\Delta v} = \Phi^T \vec{\Delta v}_q$$

If the event is a compute-only mode, the return is made to GIDANS.

If the event is to be executed the flag IEX (set by COPINS or NONINS to indicate success or failure) is then interrogated. If IEX = 1, no acceptable insertion event was found and so the executive flag KWIT is set to 1 before returning. If IEX = 0 an acceptable insertion was determined and so it is set up.

INSERS Flow Chart



JACOBI Analysis

The Jacobi method subjects a real, symmetric matrix A to a sequence of transformations based on a rotation matrix:

$$O_K = \begin{bmatrix} \cos\phi_K & -\sin\phi_K \\ \sin\phi_K & \cos\phi_K \end{bmatrix}$$

where all other elements of the rotation matrix are identical with the unit matrix. After n multiplications A is transformed into:

$$A' = O_N^{-1} \ldots O_1^{-1} A O_1 \ldots O_N$$

If ϕ_K is chosen at each step to make a pair of off-diagonal elements zero, then A' will approach diagonal form with the eigenvalues on the diagonal. The columns of $O_1 O_2 \ldots O_N$ correspond to the eigenvectors of A.

The angle of rotation ϕ is chosen in the following way. If the four entries of O_K are in (i,i), (i,j), (j,i) and (j,j) then the corresponding elements of $O_1^{-1} A O_1$ are

$$b_{ii} = a_{ii} \cos^2\phi + 2a_{ij} \sin\phi \cos\phi + a_{jj} \sin^2\phi$$
$$b_{ij} = b_{ji} = (a_{jj} - a_{ii}) \sin\phi \cos\phi + a_{ij}(\cos^2\phi - \sin^2\phi)$$
$$b_{jj} = a_{ii} \sin^2\phi - 2a_{ij} \sin\phi \cos\phi + a_{jj} \cos^2\phi$$

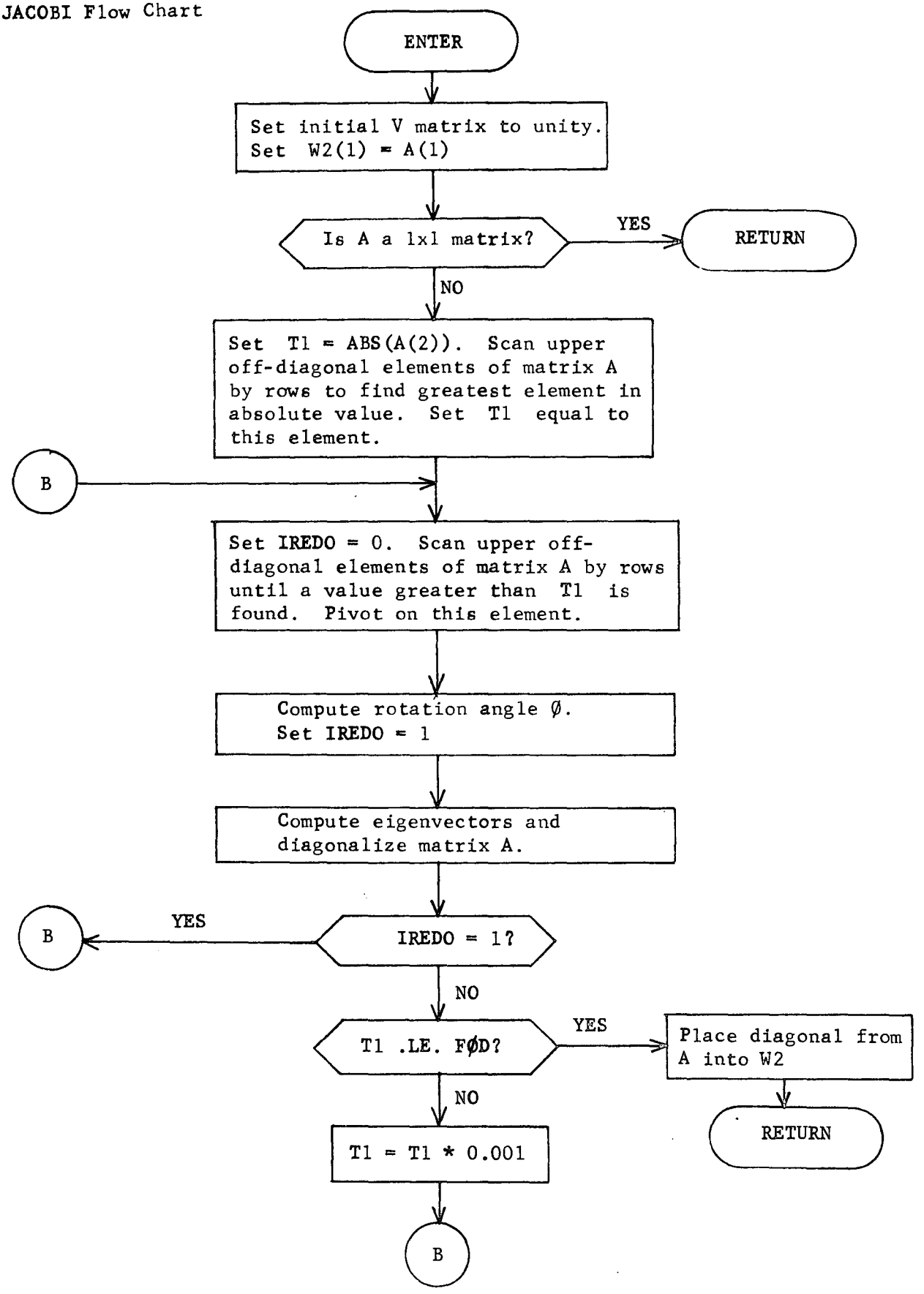
If ϕ is chosen so that $\tan 2\phi = 2a_{ij}/(a_{ii} - a_{jj})$ then

$$b_{ij} = b_{ji} = 0$$

Each multiplication creates a new pair of zeros but will introduce a non-zero contribution to positions zeroed out on previous steps. However, successive matrices of the form $O_2^{-1} O_1^{-1} A O_1 O_2$ will approach the required diagonal form.

Reference: Scheid, Frances: Theory and Problems of Numerical Analysis, McGraw-Hill Book Company, Inc., New York, 1968.

JACOBI Flow Chart



KTRØL Analysis

KTRØL calculates the targeting velocity increment $\underline{\Delta v}$ given the targeting state vector $\underline{X} = (\underline{r} / \underline{v})^T$ of the spacecraft relative to the launch planet and the launch-planetocentric velocity controls c_1, c_2, and c_3. This computation is required in two distinct situations. The first is in calculating the sensitivity matrix of the auxiliary parameters to the velocity controls by successively perturbing each control while holding the remaining two constant. The second is in applying the control correction indicated by the Newton-Raphson algorithm to arrive at the next iterate to the postimpulse targeting state.

In either case the three unit vectors $\underline{V}$, $\underline{U}$, and $\underline{W}$ that serve to define the local, spherical, velocity-control coordinate system are first computed

$$\underline{V} = \frac{\underline{v}}{v} \tag{1}$$

$$\underline{W} = \frac{\underline{r} \times \underline{v}}{\| \underline{r} \times \underline{v} \|} \tag{2}$$

$$\underline{U} = \underline{W} \times \underline{V} \, . \tag{3}$$

$\underline{V}$ specifies the direction of zero latitude and zero longitude in the control frame while the W axis determines the +z or polar direction. Then c_2 and c_3 are, respectively, the latitude and longitude of the posttargeting velocity while c_1 is the increase in length of that velocity. Figure 1 defines the controls pictorially when the earth is the launch planet. The velocity increments required in either of the two situations mentioned above can readily be calculated in terms of the vector $\underline{V}$, $\underline{U}$, and $\underline{W}$.

First consider the calculation of the increment $\underline{\Delta v}_i$ produced by perturbing the ith control an amount c_i while fixing the other controls at zero as required in the sensitivity approximation. KTRØL performs this conputation when IOPT = i. Reasoning from Figure 1 it follows immediately that

$$\underline{\Delta v}_1 = c_1 \underline{V} \tag{4}$$

$$\underline{\Delta v}_2 = \| \underline{v} \| \left(\cos c_2 - 1 \right) V + \sin c_2 \, \underline{U} \tag{5}$$

$$\underline{\Delta v}_3 = \| \underline{v} \| \left(\cos c_3 - 1 \right) \underline{V} + \sin c_3 \, \underline{W}. \tag{6}$$

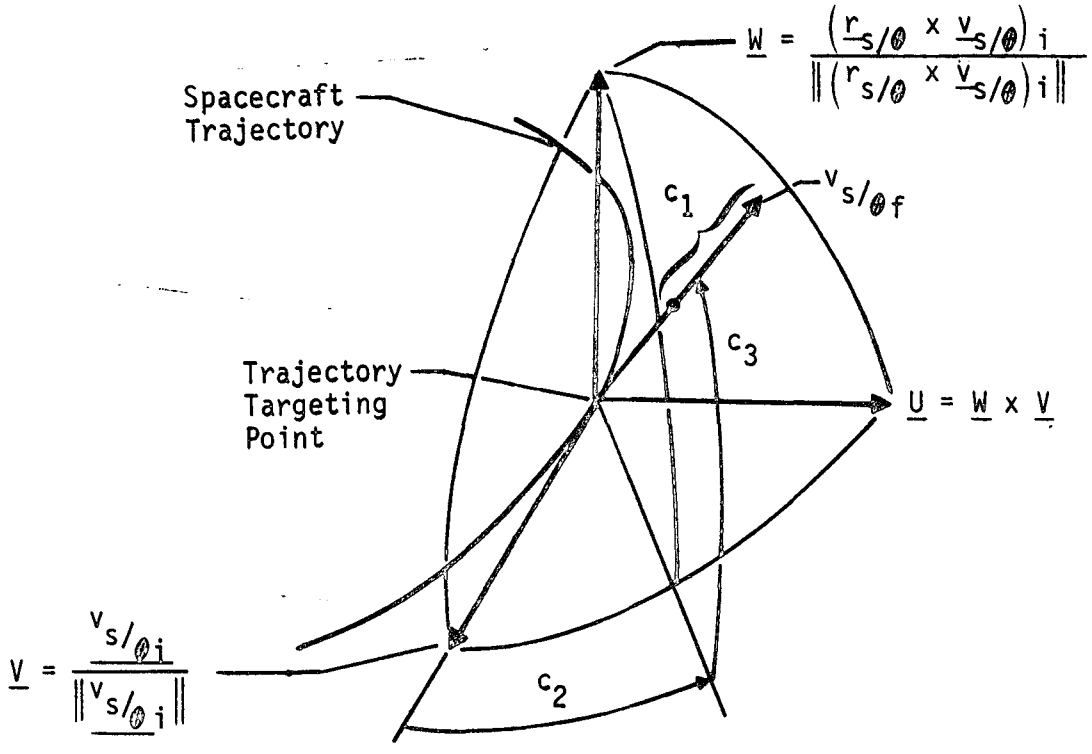


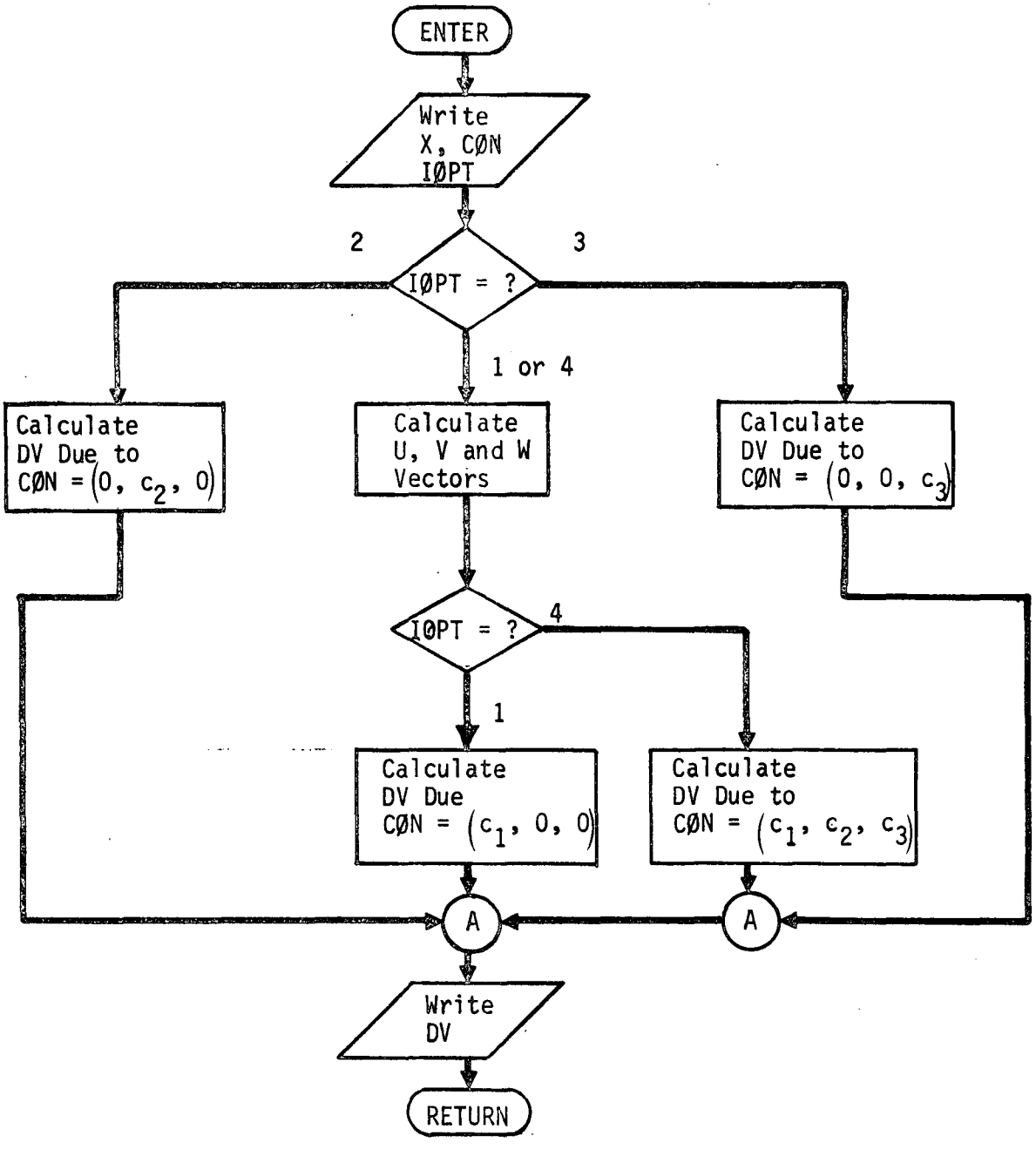
Figure 1 Pictorial Definition of Launch-Planetocentric Targeting Controls

Next consider the calculation of the increment $\Delta\underline{v}$ produced by perburbing all three controls simultaneously as required in the Newton-Raphson control correction. KTRØL performs this computation when IØPT = 4. Reasoning again from Figure 1

$$\Delta v = \left[\left(\|\underline{v}\| + c_1\right) \cos c_2 \cos c_3 - \|\underline{v}\|\right]\underline{V} + \left(\|\underline{v}\| + c_1\right) \sin c_2 \cos c_3\, \underline{U} + \left(\|\underline{v}\| + c_1\right) \sin c_3\, \underline{W}. \quad (7)$$

Note that equation (7) degenerates to equations (4), (5), and (6) when the appropriate controls are set to zero.

KTRØL Flow Chart



LAUNCH Analysis

LAUNCH computes the injection time, position and velocity from the departure velocity $\vec{v}_{HE}$ (computed in HELIO) and the launch profile parameters input by the user.

The rotation matrix Φ_{ECEQ} defining the transformation from ecliptic to equatorial coordinates is first computed (PECEQ). The departure velocity $\vec{v}_{HE}$ is then normalized and converted into ecliptic coordinates to yield the departure asymptote $\hat{S}$.

$$\hat{S} = \Phi_{ECEQ} \frac{\vec{v}_{HE}}{v_{HE}} \tag{1}$$

Auxiliary information associated with $\hat{S}$ is then computed. The energy C_3, the declination Φ_S and the right ascension θ_S of the departure asymptote, and the eccentricity of the departure hyperbola are given by

$$C_3 = v_{HE}^2$$

$$\sin \Phi_S = S_z$$

$$\tan \theta_S = \frac{S_y}{S_x}$$

$$e = 1 + \frac{r_p C_3}{\mu} \tag{2}$$

where r_p is the desired parking orbit radius and μ is the gravitational constant of the launch planet.

The unit normal $\hat{W}$ to the launch plane in equatorial coordinates is then computed. $\hat{W}$ is defined by

$$W_z = \cos \Phi_L \sin \Sigma_L$$

$$W_y = \frac{-W_z S_y S_z + k S_x \left[1 - (S_z^2 + W_z^2)\right]^{\frac{1}{2}}}{S_x^2 + S_y^2}$$

$$W_z = \frac{-(W_y S_y + W_z S_z)}{S_x} \tag{3}$$

where Φ_L is the launch site latitude, Σ_L is the launch azimuth, and $k = +1$ or -1 for the long or short coast time models respectively. The second equation defines an implicit constraint on Σ_L

$$\sin^2 \Sigma_L \leq \frac{\cos^2 \Phi_s}{\cos^2 \Phi_L} \tag{4}$$

The right ascension at launch Θ_L may now be defined by

$$\cos \Theta_L = \frac{W_x \sin \Phi_L \sin \Sigma_L + W_y \cos \Sigma_L}{W_z^2 - 1}$$

$$\sin \Theta_L = \frac{W_y \sin \Phi_L \sin \Sigma_L - W_x \cos \Sigma_L}{W_z^2 - 1} \tag{5}$$

and the unit vector toward the launch position is then

$$R_L = (\cos \Phi_L \cos \Theta_L \ , \ \cos \Phi_L \sin \Theta_L \ , \ \sin \Phi_L) \tag{6}$$

The complementary unit vectors $\hat{P}, \hat{Q}$ defining the orientation of the hyperbola within the launch plane are now introduced. Let

$$\hat{B} = \hat{S} \times \hat{W} \tag{7}$$

The true anomaly of the departure asymptote is $\cos f_s = -\frac{1}{e}$. Then $\hat{P}$ and $\hat{Q}$ are given as

$$\hat{P} = \hat{S} \cos f_s + \hat{B} \sin f_s$$

$$\hat{Q} = \hat{S} \sin f_s + \hat{B} \cos f_s \tag{8}$$

The true anomaly of the launch site f_I may now be given

$$\cos f_L = \hat{R}_L \cdot \hat{P}$$

$$\sin f_L = \hat{R}_L \cdot \hat{Q} \tag{9}$$

The angle Ψ_B between launch and injection is

$$\Psi_B = 2\pi - f_L + f_I \tag{10}$$

where f_I is the desired true anomaly at injection read in as input. The coast time t_c may now be computed from

$$t_c = \left[\Psi_B - (\Psi_1 + \Psi_2)\right] k_\Phi \tag{11}$$

where Ψ_1 and Ψ_2 are the angles of the first and second burns and k_Φ is the inverse of the parking orbit coast rate, all of which are read in as input.

The time between launch and injection is therefore

$$t_B = t_1 + t_2 + t_c \tag{12}$$

where t_1 and t_2 are the input time durations of the first and second burns.

The unit vector to injection is

$$\hat{R}_I = \hat{P} \cos f_I + \hat{Q} \sin f_I \tag{13}$$

The semi-latus rectum p is

$$p = \frac{\mu(e^2 - 1)}{C_3} \tag{14}$$

The radius magnitude to injection is

$$R_I = \frac{p}{1 + e \cos f_I} \tag{15}$$

The injection speed is

$$V_I = \sqrt{C_3 + \frac{2\mu}{R_I}} \tag{16}$$

The path angle at injection is

$$\cos \Gamma_I = \frac{\sqrt{\mu p}}{R_I V_I} \tag{17}$$

The injection latitude is

$$\sin \Phi_I = \hat{R}_{I_z} \tag{18}$$

The injection right ascension is

$$\tan \Theta_I = \frac{R_{I_y}}{R_{I_x}} \tag{19}$$

The injection longitude is

$$\theta_I = \theta_L + \Theta_I - \Theta_L - \omega t_B \tag{20}$$

where θ_L is the longitude of the launch site and ω is the rotation rate of the launch planet, both being read in as input.

The injection azimuth is

$$\cos \Sigma_I = \frac{S_z - \cos (f_s - f_I) \sin \Phi_I}{\sin (f_s - f_I) \cos \Phi_I} \tag{21}$$

The launch time on the day of launch is

$$t_L = \frac{(\Theta_L - \theta_L - \text{GHA}) \bmod 2\pi}{\omega} \tag{22}$$

where GHA is the Greenwich hour angle at 0^h UT of the launch date

$$\text{GHA} = 100^\circ.07554260 + 0^\circ.9856473460\ T_d + 2^\circ.9015 \times 10^{1-3} T_d^{\ 3} \tag{23}$$

where T_d = days past 0^h January 1, 1950.

The injection radius vector is now computed from

$$\vec{R}_I = R_I \hat{R}_I \tag{24}$$

$$\vec{V}_I = \frac{V_I}{R_I} \left[(\hat{W} \times \vec{R}_I) \cos \Gamma_I + \vec{R}_I \sin \Gamma_I \right]$$

The injection time is

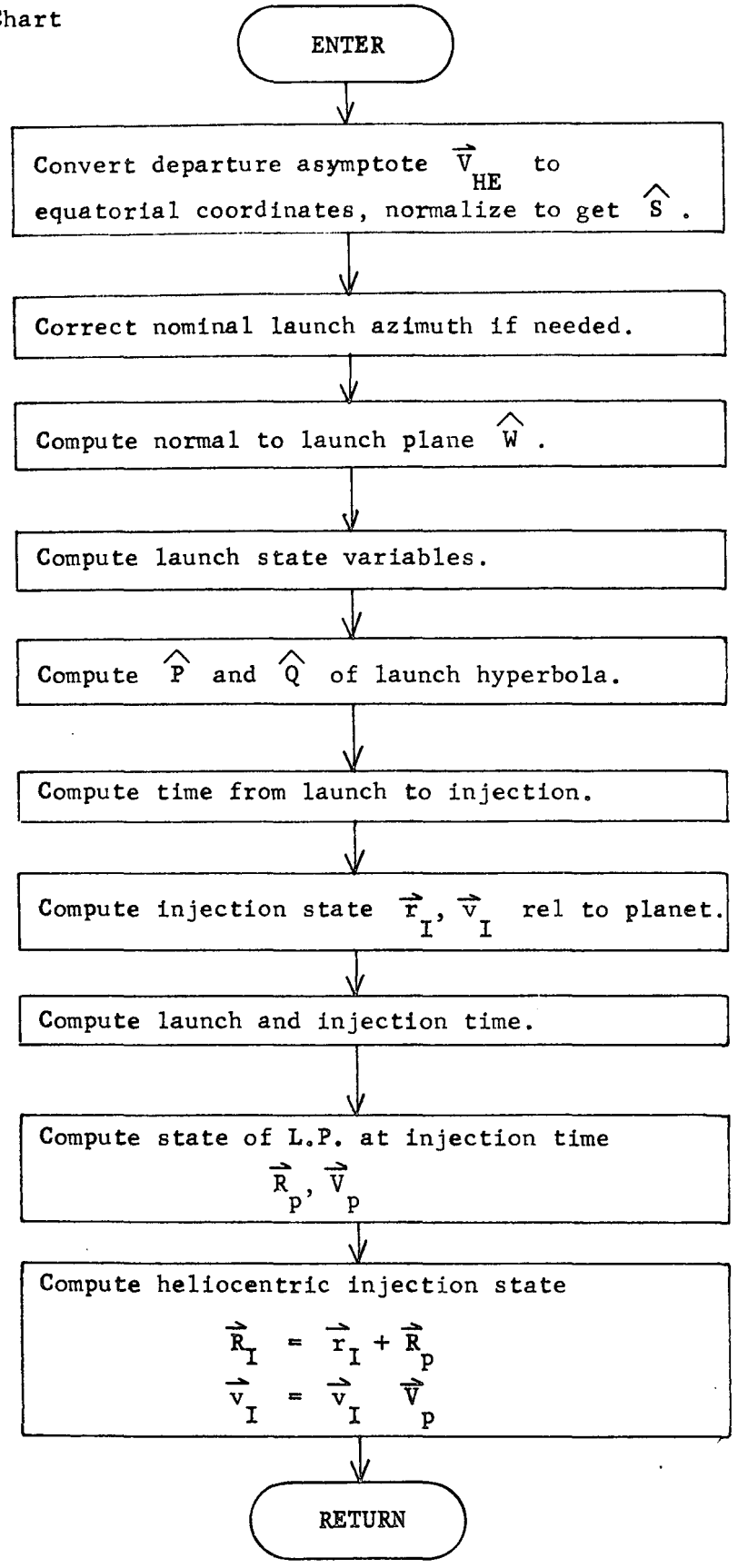
$$T_I = T_o + t_L + t_B \tag{25}$$

where T_o is the Julian date of the launch calendar date.

The injection position and velocity are now rotated into the ecliptic plane. The position and velocity of the launch planet at the time T_I are computed and added to the injection state to get the heliocentric injection state.

Reference: Space Research Conic Program, Phase III, May 1, 1969, Jet Propulsion Laboratory, Pasadena, California.

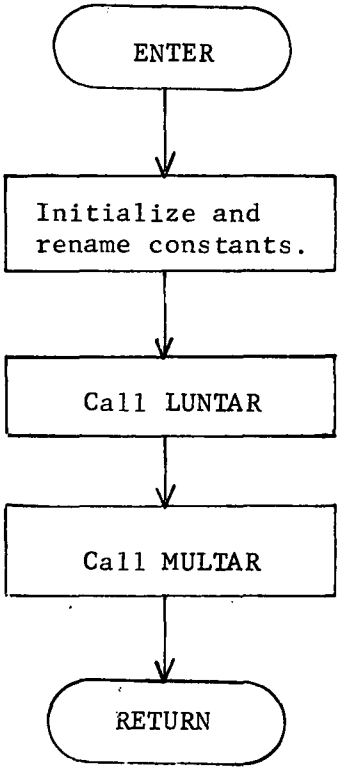
LAUNCH Flow Chart



LUNA Analysis

LUNA is the controlling subroutine for lunar zero iterate targeting. It first serves an interface role in which it initializes constants and renames variables for the other lunar targeting routines. It then calls LUNTAR for the targeting of the lunar patched conic. When that is completed it calls MULTAR for the targeting of the nulti conic trajectory. It then returns control to PRELIM.

LUNA Flow Chart



LUNCON Analysis

The point of intersection of the Earth-centered conic with the lunar sphere of influence (LSI) is determined by the angles and δ. Relative to the moon in Earth-equatorial coordinates tnat point is

$$\vec{r_{SI}} = \begin{bmatrix} R_{SI} \cos\delta \cos\theta \\ R_{SI} \cos\delta \sin\theta \\ R_{SI} \sin\delta \end{bmatrix} \quad (1)$$

where R_{SI} is the radius of the LSI. Relative to the earth that point is

$$\vec{R_q} = \vec{R_M} + \vec{r_{SI}} \quad (2)$$

where $\vec{R_M}$ is the radius vector to the center of the moon at the time of LSI intersection t_{SI} in earth equatorial coordinates.

There are at most two planes which contain $\vec{R_q}$ and satisfy tne launch latitude ϕ and azimuth Σ constraints. Let $\hat{W}$ denote the unit normal to either of these planes. Now let $\hat{R}_L$, θ_L, ϕ_L denote the unit vector, longitude, and latitude of the launch site. Construct a local horizon coordinate system at the launch site as indicated in Figure 1.

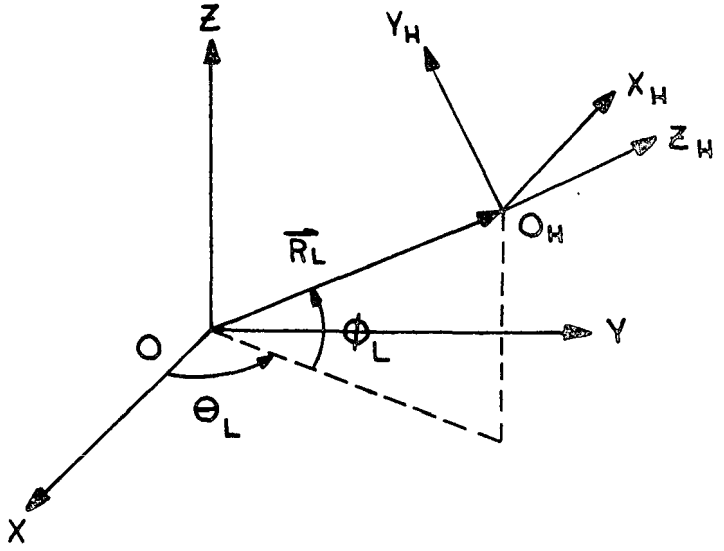


Figure 1. Local Horizon Coordinate System

Here $\hat{Z}_h = \frac{\hat{R}_L}{R_L}$, $\hat{Y}_L$ is normal to $\hat{Z}_h$ in the $\hat{Z}$-O-$\hat{R}_L$ plane, and $\hat{X}_h = \hat{Y}_h \times \hat{Z}_h$.

In the local horizon system, the position and velocity are very simply represented

$$\vec{R}_h = R \left[0,\ 0,\ 1\right]^T$$
$$\vec{V}_h = V \left[\cos\delta \sin\Sigma\ ,\ \cos\delta \cos\Sigma\ ,\ \sin\delta\right]^T \tag{3}$$

where Σ is the launch azimuth and δ is the declination wrt the local horizontal. Thus

$$\hat{W}_h = \frac{\vec{R}_h \times \vec{V}_h}{|\vec{R}_h \times \vec{V}_h|} = \begin{bmatrix} -\cos\Sigma \\ \sin\Sigma \\ 0 \end{bmatrix} \tag{4}$$

The transformation matrix converting a vector in the local horizon system to the equatorial system is

$$T = \begin{bmatrix} -\sin\theta_L & -\sin\phi_L \cos\theta_L & \cos\phi_L \cos\theta_L \\ \cos\theta_L & -\sin\phi_L \cos\theta_L & \cos\phi_L \sin\theta_L \\ 0 & \cos\phi_L & \sin\phi_L \end{bmatrix} \tag{5}$$

Therefore since $\vec{W}_q = T\,\vec{W}_h$, the z-component of $\hat{W}$ in the equatorial coordinate system is

$$\hat{W}_Z = \cos\phi_L \sin\Sigma \tag{6}$$

Since $\hat{W}$ is a unit normal it must satisfy both $\hat{W}\cdot\hat{W} = 1$ and $\hat{W}\cdot\hat{S} = 0$ where $\hat{S} = \frac{\hat{R}_q}{R_q}$. Solving for the two remaining components of $\hat{W}$,

$$\hat{W}_y = \frac{-\hat{W}_z \hat{S}_y \hat{S}_z \pm \hat{S}_x \sqrt{1 - (\hat{S}_z^2 + \hat{W}_z^2)}}{\hat{S}_x^2 + S_y^2} \tag{7}$$

$$\hat{W}_x = -\frac{(\hat{W}_y \hat{S}_y + \hat{W}_z \hat{S}_z)}{\hat{S}_x} \tag{8}$$

To eliminate the ambiguity of sign in (7) the short-coast plane corresponding to the negative sign is used. Note that (7) also imposes a constraint on the launch azimuth

$$\sin^2\Sigma \leq \frac{1 - \hat{S}_z^2}{\cos^2\phi_L} \tag{9}$$

Now choose $\hat{U} = \hat{W} \times \hat{S}$ to complete a right hand system $(\hat{S}, \hat{U}, \hat{W})$. Then the position at LSI relative to the earth is $(R_I, 0, 0)$. Now let α determine the perigee point in the orbital plane $(\hat{W} = 0)$ measured counterclockwise from the $-\hat{S}$ axis. Then the perigee point is $(-r_p \cos\alpha, -r_p \sin\alpha, 0)$ where r_p is the parking orbit radius (input). Therefore the true anomaly of the earth centered conic at the LSI is given by

$$f_{SI} = 180 - \alpha \tag{10}$$

The two equations $R_I = \frac{a(1 - e^2)}{1 + e \cos f_{SI}}$ and $r_p = a(1 - e)$ may be solved simultaneously for the semi-major axis a and eccentricity e of the unique earth centered conic

$$e_g = \frac{R_I - r_p}{r_p - R_I \cos f_{SI}} \tag{11}$$

$$a_g = \frac{r_p}{1 - e_g} \tag{12}$$

Thus the velocity of the earth centered conic at the LSI is in the $(\hat{S}, \hat{U}, \hat{W})$ system

$$\vec{V}_o = \begin{bmatrix} \sqrt{a(1-e^2)}\ e \sin f_{SI} \\ \mu a(1-e^2) / R_I \\ 0 \end{bmatrix} \tag{13}$$

Transforming to the earth equatorial coordinate system

$$\vec{V}_q = \begin{bmatrix} S_x & U_x & W_x \\ S_y & U_y & W_y \\ S_z & U_z & W_z \end{bmatrix} \vec{V}_o \tag{14}$$

Now if $(\vec{R}_{MQ}, \vec{V}_{MQ})$ are the position and velocity of the moon at t_{SI} Earth-centered coordinates and $(\vec{R}_Q, \vec{V}_Q)$ are the position and velocity of the spacecraft at t_{SI} then tne state of the spacecraft with respect to the moon at t_{SI} is in earth equatorial coordinates

$$\begin{aligned} \vec{r}_{SI} &= \vec{R}_Q - \vec{R}_{MQ} \\ \vec{v}_{SI} &= \vec{V}_Q - \vec{V}_{MQ} \end{aligned} \tag{15}$$

Using the transformation matrix ϕ_{EQLQ} defining transformations from earth equatorial to lunar equatorial the state in the LQ system is

$$\begin{aligned} \vec{r}_{sq} &= \phi_{EQLQ}\, \vec{r}_{SI} \\ \vec{v}_{sq} &= \phi_{EQLQ}\, \vec{v}_{SI} \end{aligned} \tag{16}$$

The impact plane parameters $B\cdot T$ and $B\cdot R$, and the inclination i_ℓ, may now be computed by calling subroutines ACTB and CAREL.

LUNTAR Analysis

LUNTAR generates a patched conic trajectory arriving at closest approach to ghe Moon at a specified time t_{CA} and meeting prescribed target values at that point as well as standard launch quantities. The target parameters are

t_{CA}	Julian date of required closest approach (CA) referenced 1900
r_{CA}	Radius of CA
i_{CA}	Inclination (relative to lunar equator) at CA[1]
a_{CA}	Semi-major axis at CA

The launch parameters

ϕ_L	Launch site latitude
θ_L	Launch site longitude
Σ_L	Launch azimuth (nominally set to 90°)
r_p	Parking orbit radius

The eccentricity of the moon-centered hyperbola may be computed

$$e_{CA} = 1 - \frac{r_{CA}}{a_{CA}} \tag{1}$$

where $a_{CA} < 0$. The hyperbolic time Δt to go from R_{SI} (radius of lunar sphere of influence (LSI)) to periapsis may be computed from

$$\Delta t = f(\mu_M, a_{CA}, e_{CA}, R_{SI}) \tag{2}$$

where μ_M is the lunar gravitational constant. The time at which the probe should intersect the LSI is then

$$t_{SI} = t_{CA} - \Delta t \tag{3}$$

---

[1] The inclination must be specified according to the format described in IMPACT. For $0 \leq i < 90^\circ$ the inclinations $\pm i$ prescribe posigrade orbits while $180^\circ \pm i$ define retrograde orbits. The positive signs denote approaches from the north, the negative signs designate southern approaches.

The position $\vec{R}_{ME}$ and velocity $\vec{V}_{ME}$ of the moon at t_{SI} relative to the earth in earth ecliptic (EC) coordinates are computed by calling ORB and EPHEM. Transformation matrices ϕ_{ECEQ} and ϕ_{EQLQ} defining transformations from EC to EQ (earth equatorial) and EQ to LQ (lunar equatorial) respectively are then computed by PECEQ. The position and velocity of the moon in the EQ system are

$$\begin{aligned} \vec{R}_{MQ} &= \vec{\phi}_{ECEQ}\, \vec{R}_{ME} \\ \vec{V}_{MQ} &= \vec{\phi}_{ECEQ}\, \vec{V}_{ME} \end{aligned} \tag{4}$$

Call the point of intersection of the vector R_{MQ} with the LSI the bullseye point. Then in moon-centered Earth-equatorial coordinates the vector to the bullseye point is given by

$$\vec{r}_B = -\left(\frac{\vec{R}_{MQ}}{R_{MQ}}\right) R_{SI} \tag{5}$$

From this vector one can calculate a set of angular coordinates (δ_o, θ_o) of the bullseye point. Any other point on the LSI is determined by giving general coordinates $(\delta, \theta) = (\delta_o + \Delta\delta, \theta_o + \Delta\theta)$.

Now let such a set of coordinates by given. They determine a vector $\vec{R}_I$ from earth to the LSI (in the EQ-system). The vector $\vec{R}_I$ along with the launch parameters ϕ_L, θ_L, Σ_L then determines the plane of the Earth-LSI transfer (see LUNCON). Now let α be measured counter-clockwise in that plane from $-\vec{R}_I$. The parameter α specifies the location of the perigee point of the transfer conic, thus the vector to perigee is fixed as $\vec{r}_p$ where the perigee magnitude r_p is fixed as input. The vectors $\vec{r}_p$ and $\vec{R}_I$ then determine a unique conic for the Earth-LSI phase (see LUNCON). Let the state at the LSI on that conic (relative to Earth-equatorial coordinates) be denoted by $\vec{R}_I$, $\vec{V}_I$. The state relative to the moon may then be computed as

$$\begin{aligned} \vec{r}_I &= \vec{R}_I - \vec{R}_{MQ} \\ \vec{v}_I &= \vec{V}_I - \vec{V}_{MQ} \end{aligned} \tag{6}$$

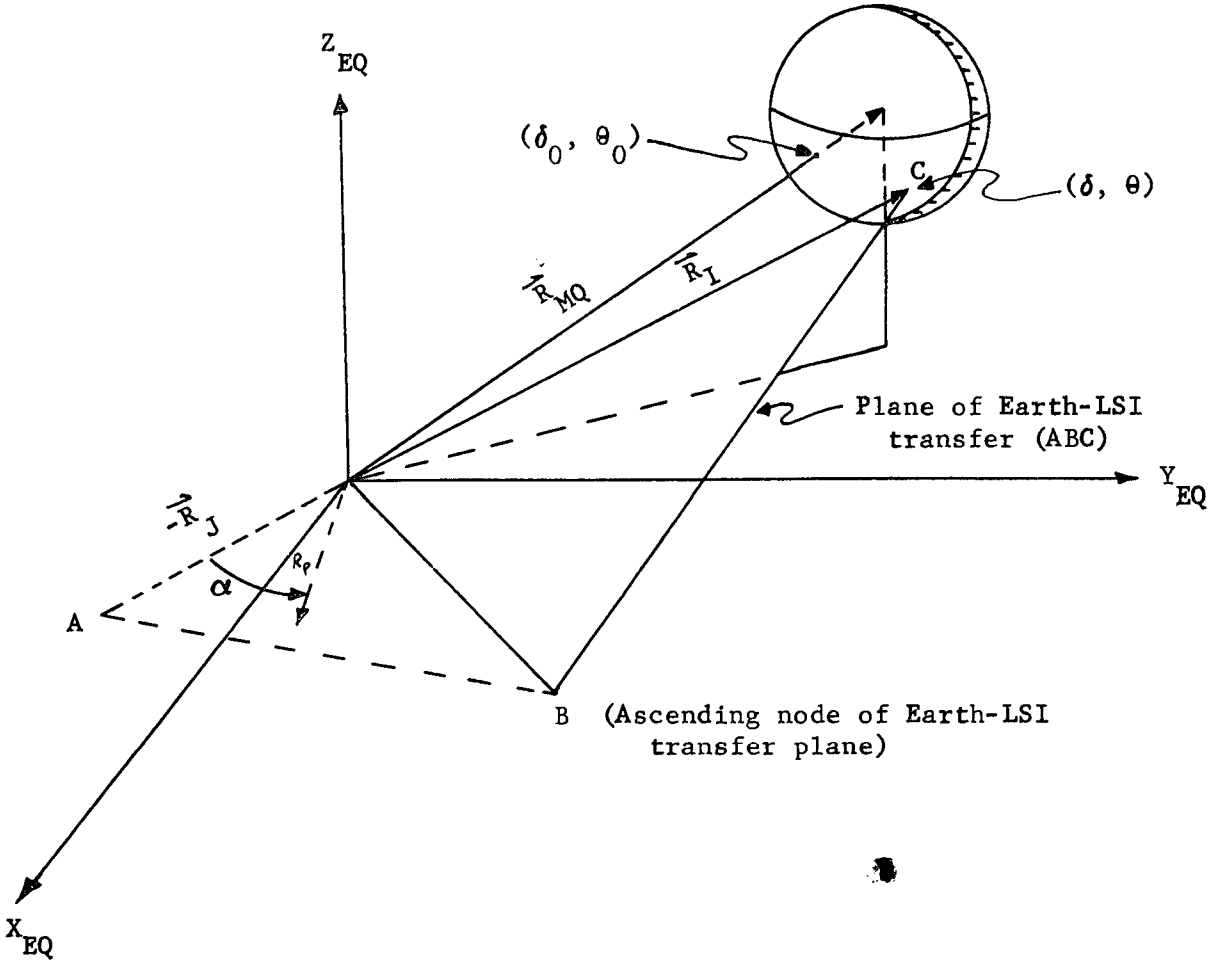


Figure 1. Lunar Patched Conic Targeting

Thus the elements relative to the moon may be computed from standard conic formula. The three angles (δ, θ, α) form a set of independent controls to be varied to meet the three constraints (r_{CA}, i_{CA}, a_{CA}). The controls are depicted in Figure 1.

LUNTAR uses the standard Newton-Raphson algorithm to refine the controls to meet the constraints. This targeting is done in two stages. In the first stage the controls δ and θ are held fixed at the bullseye point (δ_0, θ_0) while α is varied until the semi-major axis target a_{CA} is met. Then all three controls are varied to satisfy the three target constraints. The preliminary targeting of a_{CA} is essential to the success of the procedure. Once the in initial targeting is completed, the semi major axis of future

iterations in the second stage will not vary much from the target value a_{CA}. For such iterates the excess hyperbolic velocity at the moon will be generally constant. This permits the substitution of the auxiliary impact plane parameters B•T and B•R for the less linear parameters of r_{CA} and i_{CA} (see IMPACT). In LUNTAR the impact plane parameters are referenced to the LQ system.

The procedure may now be described in detail. Suppose that in the first stage of targeting the current value of α is α_K. Using the controls $(\alpha_K, \delta_0, \theta_0)$ the resulting semi-major axis is found to be a_K (LUNCON). A perturbed value for the first control is then used $(\alpha_K + \Delta\alpha, \delta_0, \theta_0)$ producing a perturbed value of semi-major axis $(a_K + \Delta a)$. The (k+1)st value of α is then given by the standard numerical differencing approximation

$$\alpha_{K+1} = \alpha_K + \frac{\Delta\alpha}{\Delta a}(a_{CA} - a_K) \tag{7}$$

The second stage of the targeting of the lunar patched conic uses the vector analogue of the above procedure. The current iterate $(\alpha_K, \delta_K, \theta_K)$ is input to LUNCON to obtain the current target values $(a_K, B\cdot T_K, B\cdot R_K)$. The target values B•T and B•R are determined from subroutine IMPACT and the errors of the kth iterate are computed (e_a, e_{BT}, e_{BR}). If all three errors are within tolerances, the procedure is terminated. Otherwise the sensitivity matrix $\emptyset$ is computed by numerical differencing as in the first stage

$$\emptyset = \begin{bmatrix} \frac{\Delta a_\alpha}{\Delta\alpha} & \frac{\Delta a_\delta}{\Delta\delta} & \frac{\Delta a_\theta}{\Delta\theta} \\ \frac{\Delta B\cdot T_\alpha}{\Delta\alpha} & \frac{\Delta B\cdot T_\delta}{\Delta\delta} & \frac{\Delta B\cdot T_\theta}{\Delta\theta} \\ \frac{\Delta B\cdot R_\alpha}{\Delta\alpha} & \frac{\Delta B\cdot R_\delta}{\Delta\delta} & \frac{\Delta B\cdot R_\theta}{\Delta\theta} \end{bmatrix} \tag{8}$$

The inverse of $\emptyset$ is the targeting matrix. The k+1 iterate is then defined to be

$$\begin{bmatrix} \alpha \\ \delta \\ \theta \end{bmatrix}_{K+1} = \begin{bmatrix} \alpha \\ \delta \\ \theta \end{bmatrix}_{K} + \phi^{-1} \begin{bmatrix} a_{CA} - a_{K} \\ B \cdot T - B \cdot T_{K} \\ B \cdot R - B \cdot R_{K} \end{bmatrix} \qquad (9)$$

This procedure is repeated until convergence is achieved.

MEAN Analysis

Subroutine MEAN propagates and updates actual estimation error means over the time interval $[t_k, t_{k+1}]$ separating two successive measurements or events. The equations programmed in MEAN are independent of the filter algorithm employed to generate gain matrices. Gain matrices are assumed to have been computed during a prior call to subroutine GNAVM. The propagation equations programmed in MEAN are also used to propagate actual deviation means over the time interval separating two successive guidance events. The update equations, of course, are not used in this situation.

The actualestimation errors for position/velocity state, solve-for parameters, dynamic consider parameters, measurement consider parameters, and ignore parameters are defined, respectively, by the following:

$$\tilde{x}_{k+1} = \hat{x}_{k+1} - x_{k+1} \tag{1}$$

$$\tilde{x}_{s_{k+1}} = \hat{x}_{s_{k+1}} - x_{s_{k+1}} \tag{2}$$

$$\tilde{u}_{k+1} = \hat{u}_{k+1} - u_{k+1} = - u_o \tag{3}$$

$$\tilde{v}_{k+1} = \hat{v}_{k+1} - v_{k+1} = - v_o \tag{4}$$

$$\tilde{w}_{k+1} = \hat{w}_{k+1} - w_{k+1} = - w_o \tag{5}$$

where (^) indicates estimated values, and x, x_s, u_o, v_o, and w_o are the actual deviations from nominal.

Only the means of $\tilde{x}$ and $\tilde{x}_s$ are paopagated and updated since the means of y, x, and w are constant. The propagation equations are summarized

$$E\left[\tilde{x}^-_{k+1}\right] = \Phi \cdot E\left[\tilde{x}^+_k\right] + \theta_{xx_s} \cdot E\left[\tilde{x}^+_{s_k}\right] - \theta_{xu}\,\bar{u}_o - \theta_{xw}\,\bar{w}_o \tag{6}$$

$$E\left[\tilde{x}^-_{s_{k+1}}\right] = E\left[\tilde{x}^+_{s_k}\right] \tag{7}$$

where Φ, θ_{xx_s}, θ_{xu}, and θ_{xw} are state transition matrices over $[t_k, t_{k+1}]$.

Before the means of x and x_s can be updated at a measurement, the mean of the measurement residual $_{k+1}$ must first be computed using

$$E\left[\varepsilon_{k+1}\right] = - H \cdot E\left[\tilde{x}^{-}_{k+1}\right] - M \cdot E\left[\tilde{x}^{-}_{s_{k+1}}\right] + G\bar{u}_o + L\bar{v}_o + N\bar{w}_o \qquad (8)$$

where H, M, G, L, and N are observation matrix partitions.

The update equations are summarized as:

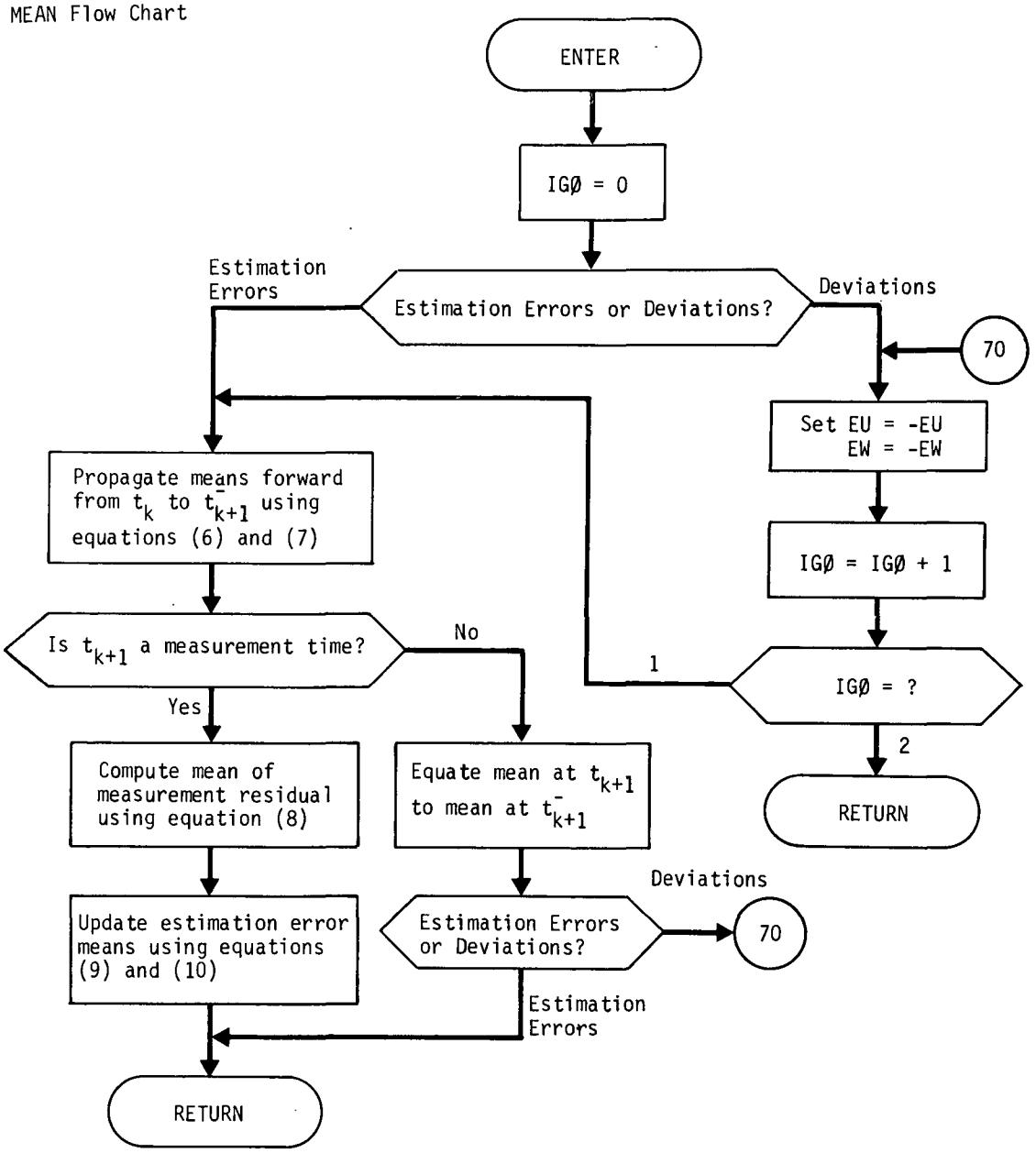
$$E\left[\tilde{x}^{+}_{k+1}\right] = E\left[\tilde{x}^{-}_{k+1}\right] + K_{k+1} \cdot E\left[\varepsilon_{k+1}\right] \qquad (9)$$

$$E\left[\tilde{x}^{+}_{k+1}\right] = E\left[\tilde{x}^{-}_{s_{k+1}}\right] + S_{k+1} \cdot E\left[\varepsilon_{k+1}\right] \qquad (10)$$

where K_{k+1} and S_{k+1} are the filter gain matrices.

To propagate actual deviation means requires that x and x_s be replaced by $\tilde{x}$ and $\tilde{x}_s$, respectively, in equations (6) and (7), and that the minus signs in equations (6) be replaced with plus signs.

MEAN Flow Chart



MENØ Analysis

The linearized observation equation employed by the navigation process is given by

$$\delta Y_k = H_k^A \, \delta X_k^A + \eta_k$$

where δY_k is the measurement deviation from the nominal measurement, H_k^A is the augmented observation matrix, δX_k^A is the augmented state deviation from the nominal augmented state, and η_k is the assumed measurement noise.

The function of subroutine MENØ is to compute the assumed measurement noise covariance matrix

$$R_k = E\left[\eta_k \, \eta_k^T\right]$$

if ICØDE = 0. The constant measurement noise variances associated with all available measurement types are stored in the vector MNCN. Subroutine MENØ selects the appropriate element from this vector to construct R_k.

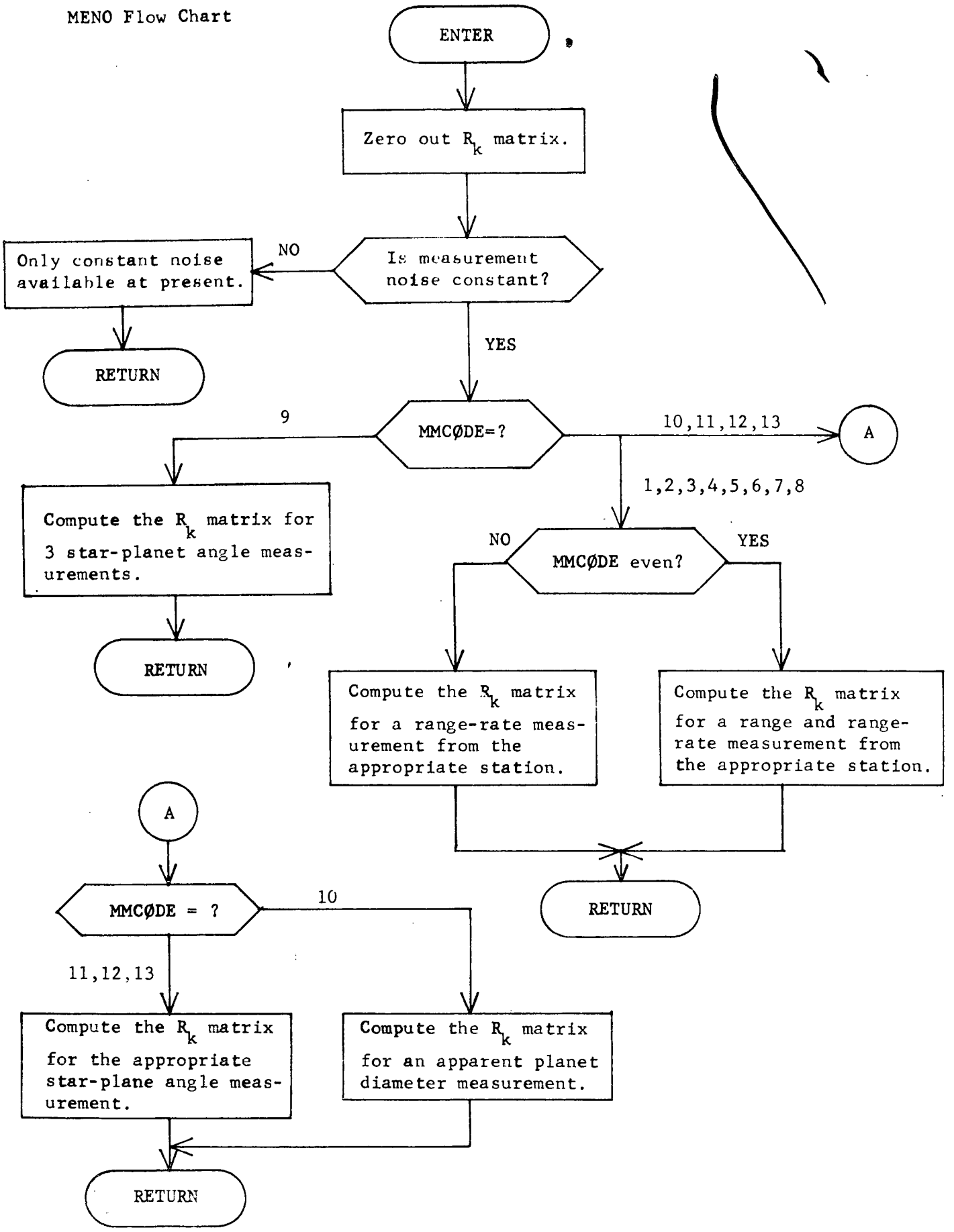
If ICØDE $\neq$ 0 the actual measurement noise covariance matrix

$$R_k' = E\left[\eta_k' \, \eta_k'^T\right]$$

where η_k' is the actual measurement noise, is computed instead. In this case subroutine MENØ selects the appropriate actual measurement noise variances from the vector GMNCN to construct R_k'.

The accompanying flow chart indicates the computational flow for computing R_k. An identical procedure is used to compute R_k'.

MENO Flow Chart



MENØS Analysis

The linearized observation equation employed by the navigation process is given by

$$\delta Y_k = H_k^A \delta X_k^A + \eta_k$$

where δY_k is the measurement deviation from the nominal measurement, H_k^A is the augmented observation matrix, δX_k^A is the augmented state deviation from the nominal augmented state, and η_k is the assumed measurement noise. The actual measurement Y_k^a is given by

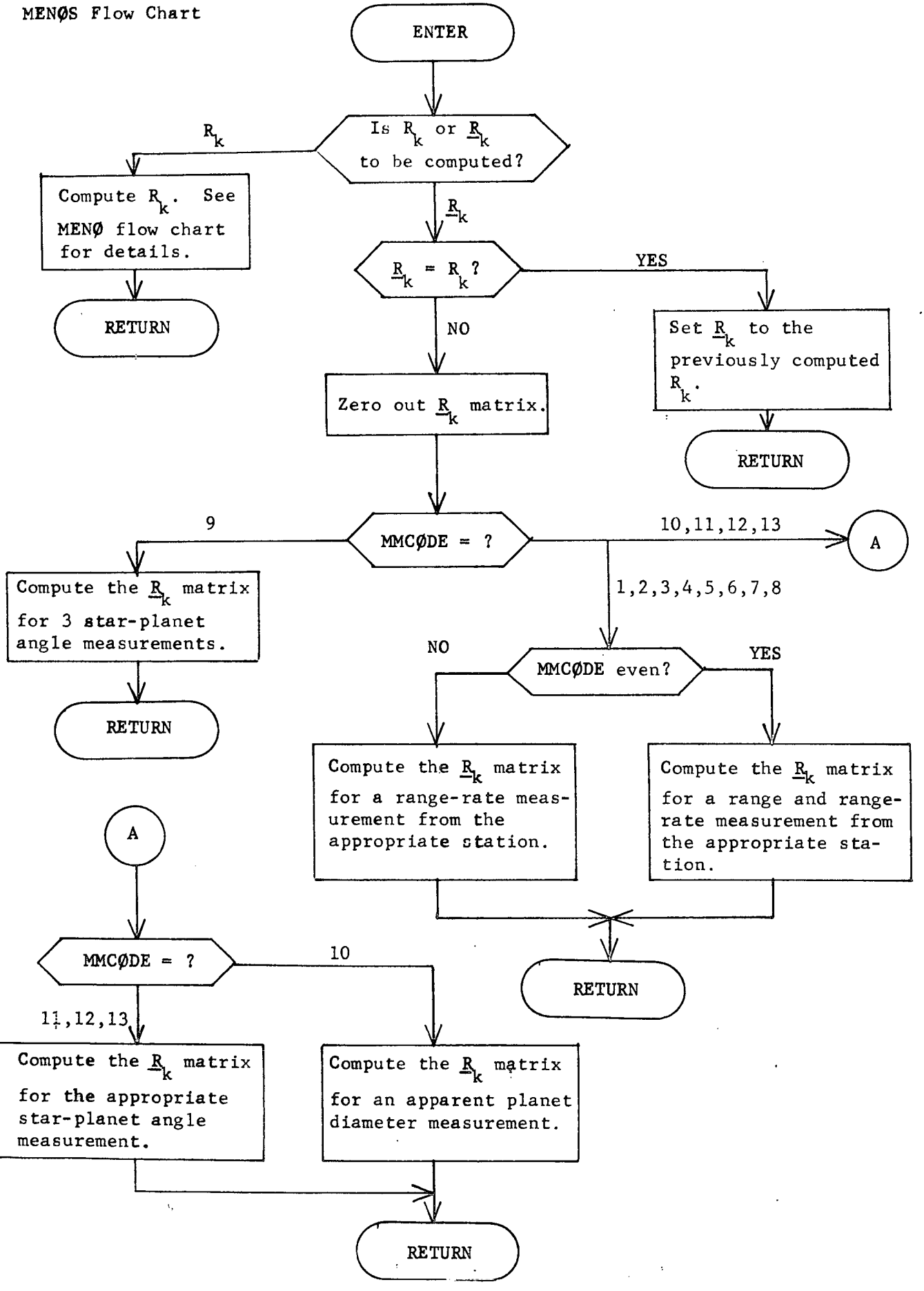
$$Y_k^a = \underline{Y}_k + b_k + \nu_k$$

where $\underline{Y}_k$ is the ideal measurement, which would be made in the absence of instrumentation errors, b_k is the actual measurement bias, and ν_k represents the actual measurement noise.

Subroutine MENOS performs two functions. It's first function, which is identical to that of subroutine MENO, is to compute the measurement noise covariance matrix R_k which describes the statistics of noise η_k. The constant variances for the assumed measurement noises associated with all available measurement devices are stored in the vector MNCN. Subroutine MENOS selects the appropriate elements from this vector to construct the measurement noise covariance matrix R_k.

The second function of MENOS is to compute the measurement noise covariance matrix $\underline{R}_k$ which describes the statistics of the actual noise ν_k. The constant variances for the actual measurement noises associated with all available measurement devices are stored in the vector AVARM. Subroutine AVARM selects the appropriate elements from this vector to construct the measurement noise covariance matrix $\underline{R}_k$.

MENØS Flow Chart



MINIQ Analysis

Subroutine MINIQ computes the execution error covariance matrix and the actual execution error associated with the spin-release of a miniprobe. The actual execution error is computed only when MINIQ is used in the simulation program SIMUL.

The velocity increment imparted to the ith probe at release is given by

$$\Delta\vec{V}^i = \vec{\omega} \times \vec{\ell}^i \qquad (1)$$

where $\vec{\omega}$ is the spin vector and $\vec{\ell}^i$ denotes the position of the ith probe relative to the primary vehicle. Referred to the $\hat{u}$ $\hat{v}$ $\hat{h}$ coordinate system, which is defined in subroutine TPRTRG, equation (1) becomes

$$\Delta\vec{V}^i = \ell\,\omega\,\cos\left\{\left|\phi + (i-1)\,\frac{2\pi}{3}\right|\hat{u} + \sin\left|\phi + (i-1)\,\frac{2\pi}{3}\right|\hat{v}\right\} \qquad (2)$$

where $\left|\phi + (i-1)\,\frac{2\pi}{3}\right|$ is the roll release angle of the ith probe, and $\hat{u}$ and $\hat{v}$ are unit vectors.

Define $\vec{p} = (\omega, \ell, \alpha, \delta, \phi)$ as the release execution parameter vector, where ω is the spin rate magnitude, ℓ is the boom length, α is the right ascension of the spin axis, δ is the declination of the spin axis, and ϕ is the roll release angle. Then the release execution error can be written as

$$\delta\Delta\vec{V}^i = \frac{\partial\Delta\vec{V}^i}{\partial\vec{p}}\,\delta\vec{p} \qquad (3)$$

where $\delta\vec{p}$ represents the error in the release parameter vector. The jth component of $\delta\Delta\vec{V}^i$ is given by

$$\delta\Delta V^i_j = \sum_{m=1}^{5} \frac{\partial\Delta V^i_j}{\partial p_m}\,\delta p_m \qquad (4)$$

The execution error covariance matrix is defined as

$$\tilde{Q}^i = E\left[\delta\Delta\vec{V}^i \cdot \delta\Delta\vec{V}^{i^T}\right] \tag{5}$$

and the element $\tilde{Q}^i_{jk}$ of matrix $\tilde{Q}^i$ is given by

$$\tilde{Q}^i_{jk} = E\left[\sum_{m=1}^{5} \frac{\partial\Delta V^i_j}{\partial p_m}\delta p_m \cdot \sum_{n=1}^{5} \frac{\partial\Delta V^i_k}{\partial p_n}\delta p_n\right] . \tag{6}$$

Assuming $E\left[\delta_{p_m}\ \delta_{p_n}\right] = 0$ for $m \neq n$, equation (6) reduces to

$$\tilde{Q}^i_{jk} = \sum_{m=1}^{5} \frac{\partial\Delta V^i_j}{\partial p_m} \cdot \frac{\partial\Delta V^i_k}{\partial p_n} \cdot E\left[\delta p_m^2\right] . \tag{7}$$

The partial derivatives required for evaluation of equations (4) and (7) are summarized as:

$$\frac{\partial\Delta\vec{V}^i}{\partial\omega} = \ell\left\{\cos\left|\phi + (i-1)\frac{2\pi}{3}\right|\hat{u} + \sin\left|\phi + (i-1)\frac{2\pi}{3}\right|\hat{v}\right\} \tag{8}$$

$$\frac{\partial\Delta\vec{V}^i}{\partial\ell} = \omega\left\{\cos\left|\phi + (i-1)\frac{2\pi}{3}\right|\hat{u} + \sin\left|\phi + (i-1)\frac{2\pi}{3}\right|\hat{v}\right\} \tag{9}$$

$$\frac{\partial\Delta\vec{V}^i}{\partial\alpha} = \ell\omega\left\{\cos\left|\phi + (i-1)\frac{2\pi}{3}\right|\frac{\partial\hat{u}}{\partial\alpha} + \sin\left|\phi + (i-1)\frac{2\pi}{3}\right|\frac{\partial\hat{v}}{\partial\alpha}\right\} \tag{10}$$

$$\frac{\partial\Delta\vec{V}^i}{\partial\delta} = \ell\omega\left\{\cos\left|\phi + (i-1)\frac{2\pi}{3}\right|\frac{\partial\hat{u}}{\partial\delta} + \sin\left|\phi + (i-1)\frac{2\pi}{3}\right|\frac{\partial\hat{v}}{\partial\delta}\right\} \tag{11}$$

$$\frac{\partial\Delta\vec{V}^i}{\partial\phi} = \ell\omega\left\{-\sin\left|\phi + (i-1)\frac{2\pi}{3}\right|\hat{u} + \cos\left|\phi + (i-1)\frac{2\pi}{3}\right|\hat{v}\right\} \tag{12}$$

where

$$\hat{u} = (\sin \alpha, -\cos \alpha, 0) \tag{13}$$

$$\hat{v} = (\sin \delta \cos \alpha, \sin \delta \sin \alpha, -\cos \delta) \tag{14}$$

$$\frac{\partial \hat{u}}{\partial \alpha} = (\cos \alpha, \sin \alpha, 0) \tag{15}$$

$$\frac{\partial \hat{v}}{\partial \alpha} = (-\sin \delta \sin \alpha, \sin \delta \cos \alpha, 0) \tag{16}$$

$$\frac{\partial \hat{u}}{\partial \delta} = (0, 0, 0) \tag{17}$$

$$\frac{\partial \hat{v}}{\partial \delta} = (\cos \delta \cos \alpha, \cos \delta \sin \alpha, \sin \delta) \tag{18}$$

when referred to the ecliptic coordinate system.

MØMENT Analysis

Subroutine MØMENT transforms an arbitrary 2nd moment matrix $E[xy^T]$ into a correlation matrix and, if $x = y$, into a vector of standard deviations. The transformation consists of two steps:

1) Transform $E[xy^T]$ into the covariance matrix

$$\text{cov}\ (x,y) = E[xy^T] - E[x] \cdot E[y^T];$$

2) Transform cov (x,y) into the correlation matrix having correlation coefficients

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i\ \sigma_j} \qquad i \neq j$$

where

$$\sigma_{ij} = E[x_i\ y_j]$$

$$\sigma_i = E[x_i^2]^{\frac{1}{2}}$$

$$\sigma_j = E[y_j^2]^{\frac{1}{2}} \quad .$$

Subroutine MØMENT writes out the correlation matrix and, if they exist, the standard deviations. Subroutine MØMENT can also compute and write out the eigenvalues, eigenvectors, and hyperellipsoid of cov (x,y) if x = y.

MPPRØP Analysis

MPPRØP serves the single purpose of providing a time history of the targeted main probe trajectory from release to the appropriate stopping condition. The process of providing such a time history is classified as a separate type of guidance event in the GIDANS execution logic. Although the main probe propagation event is currently only applied in the NØMNAL program to propagate the main probe once it begins to deviate from the bus trajectory due to some guidance maneuver on the latter, it could be used to treat any branched trajectory.

On being called by GIDANS, MPPRØP first prints out the title "Main Probe Propagation Event" followed by the heading "Main Probe Approach Trajectory." Next it stores the current spacecraft heliocentric ecliptic state and the VMP trajectory condition and instrument flags so that they can be returned to after the event. Then the VMP flags are set for propagating the main probe. Codes are used to stop the trajectory after 90 days of propagation or at closest approach but not at the sphere of influence. The VMP stopping condition of impacting the planet is slightly modified. Rather than using NØMNAL's own value of target-planet radius, MPPRØP transmits to VMP the radius of the probe sphere input by the user and applied throughout probe targeting. The print routine is activated and the print increments are set at 5-day and 100-integration steps. Next VMP is called to propagate and print the trajectory until a stopping condition is reached. Finally, the original spacecraft state and the VMP flags are restored before returning to GIDANS.

MULCØN Analysis

The equations of motion of a spacecraft traveling under the influence of the earth and moon may be written

$$\ddot{\vec{r}}_E = -\frac{\mu_E \vec{r}_E}{r_E^3} - \frac{\mu_M \vec{r}_M}{r_M^3} - \frac{\mu_M \vec{R}_{EM}}{r_{EM}^3} \tag{1}$$

where $\vec{r}_E$, $\vec{r}_M$, $\vec{R}_{EM}$ are the position vectors of the spacecraft-to-earth, the spacecraft-to-moon, and the moon-to-earth respectively and μ_E, μ_M are the gravitational constants of the earth and moon respectively.

The multi-conic approximation of the solution to (1) proceeds as follows. Let $\vec{r}_{E,k}$, $\vec{v}_{E,k}$ be the geocentric state at some time t_k. This state is propagated by conic formulae to obtain an estimate of the geocentric state at time $t_{k+1} = t_k + \Delta t$ given by $\vec{r}_{E,k+1}'$, $\vec{v}_{E,k+1}'$.

To account for the third term perturbations, the state of the moon relative to the earth at the two timepoints is computed, denoted by $(\vec{R}_{EM,k}, \vec{V}_{EM,k})$ and $(\vec{R}_{EM,k+1}, \vec{V}_{EM,k+1})$. The average value of this acceleration is then determined from

$$\vec{A} = -\frac{\mu_M}{2}\left[\frac{\vec{R}_{EM,k}}{R_{EM,k}^3} + \frac{\vec{R}_{EM,k+1}}{R_{EM,k+1}^3}\right] \tag{2}$$

The corrected geocentric state is then given by

$$\begin{aligned} \vec{r}_{E,k+1}'' &= \vec{r}_{E,k+1}' + \frac{1}{2}\vec{A}\,(\Delta t)^2 \\ \vec{v}_{E,k+1}'' &= \vec{v}_{E,k+1}' + \vec{A}\,\Delta t \end{aligned} \tag{3}$$

The effect of the direct lunar perturbations is then added. The state of the spacecraft relative to the moon is first computed

$$\begin{aligned} \vec{r}_{M,k+1}' &= \vec{r}_{E,k+1}'' - \vec{R}_{EM,k+1} \\ \vec{v}_{M,k+1}' &= \vec{v}_{E,k+1}'' - \vec{V}_{EM,k+1} \end{aligned} \tag{4}$$

This state is then propagated linearly backwards in time over the time interval Δt to obtain

$$\vec{r}'_{M,k} = \vec{r}'_{M,k+1} - \vec{v}_{M,k+1}\,\Delta t \qquad (5)$$

$$\vec{v}'_{M,k} = \vec{v}'_{M,k+1}$$

This state is now propagated forward in a selenocentric conic to obtain a final state relative to the moon ($\vec{r}_{M,k+1}$, $\vec{v}_{M,k+1}$). The geocentric state of the spacecraft at time t_{k+1} after considering all terms of (1) is then given by

$$\vec{r}_{E,k+1} = \vec{r}_{M,k+1} + \vec{R}_{EM,k+1}$$

$$\vec{v}_{E,k+1} = \vec{v}_{M,k+1} + \vec{V}_{EM,k+1} \qquad (6)$$

This completes one cycle of the multi-conic propagation.

The multi-conic propagation proceeds until an input final time is reached or until the selenocentric conic passes through pericynthion.

Reference: Byrnes, D. V. and Hooper, H. L., Multi-Conic: A Fast and Accurate Method of Computing Space Flight Trajectories, AAS/AIAA Astrodynamics Conference, Santa Barbara, Cal., 1970, AIAA Paper 70-1062.

MULTAR Analysis

Let the earth equatorial state of the probe at the LSI as computed from the patched conic targeting be denoted $\vec{r}_{LS}$, $\vec{v}_{LS}$. Subroutine CAREL is called to compute the conic elements and conic time from perigee Δt based on the geocentric conic. The time of injection is then computed as

$$t_{TLI} = t_{SI} - \Delta t \tag{1}$$

The position and velocity of the probe at t_{TLI} is given by the state along the conic at perigee (true anomaly of zero) and determined by ELCAR to be $\vec{r}_{TLI}$, $\vec{v}_{TLI}$. If Φ_{ECEQ} is the transformation matrix from the EC (earth ecliptic) to the EQ (earth equatorial) system, then the patched conic injection state in EC coordinates is

$$\begin{aligned} \vec{r}_I &= \Phi^T_{ECEQ}\, \vec{r}_{TLI} \\ \vec{v}_I &= \Phi^T_{ECEQ}\, \vec{v}_{TLI} \end{aligned} \tag{2}$$

Since the earth is revolving about the E-M barycenter in time, the EC injection state must be rotated if an earlier or later injection time is to be used. The necessary rotation matrix may be easily computed through the introduction of the R-T-W coordinate system. Let the state of the earth at some time t_k in BC (barycentric ecliptic) coordinates be denoted $\vec{R}_k$, $\vec{V}_k$. Construct the $\hat{R}$-$\hat{T}$-$\hat{W}$ system at that point as

$$\hat{R}_k = \frac{\vec{R}_k}{R_k} \qquad \hat{W}_k = \frac{\vec{R}_k \times \vec{V}_k}{\left|\vec{R}_k \times \vec{V}_k\right|} \qquad \hat{T}_k = \hat{W}_k \times \hat{R}_k \tag{3}$$

The transformation matrix from the $\hat{R}_k$-$\hat{T}_k$-$\hat{W}_k$ system to the ecliptic system to the ecliptic system is then given by

$$\Phi_k = \left[\begin{array}{c|c|c} \hat{R}_k & \hat{T}_k & \hat{W}_k \end{array}\right] \tag{4}$$

At a time t_{k+1} the state of the earth in BC coordinates is given by $\vec{R}_{k+1}$, $\vec{V}_{k+1}$ and the transformation from the $\hat{R}_{k+1}$ - $\hat{T}_{k+1}$ - $\hat{W}_{k+1}$ system to ecliptic coordinates is given by Φ_{k+1} in accordance with (4). Injection

states at times t_k and t_{k+1} will be called "equivalent" if they are identical when expressed in the pertinent $\hat{R}$-$\hat{T}$-$\hat{W}$ system. Therefore if ($\vec{r}_k$, $\vec{v}_k$) is the injection state in EC coordinates at time t_k, the equivalent state in EC coordinates at t_{k+1} is given by

$$\begin{bmatrix} \vec{r}_{k+1} \\ \vec{v}_{k+1} \end{bmatrix} = \begin{bmatrix} \Psi_{k+1,k} & 0 \\ 0 & \Psi_{k+1,k} \end{bmatrix} \begin{bmatrix} \vec{r}_k \\ \vec{v}_k \end{bmatrix} \tag{5}$$

where the rotation matrix Ψ is defined by

$$\Psi_{k+1,k} = \Phi_{k+1} \Phi_k^T \tag{6}$$

The targeting algorithm used by MULTAR may now be described. Let the injection state in EC coordinates on the k=th iteration be denoted (t_k, $\vec{r}_k$, $\vec{v}_k$). This state is propagated forward using the multi-conic propagator MULCØN to determine a final state $\vec{r}_M$, $\vec{v}_M$ to the moon in ecliptic coordinates. IMPACT is then called to compute the $B\cdot T_k$, $B\cdot R_k$ and $t_{CA,k}$ actually achieved on the trajectory and the target values of $B^*\cdot T_k$, $B^*\cdot R_k$ required to satisfy the i_{CA} and r_{CA} constraints. The semi-major axis a_k of the k-th iterate is computed from the conic formula

$$a = r_M \left(2 - \frac{r_M v_M^2}{\mu_M} \right)^{-1} \tag{7}$$

Errors in the four target conditions

$$\Delta\tau = \begin{bmatrix} \Delta a \\ \Delta B\cdot T \\ \Delta B\cdot R \\ \Delta t_{CA} \end{bmatrix} = \begin{bmatrix} a - a^* \\ B\cdot T_k - B^*\cdot T_k \\ B\cdot R_k \quad B^*\cdot R_k \\ t_{CA,k} - t_{CA}^* \end{bmatrix} \tag{8}$$

if the error in each parameter is less than the allowable tolerance, the process stops.

If convergence has not been achieved a Newton-Raphson iteration is entered. The four controls are $\vec{v}_{k_x}$, $\vec{v}_{k_y}$, $\vec{v}_{k_z}$, and t_k. For the velocity components a perturbation Δv is added to the pertinent component while the rest of the injection state is held constant before propagating with the multi-conic. For the time perturbation, the rotation matrix Ψ_Δ corresponding to the perturbed time $t_k + \Delta t$ (6) is first computed. The injection state used in the perturbed propagation for time is then $\left[t_k + \Delta t, \Psi_\Delta \vec{r}_k, \Psi_\Delta \vec{v}_k\right]$. A sensitivity matrix is computed using the results of the numerical differencing:

$$\chi = \begin{bmatrix} \frac{\Delta a_x}{\Delta v_x} & \frac{\Delta a_y}{\Delta v_y} & \cdots & \\ \frac{\Delta B\ T_x}{\Delta v_x} & \vdots & & \\ \frac{\Delta B\ R_x}{\Delta v_x} & & & \\ \frac{\Delta t_{CA_x}}{\Delta v_x} & & & \frac{\Delta t_{CA_t}}{\Delta t} \end{bmatrix} \tag{9}$$

where in the term $\frac{\Delta\alpha_\beta}{\Delta\beta}$, $\Delta\alpha_\beta$ is the change in the α target parameter produced by the variation of the β control component and $\Delta\beta$ is the change in the β control component. The k+1 iterate controls are then given by

$$\Delta C = \begin{bmatrix} \delta v_x \\ \delta v_y \\ \delta v_z \\ \delta t \end{bmatrix} = \chi^{-1} \Delta\tau \tag{10}$$

The k+1 injection state is then computed by first determining the injection state after rotation due to the change in injection time and then adding the injection velocity corrections

$$\begin{aligned} t_{k+1} &= t_k + \delta t \\ \vec{r}_{k+1} &= \Psi_\delta \vec{r}_k \\ \vec{v}_{k+1} &= \Psi_\delta \vec{v}_k + \delta\vec{v} \end{aligned} \tag{11}$$

The iteration process is repeated until tolerable errors are met. The converged injection state is then integrated in the virtual mass trajectory.

MUND Analysis

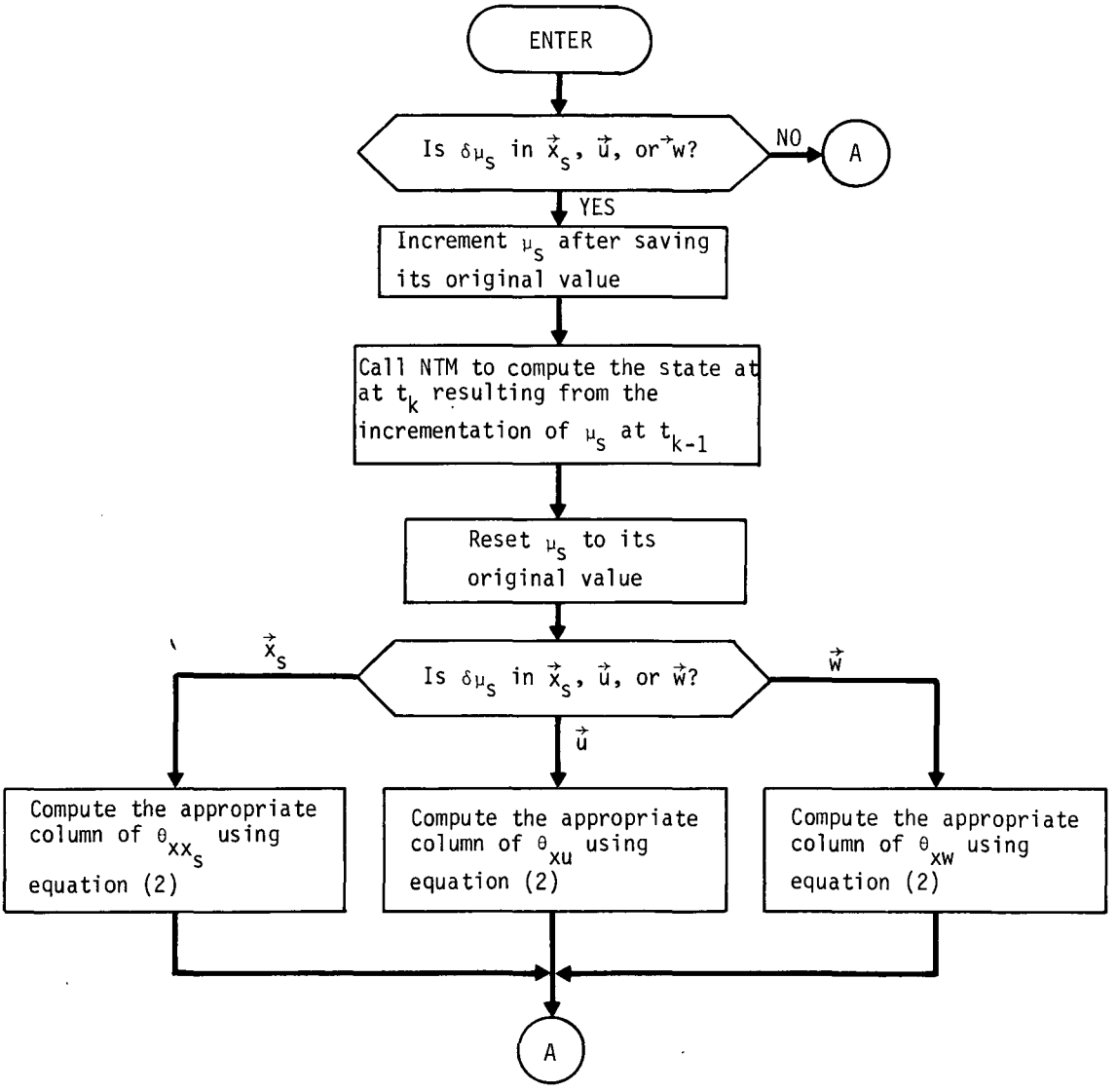
The nonlinear equations of motion of the spacecraft can be written symbolically as

$$\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{\mu}, t) \tag{1}$$

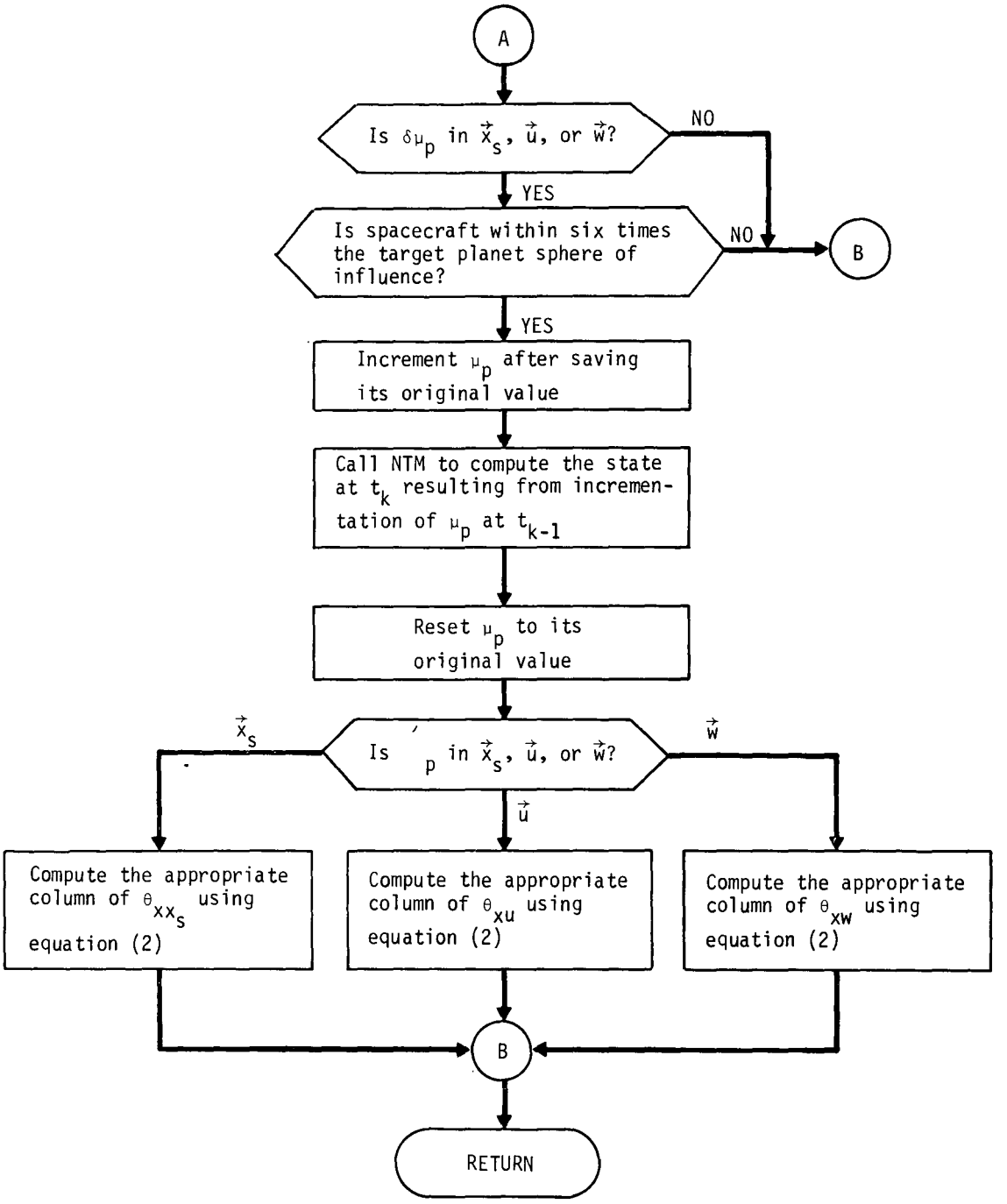
where $\vec{x}$ is the spacecraft position/velocity state and $\vec{\mu}$ is a vector composed of the gravitational constants of the sun and the target planet.

Suppose we wish to use numerical differencing to compute those columns of θ_{xx_s}, θ_{xu}, and θ_{xw} associated with gravitational constant biases included in the augmented state vector over the time interval $[t_{k-1}, t_k]$. Let $\vec{\theta}_j\ (t_k, t_{k-1})$ represent the column asasociated with the j-th gravitational constant bias. We assume we have available the nominal states $\vec{x}^*(t_{k-1})$ and $\vec{x}^*(t_k)$, which, of course, were obtained by numerically solving equation (1) using nominal $\vec{\mu}$. To obtain $\vec{\theta}_j(t_k, t_{k-1})$ we increment the j-th gravitational constant bias by the pertinent numerical differencing factor $\Delta\mu_j$ and numerically integrated equation (1) over the interval $[t_{k-1}, t_k]$ to obtain the new spacecraft state $\vec{x}_j(t_k)$, where the j-subscript on the spacecraft state indicates that it was obtained by incrementing the j-th gravitational constant bias. Then

$$\vec{\theta}_j\ (t_k, t_{k-1}) = \frac{\vec{x}_j(t_k) - \vec{x}^*(t_k)}{\Delta\mu_j} \quad . \tag{2}$$



MUND Flow Chart


A
Is $\delta\mu_p$ in $\vec{x}_s$, $\vec{u}$, or $\vec{w}$?
NO
YES
Is spacecraft within six times the target planet sphere of influence?
NO
B
YES
Increment μ_p after saving its original value
Call NTM to compute the state at t_k resulting from incrementation of μ_p at t_{k-1}
Reset μ_p to its original value
$\vec{x}_s$
Is $'_p$ in $\vec{x}_s$, $\vec{u}$, or $\vec{w}$?
$\vec{w}$
$\vec{u}$
Compute the appropriate column of θ_{xx_s} using equation (2)
Compute the appropriate column of θ_{xu} using equation (2)
Compute the appropriate column of θ_{xw} using equation (2)
B
RETURN

NAVM Analysis

The augmented deviation state vector is defined as

$$\vec{x}^A = \left[\vec{x},\ \vec{x}_s,\ \vec{u},\ \vec{v}\right]^T$$

where

$\vec{x}$ = position and velocity state (dimension 6)

$\vec{x}_s$ = solve-for parameter state (dimension n_1)

$\vec{u}$ = dynamic consider parameter state (dimension n_2)

$\vec{v}$ = measurement consider parameter state (dimension n_3)

The linearized equations of motion have form

$$\dot{\vec{x}} = F_1 \vec{x} + F_2 \vec{x}_s + F_3 \vec{u}$$

$$\dot{\vec{x}}_s = 0$$

$$\dot{\vec{u}} = 0$$

$$\dot{\vec{v}} = 0$$

and solution

$$\vec{x}_{k+1} = \Phi(k+1,k)\vec{x}_k + \theta_{xx_s}(k+1,k)\vec{x}_{s_k} + \theta_{xu}(k+1,k)\vec{u}_k + \vec{q}_k$$

$$\vec{x}_{s_{k+1}} = \vec{x}_{s_k}$$

$$\vec{u}_{k+1} = \vec{u}_k$$

$$\vec{v}_{k+1} = \vec{v}_k$$

where dynamic noise $\vec{q}_k$ has been added to the solution of $\vec{x}_{k+1}$. This solution can be written in augmented form

$$\vec{x}^A_{k+1} = \Phi^A(k+1,k)\,\vec{x}^A_k + \vec{q}^A_k$$

where the augmented state transition matrix $\Phi^A(k+1,k)$ is defined as

$$\Phi^A(k+1,k) = \begin{bmatrix} \Phi & \theta_{xx_s} & \theta_{xu} & 0 \\ 0 & I_{n_1 x\, n_1} & 0 & 0 \\ 0 & 0 & I_{n_2 x\, n_2} & 0 \\ 0 & 0 & 0 & I_{n_3 x\, n_3} \end{bmatrix}$$

Henceforth state transition matrix partitions will be written without stating the associated interval of time, which will always be assumed to be $[k, k+1]$.

The measurement deviation vector $\vec{y}$ (dimension m) is related to the augmented deviation state vector through the equation

$$\vec{y}_k = H_k^A \vec{x}_k^A + \vec{\eta}_k$$

where the augmented observation matrix is defined as

$$H_k^A = \begin{bmatrix} H_k & M_k & G_k & L_k \end{bmatrix}$$

and $\vec{\eta}_k$ is measurement noise.

The augmented state covariance matrix P_k^A can be written in terms of its partitions as

$$P_k^A = \begin{bmatrix} P_k & C_{xx_{s_k}} & C_{xu_k} & C_{xv_k} \\ C^T_{xx_{s_k}} & P_{s_k} & C_{x_s u_k} & C_{x_s v_k} \\ C^T_{xu_k} & C^T_{x_s u_k} & U_o & C_{uv_k} \\ C^T_{xv_k} & C^T_{x_s v_k} & C^T_{uv_k} & V_o \end{bmatrix}$$

Propagation and update equations for the partitions appearing in the previous equation will be written below. Equations need not be written for the consider parameter covariances U_o and V_o since these do not change with time. Also, C_{uv} will be set to zero because of the assumption that no cross-correlation exists between dynamic- and measurement-consider parameters. In the equations below, Q and R represent the covariances of the dynamic and measurement noises, respectively, defined previously. A minus superscript on covariance partitions indicates the covariance partition immediately prior to processing a measurement; a plus superscript, immediately after processing a measurement. If ICØDE indicates that a measurement is not to be processed, the update equations are bypassed. To improve numerical accuracy and avoid nonpositive definite covariance matrices, P^-, P_s^-, P^+, and P_s^+, are always symmetrized after their computation.

The propagation equation are given by

$$P^-_{k+1} = \left(\phi P^+_k + \theta_{xx_s} C^{+T}_{xx_{s_k}} + \theta_{xu} C^{+T}_{xu_k} \right) \phi^T +$$

$$C^-_{xx_{s_{k+1}}} \theta^T_{xx_s} + C^-_{xu_{k+1}} \theta^T_{xu} + Q_k$$

$$C^-_{xx_{s_{k+1}}} = \phi C^-_{xx_{s_k}} + \theta_{xx_s} P^+_{s_k} + \theta_{xu} C^{+T}_{x_s u_k}$$

$$P^-_{s_{k+1}} = P^+_{s_k}$$

$$C^-_{xu_{k+1}} = \phi C^+_{xu_k} + \theta_{xx_s} C^+_{x_s u_k} + \theta_{xu} U_o$$

$$C^-_{x_s u_{k+1}} = C^+_{x_s u_k}$$

$$C^-_{xv_{k+1}} = \phi C^+_{xv_k} + \theta_{xx_s} C^+_{x_s v_k}$$

$$C^-_{x_s v_{k+1}} = C^+_{x_s v_k} .$$

The measurement residual covariance matrix is given by

$$J_{k+1} = H_{k+1} A_{k+1} + M_{k+1} B_{k+1} + G_{k+1} D_{k+1} + L_{k+1} E_{k+1} + R_{k+1}$$

where

$$A_{k+1} = P^-_{k+1} H^T_{k+1} + C^-_{xx_{s_{k+1}}} M^T_{k+1} + C^-_{xu_{k+1}} G^T_{k+1} + C^-_{xv_{k+1}} L^T_{k+1}$$

$$B_{k+1} = P^-_{s_{k+1}} M^T_{k+1} + C^{-T}_{xx_{s_{k+1}}} H^T_{k+1} + C^-_{x_s u_{k+1}} G^T_{k+1} + C^-_{x_s v_{k+1}} L^T_{k+1}$$

$$D_{k+1} = C^{-T}_{xu_{k+1}} H^T_{k+1} + C^{-T}_{x_s u_{k+1}} M^T_{k+1} + U_o G^T_{k+1}$$

$$E_{k+1} = C^{-T}_{xv_{k+1}} H^T_{k+1} + C^{-T}_{x_s v_{k+1}} M^T_{k+1} + V_o L^T_{k+1}.$$

The covariance matrix partitions immediately after processing a measurement are given by the following update equations:

$$P^+_{k+1} = P^-_{k+1} - K_{k+1} A^T_{k+1} - A_{k+1} K^T_{k+1} + K_{k+1} J_{k+1} K^T_{k+1}$$

$$C^+_{xx_{s_{k+1}}} = C^-_{xx_{s_{k+1}}} - K_{k+1} B^T_{k+1} - A_{k+1} S^T_{k+1} + K_{k+1} J_{k+1} S^T_{k+1}$$

$$P^+_{s_{k+1}} = P^-_{s_{k+1}} - S_{k+1} B^T_{k+1} - B_{k+1} S^T_{k+1} + S_{k+1} J_{k+1} S^T_{k+1}$$

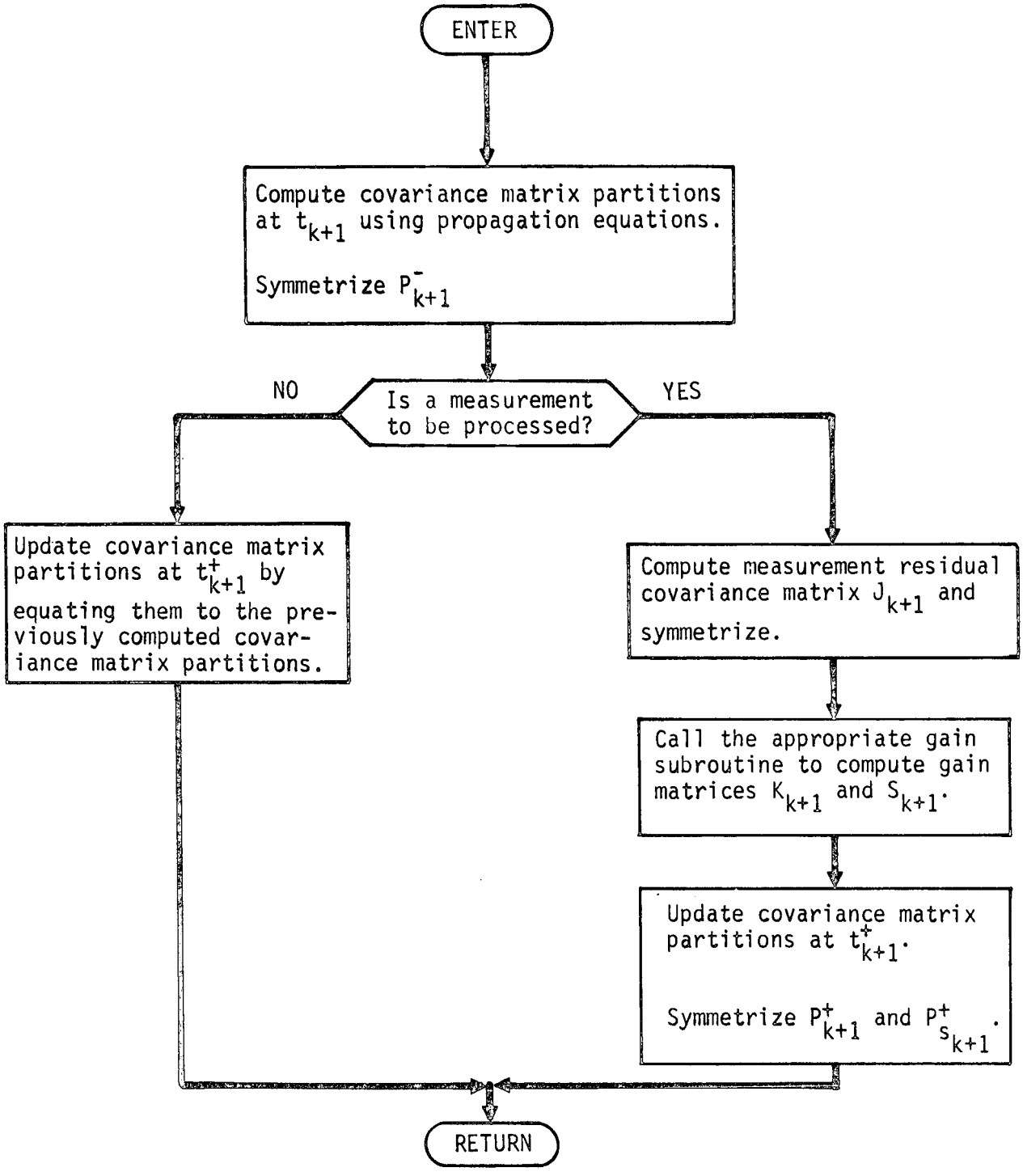
$$C^+_{xu_{k+1}} = C^-_{xu_{k+1}} - K_{k+1} D^T_{k+1}$$

$$C^+_{x_s u_{k+1}} = C^-_{x_s u_{k+1}} - S_{k+1} D^T_{k+1}$$

$$C^+_{xv_{k+1}} = C^-_{xv_{k+1}} - K_{k+1} E^T_{k+1}$$

$$C^+_{x_s v_{k+1}} = C^-_{x_s v_{k+1}} - S_{k+1} E^T_{k+1}$$

where gain matrices K_{k+1} and S_{k+1} are computed in the appropriate gain subroutine--GAIN1, if Kalman-Schmidt, GAIN2, if WLS.


ENTER
Compute covariance matrix partitions at t_{k+1} using propagation equations.
Symmetrize P^-_{k+1}
Is a measurement to be processed?
NO
YES
Update covariance matrix partitions at t^+_{k+1} by equating them to the previously computed covariance matrix partitions.
Compute measurement residual covariance matrix J_{k+1} and symmetrize.
Call the appropriate gain subroutine to compute gain matrices K_{k+1} and S_{k+1}.
Update covariance matrix partitions at t^+_{k+1}.
Symmetrize P^+_{k+1} and $P^+_{s_{k+1}}$.
RETURN

NDTM Analysis

The nonlinear equations of motion of the spacecraft can be written symbollically as

$$\dot{\vec{x}} = \vec{f}(\vec{x}, t) \tag{1}$$

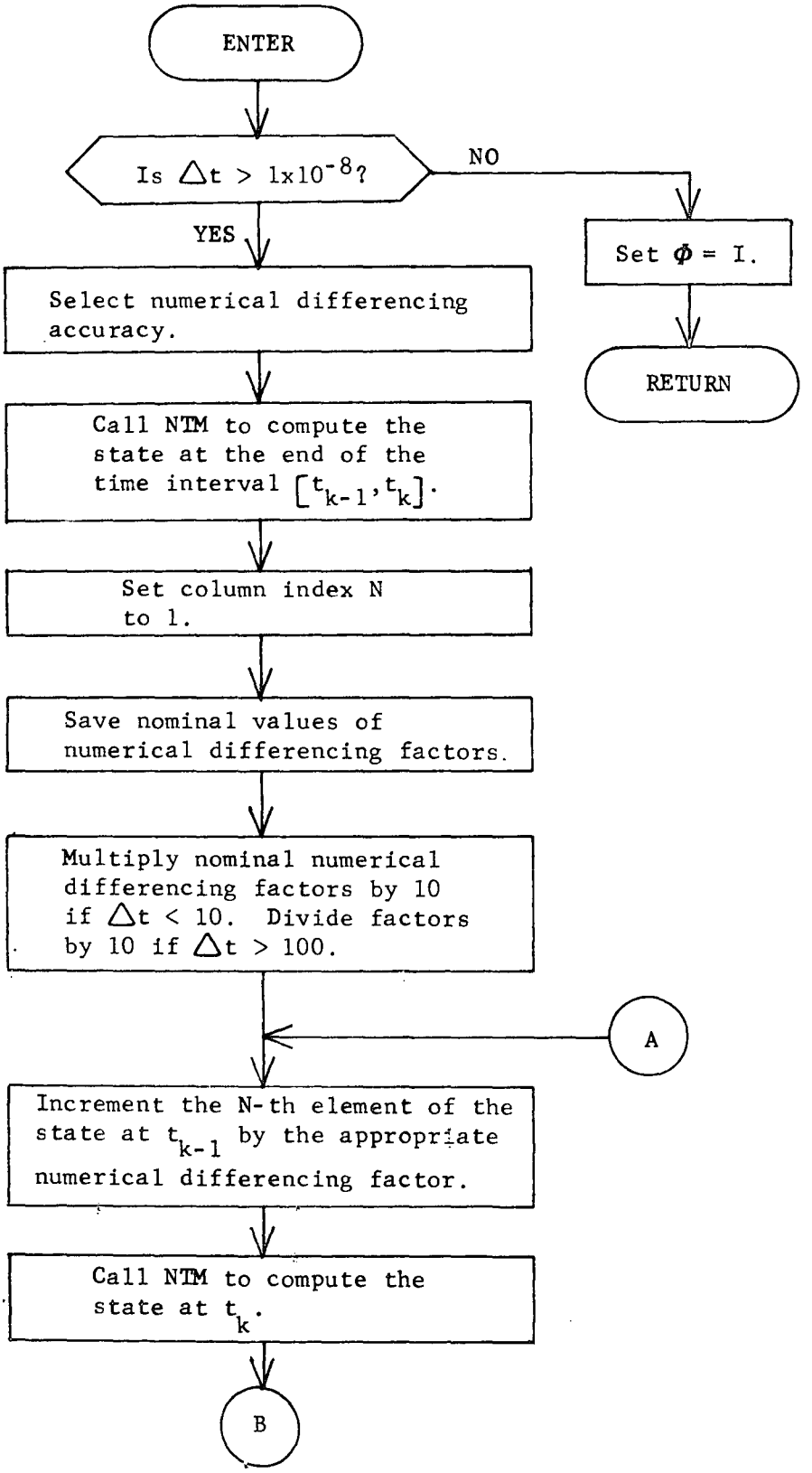
where $\vec{x}$ is the spacecraft position/velocity state.

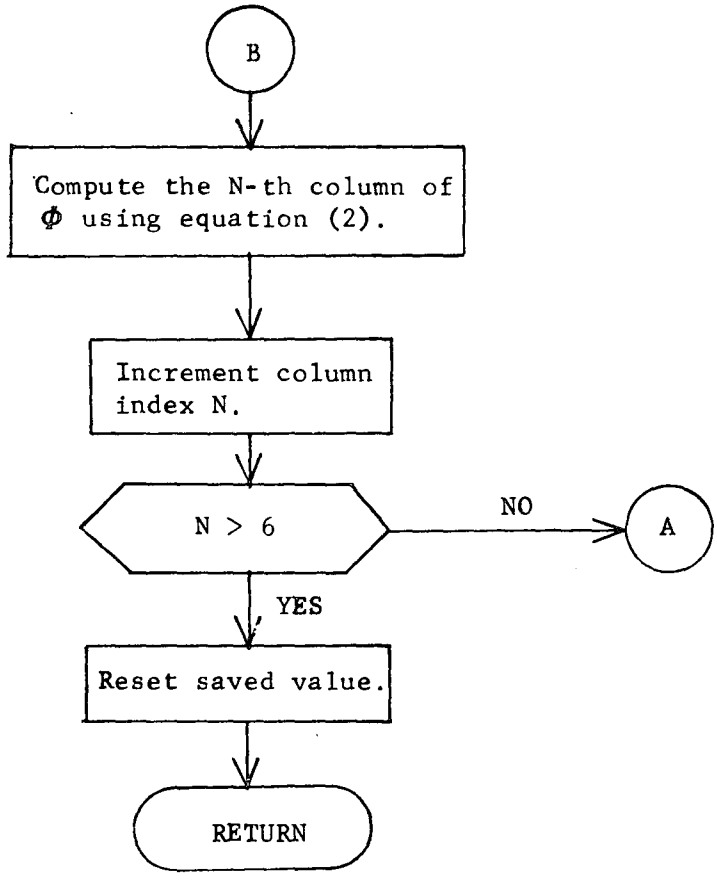
Suppose we wish to use numerical differencing to compute the state transition matrix $\Phi(t_k, t_{k-1})$. Let $\vec{\phi}(t_k, t_{k-1})$ represent the j-th column of $\Phi_j(t_k, t_{k-1})$. We assume we have available the nominal states $\vec{x}^*(t_{k-1})$ and $\vec{x}^*(t_k)$. To obtain $\vec{\phi}_j(t_k, t_{k-1})$ we increment the j-th element of $\vec{x}^*(t_{k-1})$ by the numerical differencing factor Δx_j and numerically integrate equation (1) over the time interval $\left[t_{k-1}, t_k\right]$ to obtain the new spacecraft state $\vec{x}_j(t_k)$. The j-subscript indicates $\vec{x}_j(t_k)$ was obtained by incrementing the j-th element of $\vec{x}^*(t_{k-1})$. Then

$$\vec{\phi}_j(t_k, t_{k-1}) = \frac{\vec{x}_j(t_k) - \vec{x}^*(t_k)}{\Delta x_j} \tag{2}$$

$$j = 1,2,\ldots, 6$$

NDTM Flow Chart




B
Compute the N-th column of Φ using equation (2).
Increment column index N.
N > 6
NO
A
YES
Reset saved value.
RETURN

NOMNAL Analysis

NOMNAL is the executive program controlling the entire generation of a nominal trajectory from injection targeting through midcourse corrections and orbit insertion.

NOMNAL begins by calling PRELIM for the preliminary work including initialization of variables, reading of the input data, and computation of zero iterate values of initial time, position, and velocity if required.

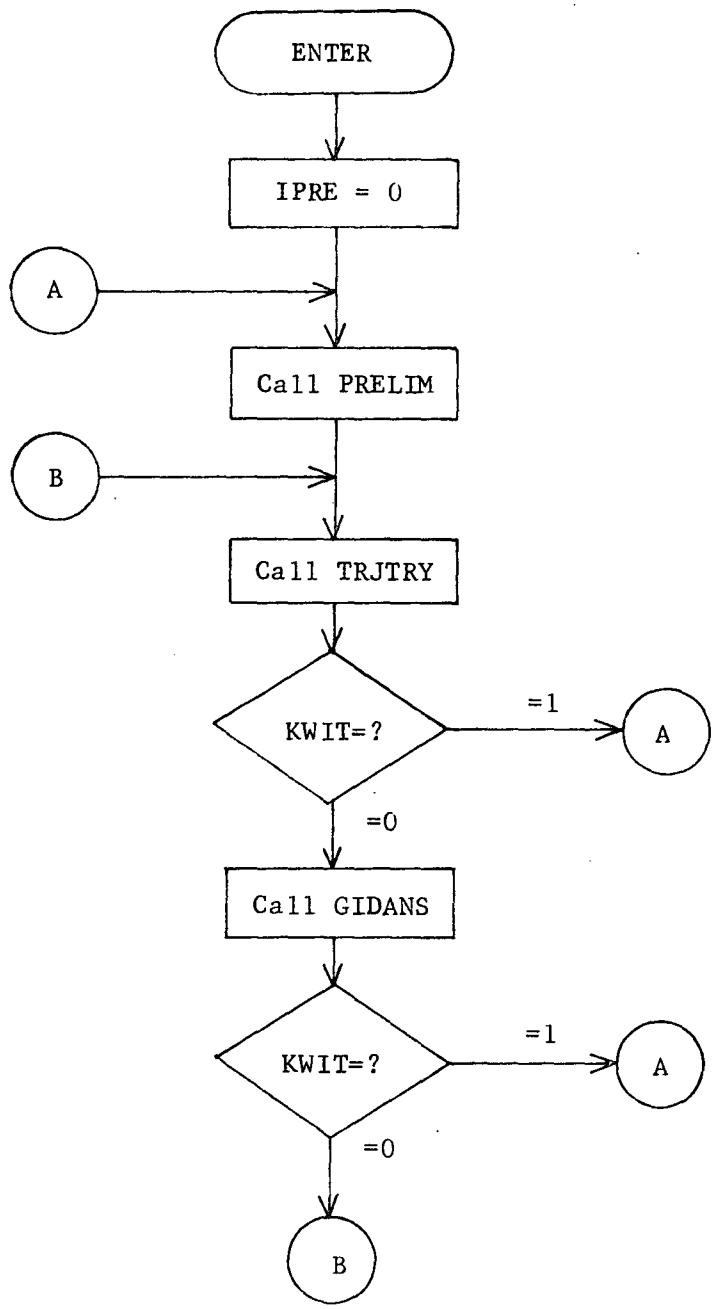
NOMNAL then calls TRJTRY. TRJTRY first determines the time of the next guidance event. It then integrates and records the nominal trajectory to that time. TRJTRY then returns control to NOMNAL.

NOMNAL next calls GIDANS. GIDANS processes the computation and execution of the current guidance event. NOMNAL then reenters its basic cycle by calling TRJTRY to propagate the corrected trajectory to its next guidance event.

Two flags are used by NOMNAL. The flag IPRE is initialized at zero in NOMNAL. During the processing of the first data case PRELIM sets it to unity. PRELIM uses IPRE to determine whether to preset constants to internally stored values or leave them at their previous values before reading the next data case.

The second flag KWIT determines whether the current case should be continued or terminated according to the flag value zero or unity respectively. Termination is indicated when a fatal error occurs during trajectory propagation or guidance event computation or when the desired end time is reached.

NOMNAL Flow Chart



NONINS Analysis

NONINS determines the time and correction vector for an impulsive insertion from an approach hyperbola into a specified plane and as near as possible to a prescribed closed orbit. The approach hyperbola is specified by giving the planetocentric equatorial state $\vec{r}, \vec{v}$ at the time of decision t_d. The final orbit is defined by giving its desired orbital elements $(a_E, e_E, i_E, \omega_E, \Omega_E)$ again in planetocentric equatorial coordinates.

Subroutine CAREL is first called to convert the hyperbolic state at decision $\vec{r}, \vec{v}$ into Keplerian conic elements $(a_H, e_H, i_H, \omega_H, \Omega_H, t_{Hd})$ where t_{Hd} is the time from periapsis at decision (negative on the approach ray).

The points of intersection of the approach orbital plane and the desired orbital plane are then determined. The elements defining the two planes are therefore given by i_H, Ω_H and i_E, Ω_E. Let $\hat{A}$ denote the unit vector toward the ascending node of an orbit and $\hat{B}$ denote the in-plane normal to $\hat{A}$ in the direction of motion. Then

$$\hat{A} = (\cos\Omega, \sin\Omega, 0) \tag{1}$$

$$\hat{B} = (-\sin\Omega \cos i, \cos\Omega \cos i, \sin i) \tag{2}$$

Hence the normal to the orbital plane $\hat{C}$ is given by $\hat{C} = \hat{A} \times \hat{B}$ or

$$\hat{C} = (\sin\Omega \sin i, -\cos\Omega \sin i, \cos i) \tag{3}$$

The direction of the line of intersection of the two planes is therefore determined by $\vec{X} = \hat{C}_H \times \hat{C}_E$ or

$$\begin{aligned}\vec{X} = (&\cos i_H \sin i_E \cos\Omega_E - \sin i_H \cos i_E \cos\Omega_H,\\ &\cos i_H \sin i_E \sin\Omega_E - \sin i_H \cos i_E \sin\Omega_H,\\ &\sin i_H \sin i_E (\cos\Omega_H \sin\Omega_E - \sin\Omega_H \cos\Omega_E))\end{aligned} \tag{4}$$

Then the unit vector along the line of intersection toward the northern hemisphere is given by

$$\hat{X} = \operatorname{sgn} \vec{X}_3 \frac{\vec{X}}{\vec{X}} \tag{5}$$

Therefore the true anomaly f_{HX} along the hyperbola at the northern intersection point is given by

$$\cos(\omega_H + f_{HX}) = \hat{X} \cdot \hat{A}_H \tag{6}$$

The true anomaly on the hyperbola at the southern point is therefore $f_{HX} + 180^{\circ}$. Note that there exists a region of true anomalies lying between the incoming and outgoing asymptotes for which the hyperbola is not defined. Similar equations define the true anomaly on the ellipse at the two points of intersection. Note that this implies that the modified ellipse will have the same ω as the desired ellipse.

For the intersection true anomaly f_{HX} the radius magnitude on the hyperbola may be determined

$$r_I = \frac{a_H(1 - e_H^2)}{1 + e_H \cos f_H} \tag{7}$$

To permit an impulsive insertion, a_E and e_E must be modified to satisfy

$$r_I = \frac{a_E(1 - e_E^2)}{1 + e_E \cos f_E} \tag{8}$$

There are three candidate modifications examined to determine a "best" one: (1) Vary r_a while holding r_p constant

(2) Vary r_p while holding r_a constant

(3) Vary a while holding e constant

"Best" is defined below in terms of a weighted scalar function of the changes in r_a and r_p.

Rewriting (8) in terms of r_a and r_p (using $a = \frac{r_a+r_p}{2}$, $e = \frac{r_a-r_p}{r_a r_p}$) yields the useful relation

$$r_a(1 + \cos f_E) + r_p(1 - \cos f_E) = \frac{2r_a r_p}{r_I} \tag{9}$$

Equation (9) may be solved for r_a as

$$r_a = \frac{r_I r_p(1 - \cos f_E)}{2r_p - r_I(1 + \cos f_E)} \tag{10}$$

This yields the r_a which defines the modified orbit holding r_p at its desired value. The semi-major axis and eccentricity are then computed from $a = \frac{r_a+r_p}{2}$, $e = \frac{r_a-r_p}{r_a+r_p}$.

Similarly (9) may be solved for r_p as

$$r_p = \frac{r_I r_a (1 + \cos f)}{2 r_a - r_I (1 - \cos f)} \tag{11}$$

This determines the modification is r_p required to achieve an intersecting ellipse having the desired r_a.

Finally (8) may be solved trivially for the a_E required to produce intersection for the desired eccentricity.

$$a_E = \frac{r_I (1 + e_E \cos f_E)}{(1 - e_E^2)} \tag{12}$$

An error is assigned to each of the candidate solutions as

$$E_i = W_i \left[|\Delta r_a| + |\Delta r_p| \right]$$

where Δr_a, Δr_p are the errors between the desired and modified values of r_a and r_p. The weighting factor W_i is assigned rather arbitrarily. Currently the weighting factor is $W_i = w_{1i}\, w_{2i}$ where

$w_{1i} = 1$ if the true anomaly is on the incoming ray
$\quad\quad 2$ if the true anomaly is on the outgoing ray

$w_{2i} = 1$ if option 1
$\quad\quad 2$ if option 2
$\quad\quad 3$ if option 3

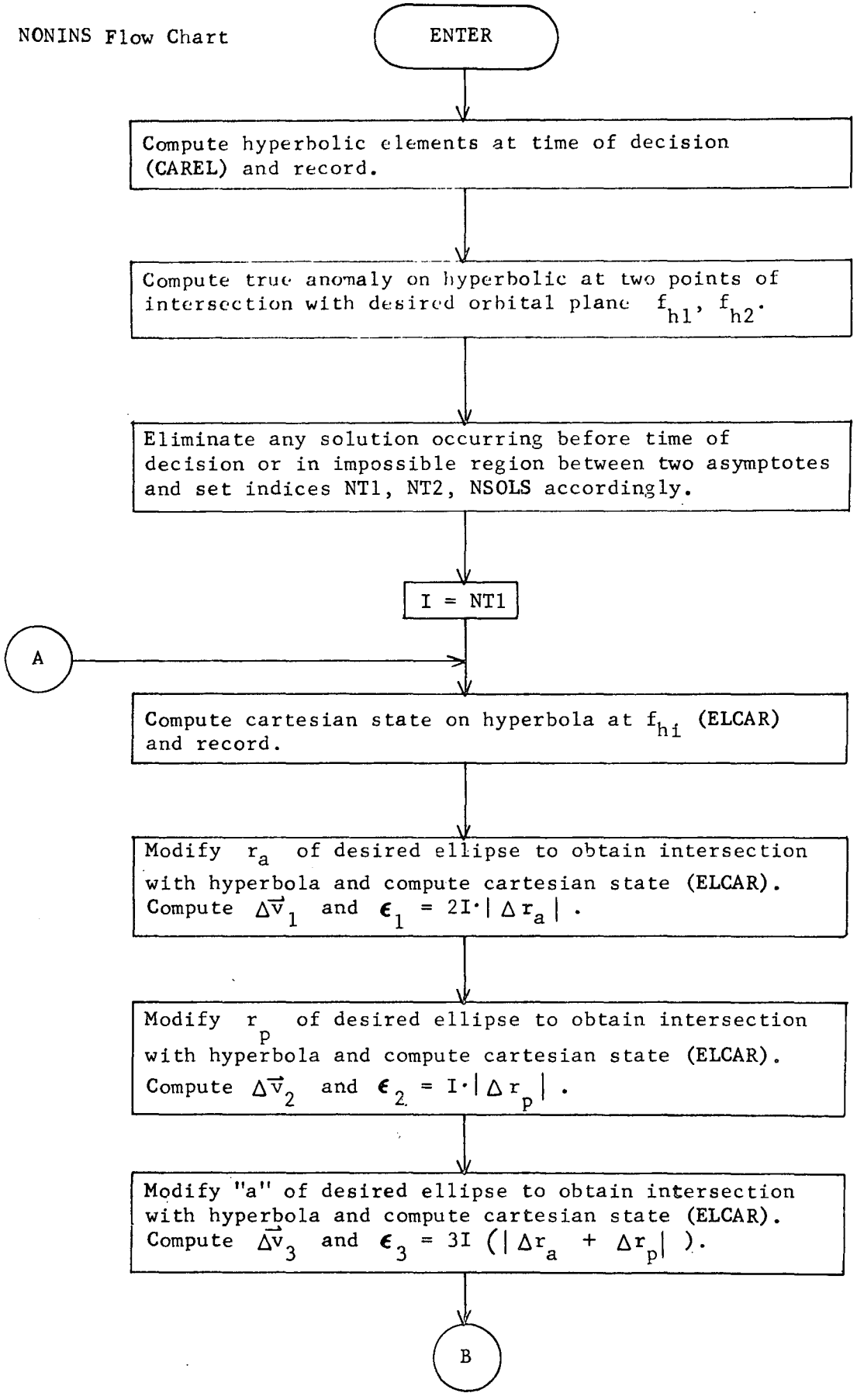
Thus a solution on the incoming asymptote is preferred over one on the outgoing asymptote and one subsequent trim is preferred over two subsequent trims.

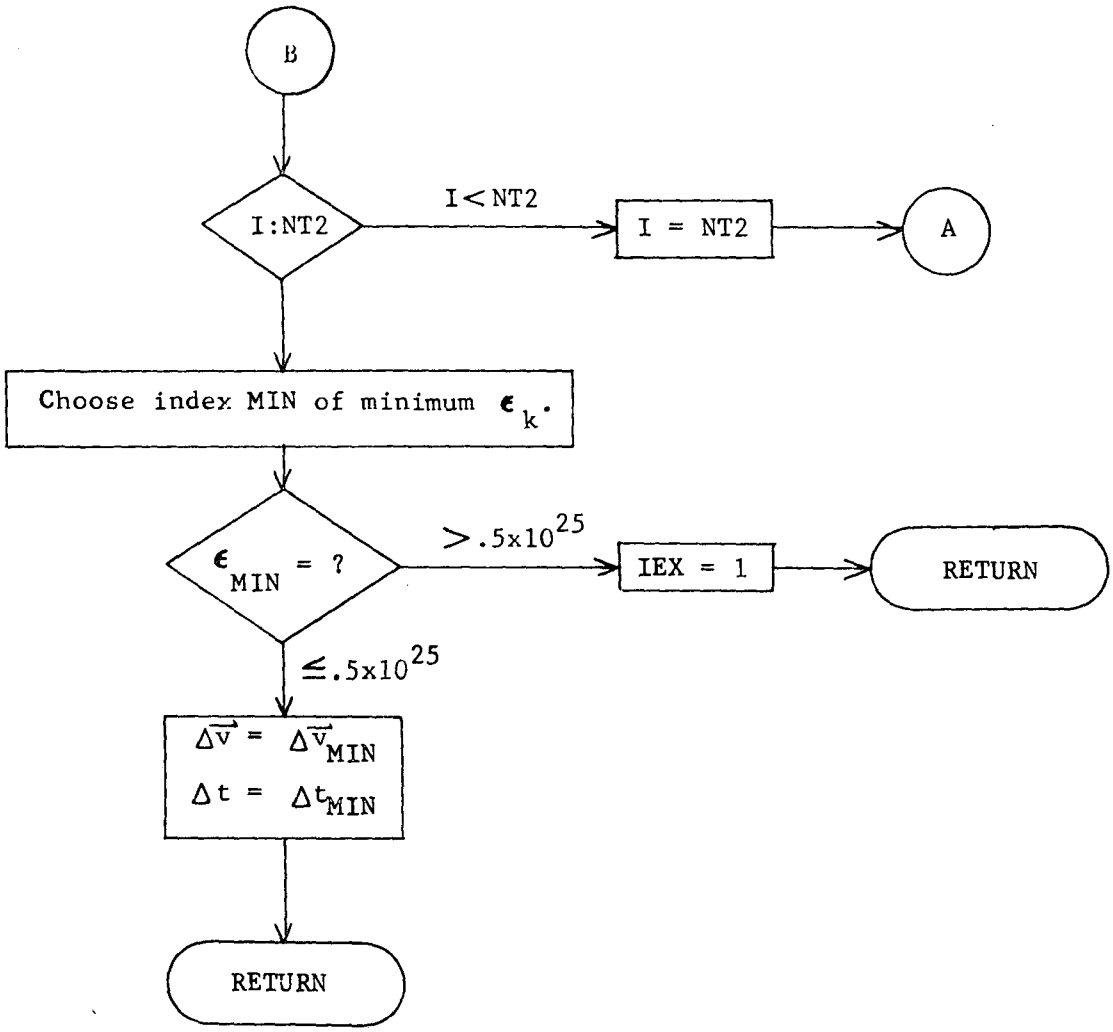
Having determined the elements of an intersecting orbit the insertion parameters are easily computed. The velocity on the hyperbola at the intersection point may be computed from ELCAR as $\vec{v}_H$. The velocity on the ellipse following the insertion is computed by calling ELCAR with the modified elliptical element to get $\vec{v}_E$. The impulsive $\vec{\Delta v}$ is then given by

$$\vec{\Delta v} = \vec{v}_E - \vec{v}_H$$

The time interval from the decision to the execution is given by the hyperbolic time from the initial point to the relevant intersection point.

NONINS Flow Chart




B
I:NT2
I<NT2
I = NT2
A
Choose index MIN of minimum ϵ_k.
ϵ_{MIN} = ?
>.5x10^25
IEX = 1
RETURN
≤.5x10^25
$\Delta\vec{v} = \Delta\vec{v}_{MIN}$
$\Delta t = \Delta t_{MIN}$
RETURN

NØNLIN Analysis

NØNLIN is the interface subroutine between the non-linear guidance subroutines of NØMNAL and subroutine GUIDM of ERRAN and subroutine GUISIM of SIMUL. NØNLIN selects the necessary data from the ERRAN and SIMUL common blocks and stores into the common blocks of NØMNAL the information needed to compute and/or execute the $\Delta\vec{V}$ required in order to meet specified target conditions.

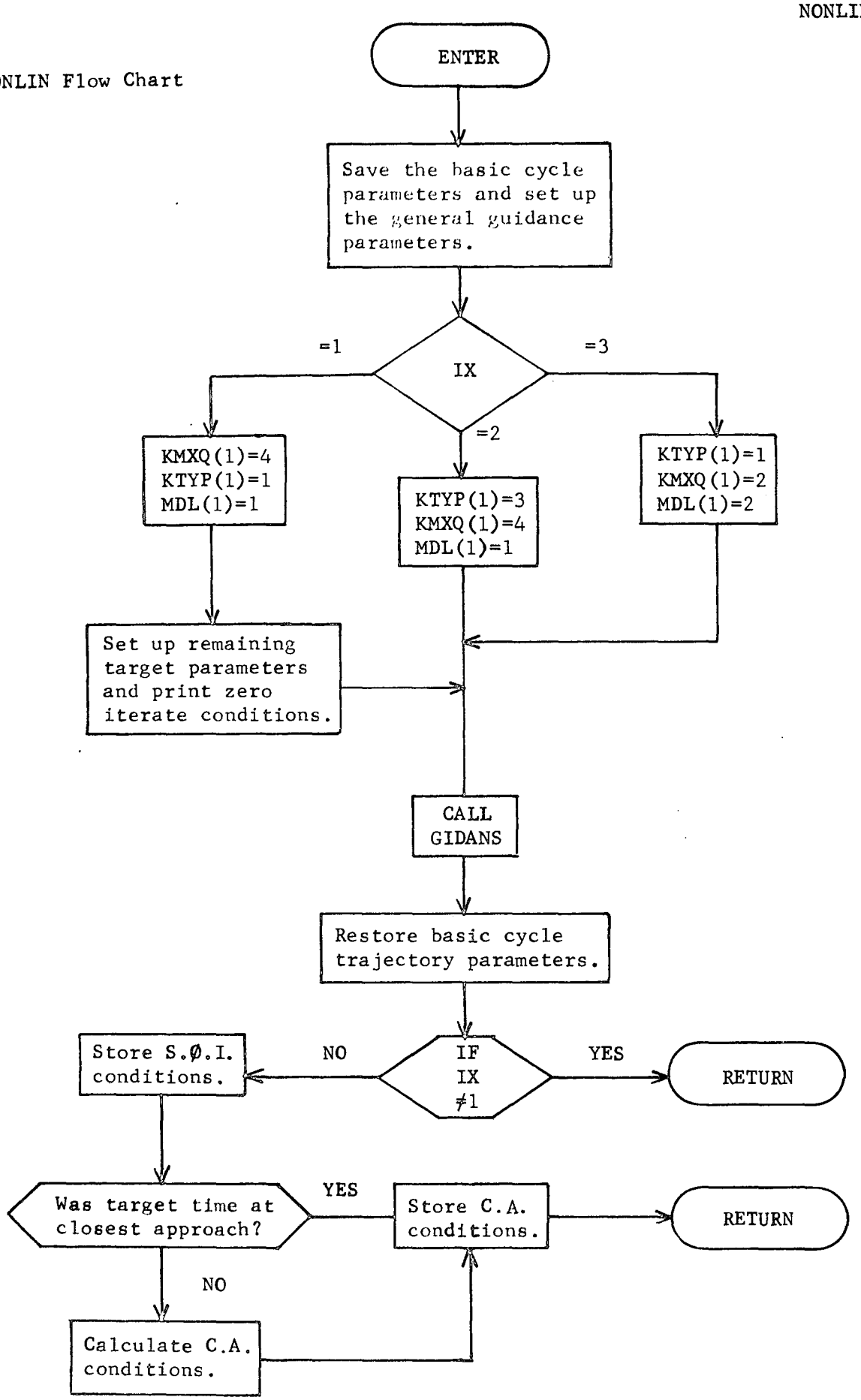
The most important task performed by NØNLIN is the selection of the desired guidance scheme. The variable IX is tested and control is transformed according to the following:

IX = 1, retargeting to specified target parameters
= 2, orbit insertion to specified orbit
= 3, $\Delta\vec{V}$ execution by a series of specified pulses

For each type of event, NØNLIN then sets up values controlling the type of guidance event (KTYP), implementation code (KMXQ), and execution model code (MDL). For retargeting only, NØNLIN stores the remaining values needed for $\Delta\vec{V}$ calculation and prints the zero iterate conditions.

NØMNAL calls GIDANS to perform the guidance event and restores parameters necessary for the basic cycles of ERRAN and SIMUL. For retargeting only, NØMNAL then stores the conditions at sphere of influence and closest approach of the target planet which were calculated by subroutine TARGET.

NØNLIN Flow Chart



NTM Analysis

Subroutine NTM is used to generate the (most recent) targeted nominal trajectory in the error analysis mode. Subroutine NTM is equivalent to a subroutine NTMS from which all loops associated with ICODE = -3, -2, 2, 3 have been removed. For this reason no further analysis and no flow chart will be presented for subroutine NTM. Refer to subroutine NTMS.

NTMS Analysis

Subroutine NTMS is used to generate any of the three trajectories required in the simulation mode -- the (most recent) targeted nominal trajectory, the most recent nominal trajectory, and the actual trajectory.

The input variable ICODE is used to distinguish between these trajectories. It is unimportant to the virtual mass technique which trajectory is being computed. However, it is important to keep them separated so that the proper codes are set that check for approaching the sphere of influence of the target planet and reaching closest approach. It is also important to keep separate the conditions at which these occur for each trajectory. The following list describes ICODE completely.

ICODE = 3, NTMS will check to see if the sphere of influence and/or closest approach has been reached on the actual trajectory. If not, VMP will check for these conditions and on encountering either, NTMS places the conditions in special storage locations so they will be saved for future reference.

ICODE = 2, NTMS performs the same operations as described above for the most recent nominal trajectory.

ICODE = 1, NTMS again checks for sphere of influence and closest approach as above for the targeted nominal trajectory.

ICODE = 0, the only important information in this situation is the state vector at the end of the time interval. Therefore, NTMS does not check to see if closest approach or sphere of influence is encountered. This might occur in numerical differencing, for example.

ICODE = -1, it is important to know if sphere of influence or closest approach is reached on the targeted nominal trajectory. However, it is not desired that the information be stored for future use. This situation occurs in the guidance event.

ICODE = -2, the same comments may be made as if ICODE = -1, except this is on the most recent nominal trajectory.

ICODE = -3, again, this value of ICODE is treated the same as is ICODE = -1, for the actual trajectory.

Physical constants, planetary ephemerides, and other information relating to the dynamic model are the same for the targeted and most recent nominal trajectories. This is not true for the actual trajectory. There may be biases in the target planet ephemerides and the gravitational constants of the Sun and target planet. The numerical accuracy and the number of celestial bodies employed in the generation of the actual trajectory may also differ.

Ephemeris biases are specified as biases in orbital elements a, e, i, Ω, ω, and M . However, within the program are stored the ephemeris constants of a, e, i, Ω, $\tilde{\omega}$, and M for the planets and a, e, i, Ω, $\tilde{\omega}$, and L for the moon, where

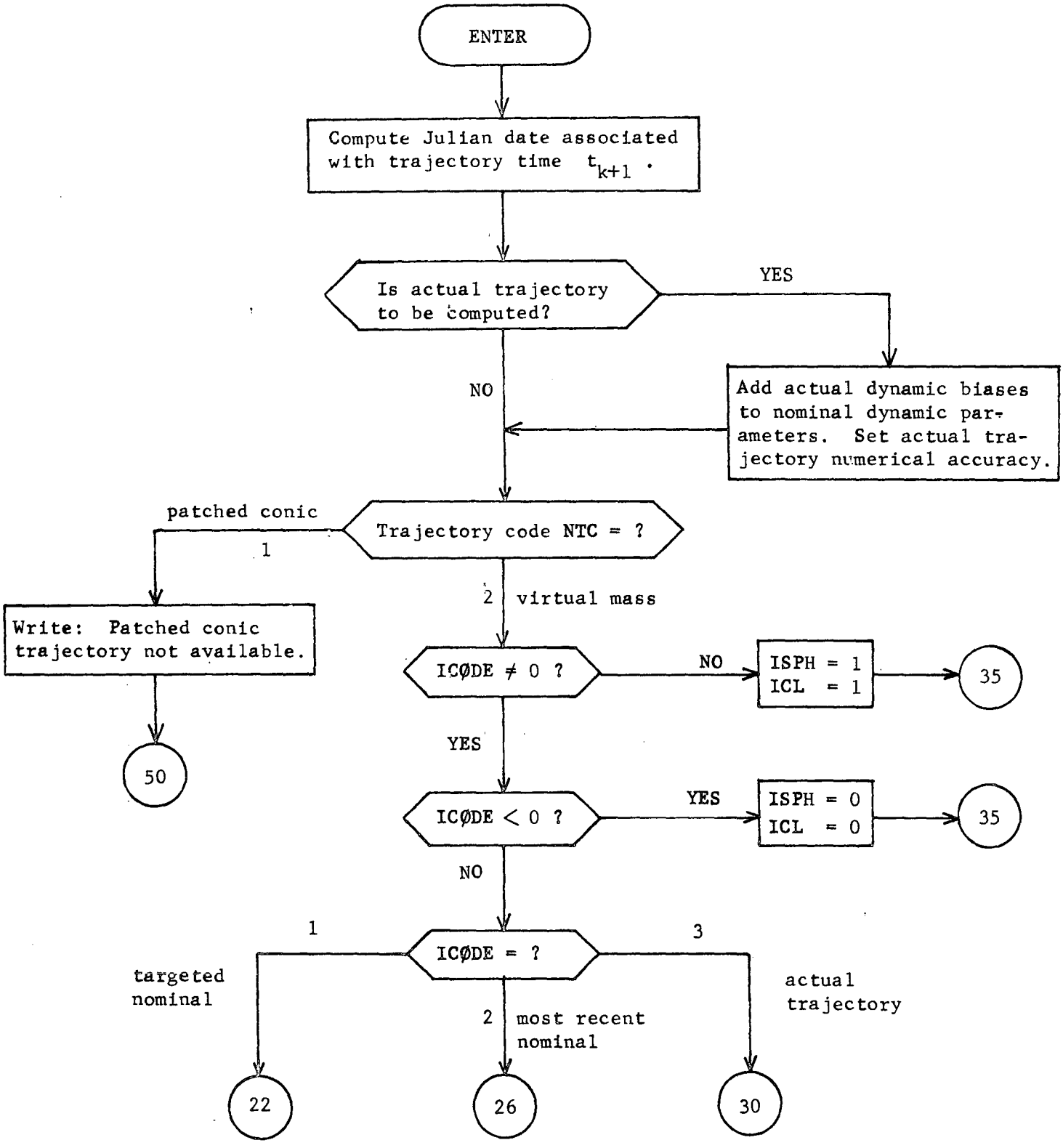
$$\tilde{\omega} = \mathring{\omega} + \Omega$$

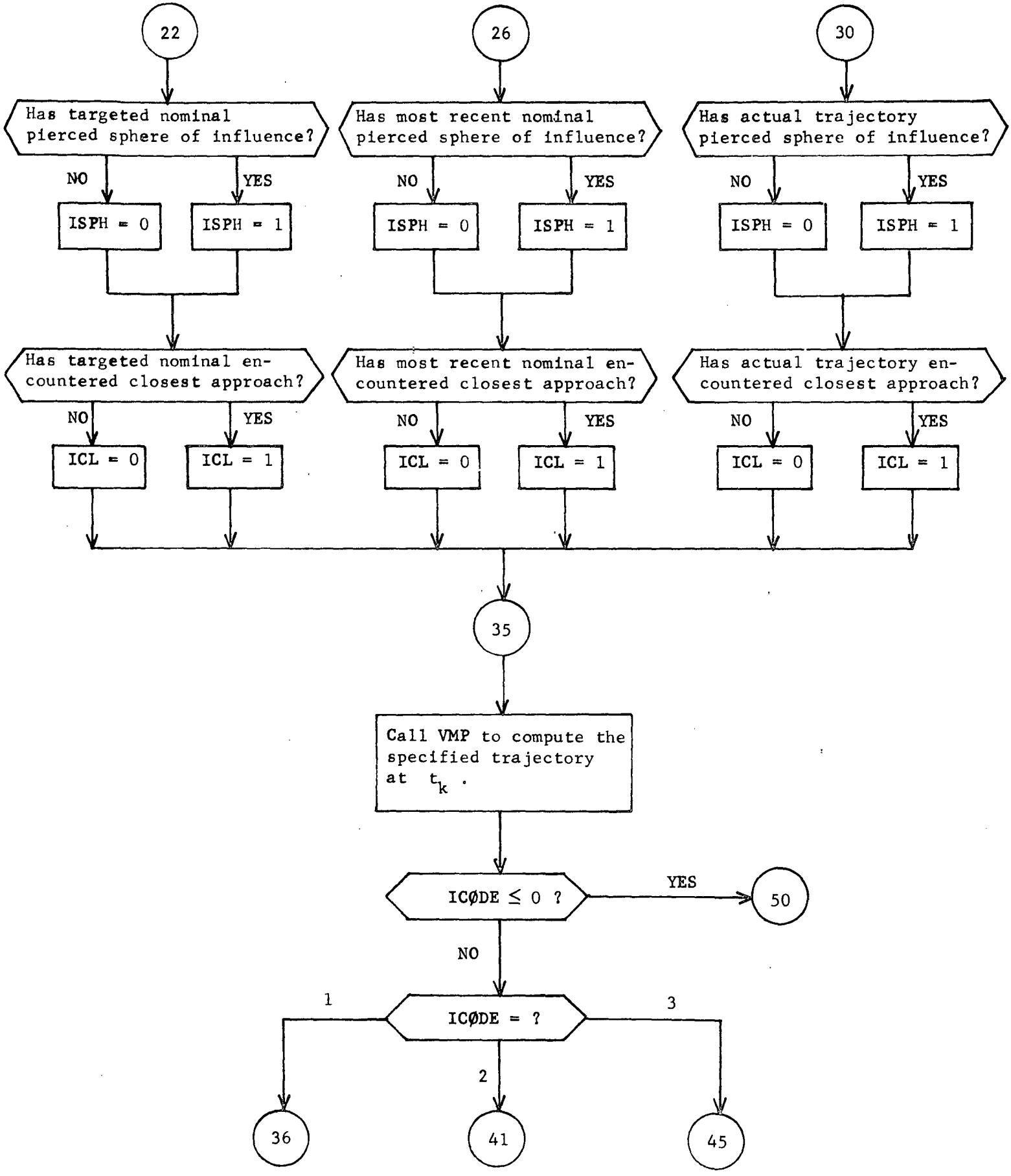
and

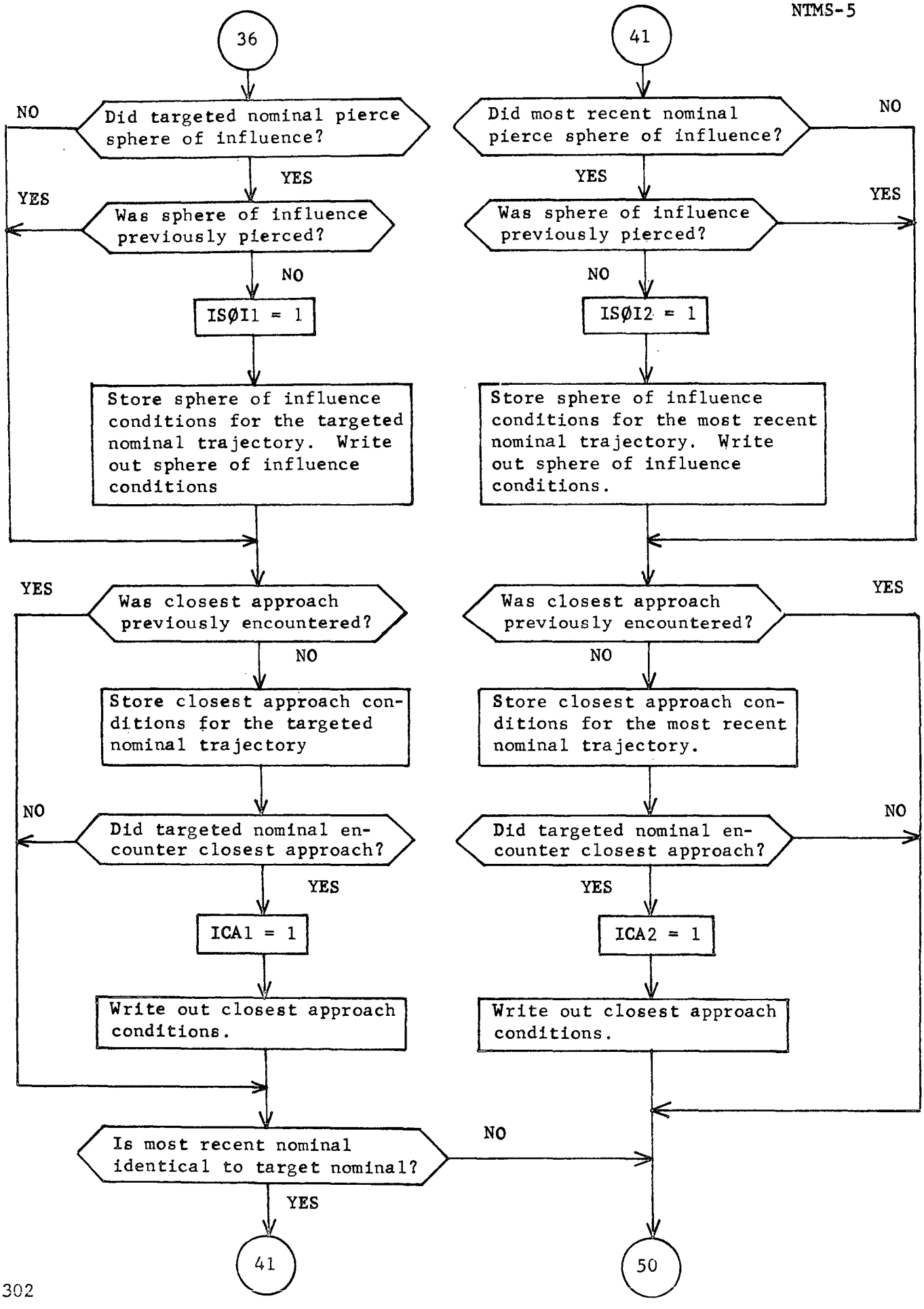
$$L = M + \omega + \Omega \ .$$

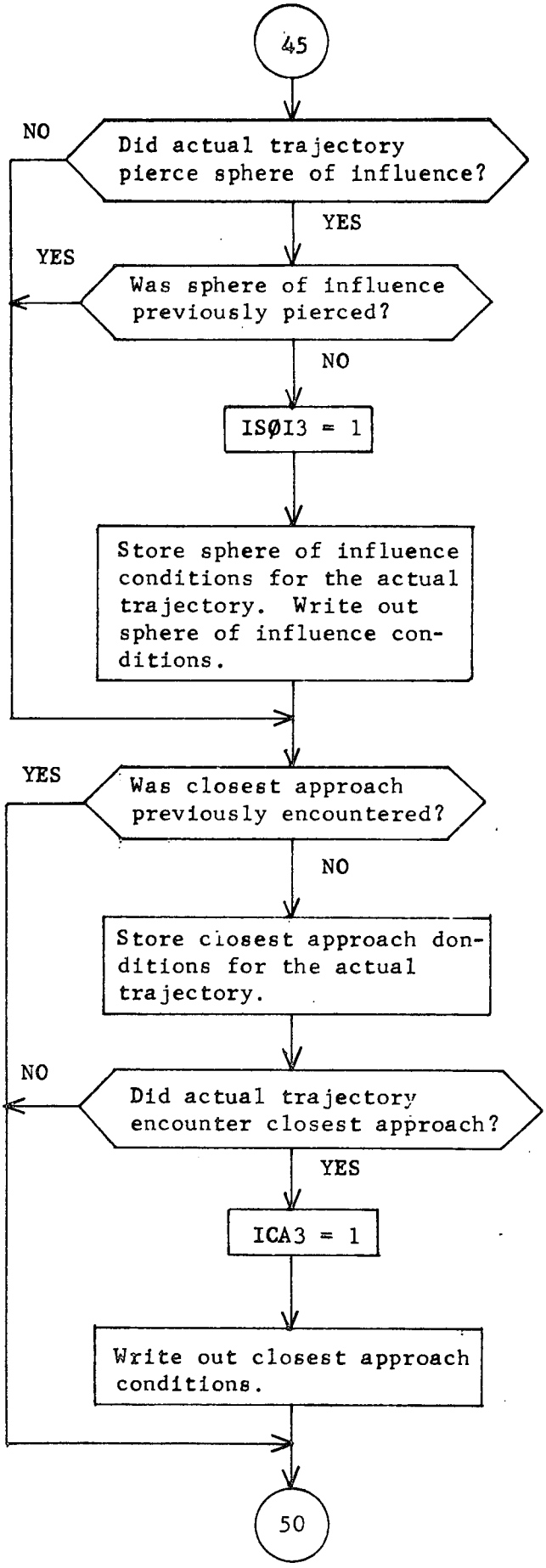
Incrementation of $\tilde{\omega}$ and L requires addition of biases in Ω, ω, and M as indicated by the above equations.

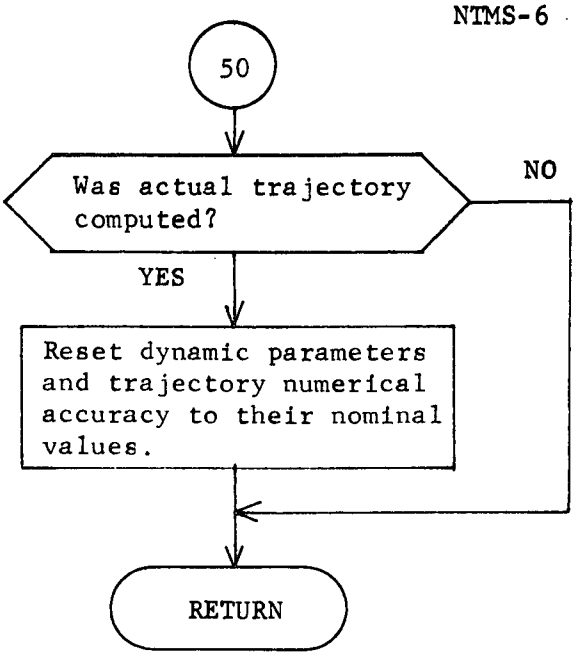
NTMS Flow Chart




22
26
30
Has targeted nominal pierced sphere of influence?
Has most recent nominal pierced sphere of influence?
Has actual trajectory pierced sphere of influence?
NO
YES
ISPH = 0
ISPH = 1
Has targeted nominal encountered closest approach?
Has most recent nominal encountered closest approach?
Has actual trajectory encountered closest approach?
ICL = 0
ICL = 1
35
Call VMP to compute the specified trajectory at t_k .
ICØDE ≤ 0 ?
50
ICØDE = ?
1
2
3
36
41
45


36
Did targeted nominal pierce sphere of influence?
NO
YES
Was sphere of influence previously pierced?
YES
NO
ISØI1 = 1
Store sphere of influence conditions for the targeted nominal trajectory. Write out sphere of influence conditions
41
Did most recent nominal pierce sphere of influence?
NO
YES
Was sphere of influence previously pierced?
YES
NO
ISØI2 = 1
Store sphere of influence conditions for the most recent nominal trajectory. Write out sphere of influence conditions.
Was closest approach previously encountered?
YES
NO
Store closest approach conditions for the targeted nominal trajectory
Was closest approach previously encountered?
YES
NO
Store closest approach conditions for the most recent nominal trajectory.
Did targeted nominal encounter closest approach?
NO
YES
ICA1 = 1
Write out closest approach conditions.
Did targeted nominal encounter closest approach?
NO
YES
ICA2 = 1
Write out closest approach conditions.
Is most recent nominal identical to target nominal?
NO
YES
41
50


45
NO
Did actual trajectory pierce sphere of influence?
YES
YES
Was sphere of influence previously pierced?
NO
ISØI3 = 1
Store sphere of influence conditions for the actual trajectory. Write out sphere of influence conditions.
YES
Was closest approach previously encountered?
NO
Store closest approach conditions for the actual trajectory.
NO
Did actual trajectory encounter closest approach?
YES
ICA3 = 1
Write out closest approach conditions.
50


50
Was actual trajectory computed?
NO
YES
Reset dynamic parameters and trajectory numerical accuracy to their nominal values.
RETURN

ORB Analysis

ORB determines the mean orbital elements for any gravitational body at a specified time.

The elements used are semi-major axis a, eccentricity e, inclination i, longitude of the ascending node Ω, and longitude of periapsis $\tilde{\omega}$. These elements are referenced to heliocentric ecliptic for the planets or geocentric ecliptic for the moon.

The mean elements are computed from time expansions as follows. Let α be any of the elements. Then the value of α at any time t is given by

$$\alpha(t) = \alpha_o + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3$$

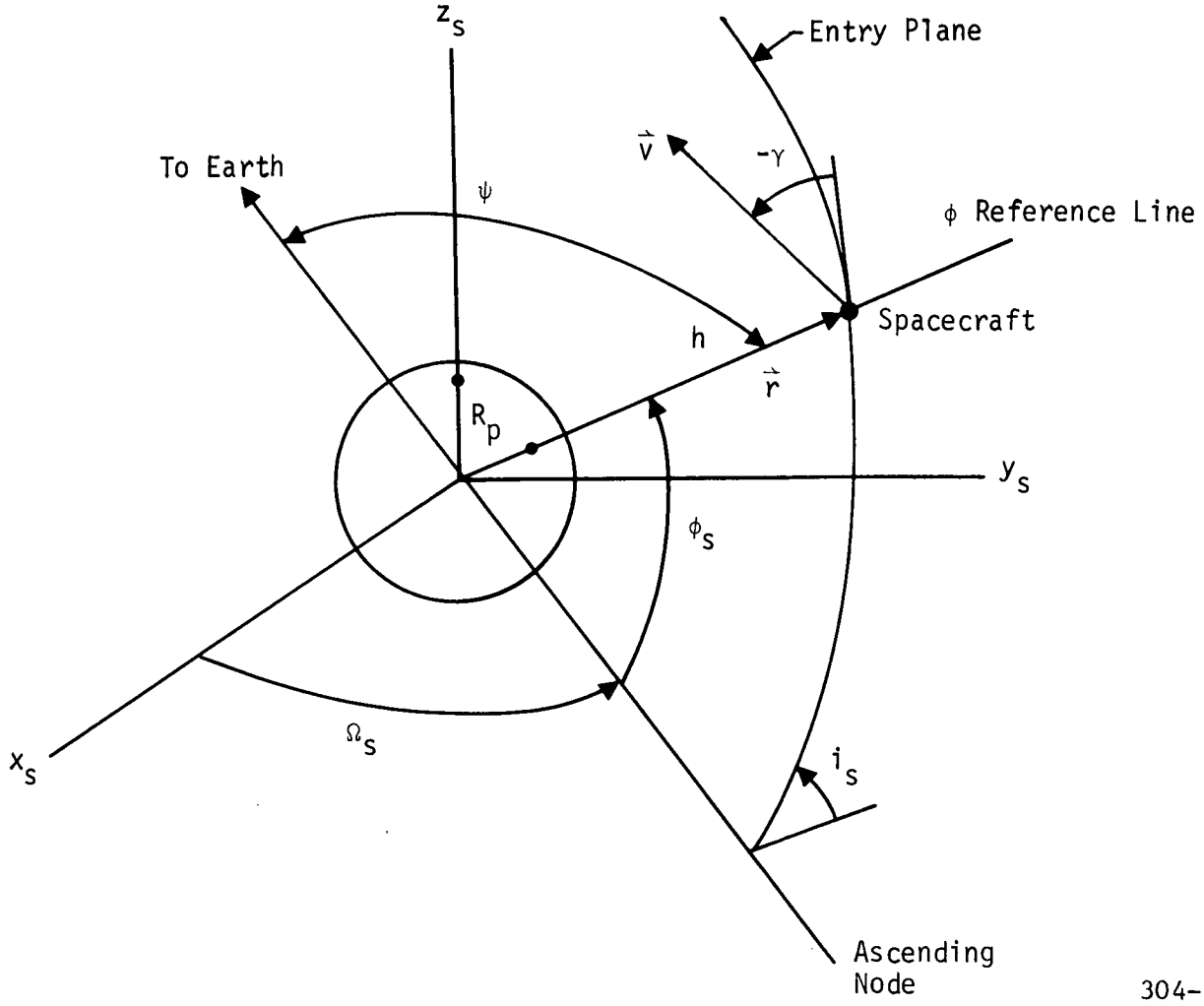
where the constants α_k are stored by BLKDAT. These constants are stored into the arrays CN, ST, and EMN for inner planets, outer planets, and the moon respectively. The definitions of these arrays and the values stored are provided in the analysis of the previous subroutine BLKDAT. The element value as computed from the above equation is then returned in the ELMNT array according to the gravitational body code k as

ELMNT(8k-15) = i	k = 1	Sun	= 1	Saturn
ELMNT(8k-14) = Ω	2	Mercury	8	Uranus
ELMNT(8k-13) = $\tilde{\omega}$	3	Venus	9	Neptune
ELMNT(8k-12) = e	4	Earth	10	Pluto
ELMNT(8k-10) = a	5	Mars	11	Moon
ELMNT(8k-9) = ω	6	Jupiter		

NTRY Analysis

Subroutine NTRY transforms the heliocentric ecliptic spacecraft state and covariance matrix to entry parameter coordinates. This information is useful in defining initial data for the *Lander Trajectory Reconstruction (LTR)* program. Subroutine NTRY also computes the communication angle at entry.

The entry parameter state is defined by altitude h, velocity v relative to the planet, flightpath angle γ, longitude of the ascending node Ω_s, inclination i_s of the entry plane, and the angle ϕ_s between the ascending node and the ϕ reference line. These latter angles are all defined relative to the subsolar orbital-plane coordinate system x_s y_s z_s, which is defined in subroutine SUBSØL. All entry parameters are shown in the following figure, as is the communication angle ψ.



304-1

The transformation of the heliocentric ecliptic spacecraft state to the entry parameter state requires first that the target planet heliocentric ecliptic state be subtracted to obtain the relative spacecraft position $\vec{r}$ and velocity $\vec{v}$. The equations for transforming $\vec{r} = \left(r_x, r_y, r_z\right)$ and $\vec{v} = \left(v_x, v_y, v_z\right)$ to $\left(h, v, \gamma, \phi_s, i_s, \Omega_s\right)$ are summarized below.

Define $\vec{e}_r = \frac{\vec{r}}{r}$ and $\vec{e}_n = \frac{\vec{r} \times \vec{v}}{|\vec{r} \times \vec{v}|}$, where $\vec{e}_r$ is a unit vector aligned with $\vec{r}$ and $\vec{e}_n$ is a unit vector normal to the entry plane. Let $\vec{e}_z$ denote a unit vector aligned with the z_s-axis. The Cartesian subsolar orbital-plane components of these three unit vectors will be denoted as follows:

$$\vec{e}_r = \left(e_{r_x}, e_{r_y}, e_{r_z}\right)$$

$$\vec{e}_n = \left(e_{n_x}, e_{n_y}, e_{n_z}\right)$$

$$\vec{e}_z = (0, 0, 1).$$

Altitude h and velocity v are readily obtained:

$$h = |\vec{r}| - R_p \tag{1}$$

$$v = |\vec{v}| \tag{2}$$

where R_p is the target planet radius. Flightpath angle γ is computed from

$$\gamma = \sin^{-1}\left(\frac{\dot{r}}{v}\right) \tag{3}$$

where

$$\dot{r} = \vec{v} \cdot \vec{e}_r.$$

Longitude of the ascending node Ω_s is given by

$$\Omega_s = \tan^{-1}\left(\frac{e_{n_x}}{-e_{n_y}}\right), \tag{4}$$

while inclination i_s is obtained from

$$i_s = \cos^{-1}\left(e_{n_z}\right). \tag{5}$$

The angle ϕ_s is given by

$$\phi_s = \tan^{-1}\left(\frac{\sin \phi_s}{\cos \phi_s}\right) \tag{6}$$

where

$$\sin \phi_s = \frac{e_{r_z}}{\sin i_s}$$

and

$$\cos \phi_s = -\frac{\left(e_{n_y} e_{r_x} + e_{n_x} e_{r_y}\right)}{\left[e_{n_y}^2 + e_{n_x}^2\right]^{\frac{1}{2}}}.$$

If i_s = 0 or 180 degrees, the following equation for ϕ_s is used instead:

$$\phi_s = \tan^{-1}\left(\frac{e_{r_y}}{e_{r_z}}\right) - \Omega_s. \tag{7}$$

The desired entry parameter covariance matrix is defined by

$$P = E\left[\vec{x}\,\vec{x}^T\right] \tag{8}$$

where $\vec{x} = \left(\delta h,\ \delta v,\ \delta\gamma\ \ \delta\phi_s\right)$. Given the covariance matrix

$$P' = E\left|\vec{x}'\ \vec{x}'^{T}\right| \tag{9}$$

where $\vec{x}' = \left(\delta r_x,\ \delta r_y,\ \delta r_z,\ \delta v_x,\ \delta v_y,\ \delta v_z\right)$, the desired covariance matrix can be obtained from

$$P = A\ P'\ A^T \tag{10}$$

where transformation matrix A is defined by

$$\vec{x} = A\ \vec{x}'. \tag{11}$$

The elements a_{ij} of the 4x6 matrix A are found by computing the differentials of equations (1), (2), (3), and (6). The results of this process are summarized as:

$$a_{11} = \frac{r_x}{r},\ a_{12} = \frac{r_y}{r},\ a_{13} = \frac{r_z}{r},\quad a_{14} = a_{15} = a_{16} = 0$$

$$a_{21} = a_{22} = a_{23} = 0,\ a_{24} = \frac{v_x}{v},\ a_{25} = \frac{v_y}{v},\ a_{26} = \frac{v_z}{v}$$

$$a_{31} = \frac{1}{h'}\left[v_x - \frac{r_x}{r^2}\left(\vec{r}\cdot\vec{v}\right)\right],\ a_{32} = \frac{1}{h'}\left[v_y - \frac{r_y}{r^2}\left(\vec{r}\cdot\vec{v}\right)\right]$$

$$a_{33} = \frac{1}{h'}\left[v_z - \frac{r_z}{r^2}\left(\vec{r}\cdot\vec{v}\right)\right],\ a_{34} = \frac{1}{h'}\left[r_x - \frac{v_x}{v^2}\left(\vec{r}\cdot\vec{v}\right)\right]$$

$$a_{35} = \frac{1}{h'}\left[r_y - \frac{v_y}{v^2}\left(\vec{r}\cdot\vec{v}\right)\right],\ a_{36} = \frac{1}{h'}\left[r_z - \frac{v_z}{v^2}\ \vec{r}\cdot\vec{v}\right)\right]$$

$$a_{41} = -\frac{r_x \tan\phi_s}{r_z\ r^2},\ a_{42} = -\frac{r_y \tan\phi_s}{r_z\ r^2},\ a_{43} = \frac{\tan\phi_s}{r_z}\left(1 - \frac{r_z}{r^2}\right)$$

$$a_{44} = a_{45} = a_{46} = 0$$

where $h' = \left|\vec{r} \times \vec{v}\right|$ and is the orbit angular momentum/unit mass.

Communication angle ψ is computed from

$$\cos \psi = \frac{\vec{r} \cdot \left(\vec{r}_e - \vec{r}_p\right)}{|\vec{r}| \cdot |\vec{r}_e - \vec{r}_p|} \qquad (12)$$

where

$\vec{r}$ = spacecraft position relative to planet

$\vec{r}_e$ = Earth position relative to sun

$\vec{r}_p$ = planet position relative to sun.

PARTL Analysis

PARTL is responsible for the computation of the partials of $B \cdot T$ and $B \cdot R$ with respect to the cartesian components of position and velocity.

Let the state of the spacecraft with respect to the target body at intersection with its sphere of influence be denoted

$$\vec{r} = \left[x, y, z \right]^T \qquad r = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

$$\vec{v} = \left[\dot{x}, \dot{y}, \dot{z} \right]^T \qquad v = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \tag{2}$$

Introduce the approach asymptote $\hat{S}$ and approximate it by the direction of $\vec{v}$

$$\hat{S} = \frac{\vec{v}}{v} \tag{3}$$

The B-plane is the plane normal to $\hat{X}$ containing the center of the target body. Any vector $\vec{\beta}$ within the B-plane must satisfy therefore

$$\hat{S} \cdot \vec{\beta} = 0 \tag{4}$$

The impact parameter vector $\vec{B}$ is determined by the intersection of the B-plane and the incoming asymptote. The incoming asymptote is given parametrically by

$$\vec{\sigma} = \vec{r} + \vec{v}\, t \tag{5}$$

The time at which the asymptote intersects the B-plane may be determined by applying the B-plane condition (4)

$$\hat{S} \cdot \vec{r} + \hat{S} \cdot \vec{v}\, t = 0$$

$$t = - \frac{\vec{r} \cdot \vec{v}}{v^2} \tag{6}$$

Therefore the B-vector is given by

$$\vec{B} = \vec{r} - \frac{\vec{r} \cdot \vec{v}}{v^2} \vec{v}$$

$$\vec{B} = \left[x - \alpha \dot{x}, y - \alpha \dot{y}, z - \alpha \dot{z} \right]^T \tag{7}$$

where $$\alpha = \frac{\vec{r} \cdot \vec{v}}{v^2} = \frac{x\dot{x} + y\dot{y} + z\dot{z}}{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$$

Now assuming that the T axis is to lie in the x-y reference plane and the B-plane, it is defined as

$$\hat{T} = \frac{\hat{S} \times \hat{K}}{|\hat{S} \times \hat{K}|}$$

$$\hat{T} = \left[\frac{\dot{y}}{u} , - \frac{\dot{x}}{u} , 0 \right] \tag{8}$$

where $u^2 = \dot{x}^2 + \dot{y}^2$. The $\hat{R}$ axis is defined by

$$\hat{R} = \hat{S} \times \hat{T}$$

$$\hat{R} = \frac{1}{uv} \left[\dot{x}\dot{z}, \dot{y}\dot{z}, -u^2 \right]^T \tag{9}$$

Now combining (7), (8), (9) B•T and B•R may be computed in terms of the state components

$$B \cdot T = \frac{1}{u} (x\dot{y} - \dot{x}y)$$

$$B \cdot R = \frac{1}{uv} \left[(x\dot{x} + y\dot{y})\dot{z} - u^2 z \right] \tag{10}$$

where $u^2 = \dot{x}^2 + \dot{y}^2$, $v^2 = u^2 + \dot{z}^2$.

The partials may now be computed from differentiation of the above equations.

$$\frac{\partial B \cdot T}{\partial x} = \frac{\dot{y}}{u} \qquad \frac{\partial B \cdot R}{\partial x} = \frac{\dot{x}\dot{z}}{uv}$$

$$\frac{\partial B \cdot T}{\partial y} = - \frac{\dot{x}}{u} \qquad \frac{\partial B \cdot R}{\partial y} = \frac{\dot{y}\dot{z}}{uv}$$

$$\frac{\partial B \cdot T}{\partial z} = 0 \qquad \frac{\partial B \cdot R}{\partial z} = - \frac{u}{v}$$

$$\frac{\partial B \cdot T}{\partial \dot{x}} = - \frac{\dot{y}}{u^3} (x\dot{x} + y\dot{y}) \qquad \frac{\partial B \cdot R}{\partial \dot{x}} \quad \frac{\dot{z}}{u^3 v^3} \left[u^2 (v^2 x - \dot{x}z\dot{z}) - \dot{x}(u^2 + v^2)(x\dot{x} + y\dot{y}) \right]$$

$$\frac{\partial B \cdot T}{\partial \dot{y}} = \frac{\dot{x}}{u^3} (x\dot{x} + y\dot{y}) \qquad \frac{\partial B \cdot R}{\partial \dot{y}} \quad \frac{\dot{z}}{u^3 v^3} \left[u^2 (v^2 y - \dot{y}z\dot{z}) - \dot{y}(u^2 + v^2)(x\dot{x} + y\dot{y}) \right]$$

$$\frac{\partial B \cdot T}{\partial \dot{z}} = 0 \qquad \frac{\partial B \cdot R}{\partial \dot{z}} = \frac{u}{v^3} (x\dot{x} + y\dot{y} + z\dot{z})$$

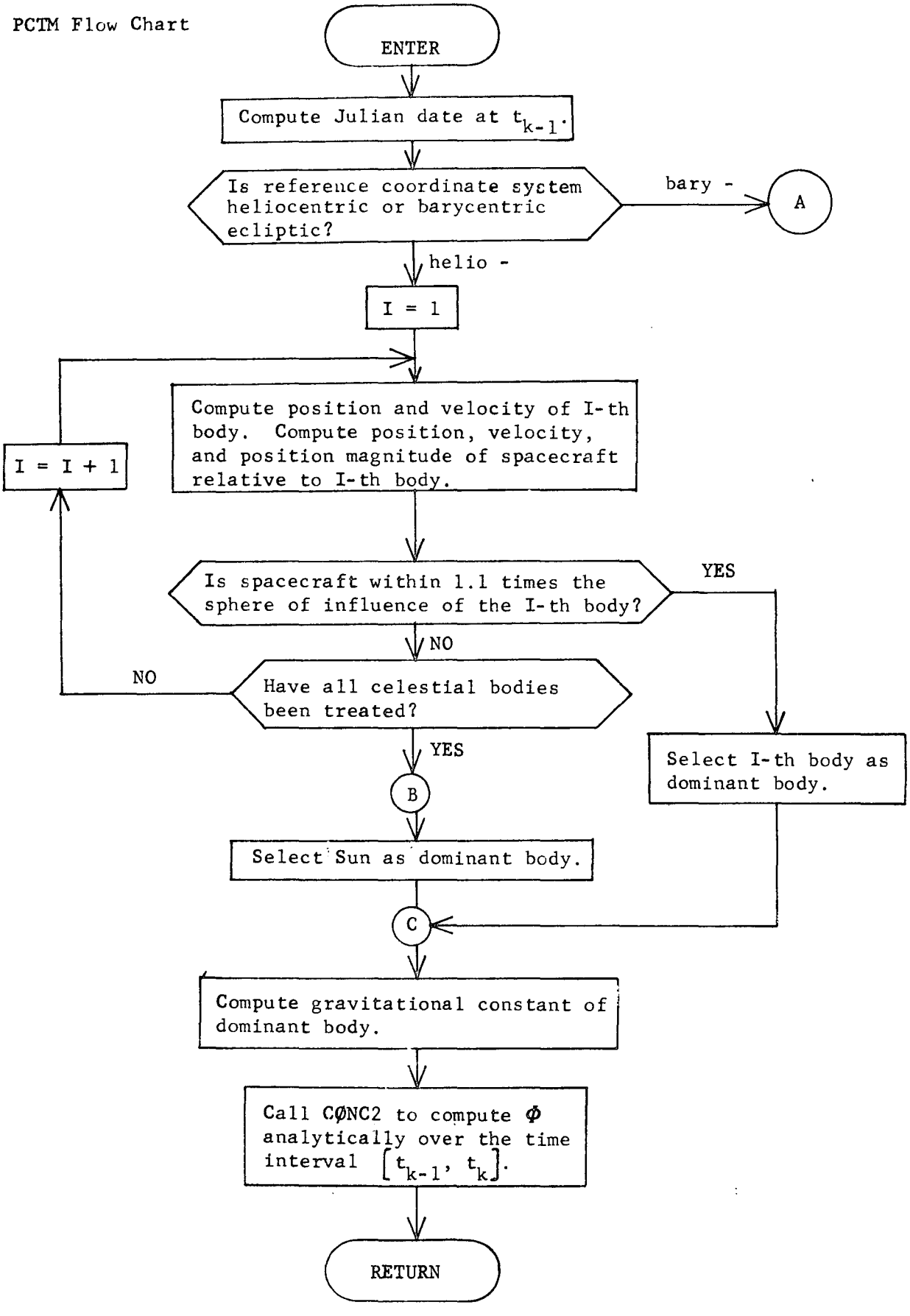
PCTM Analysis

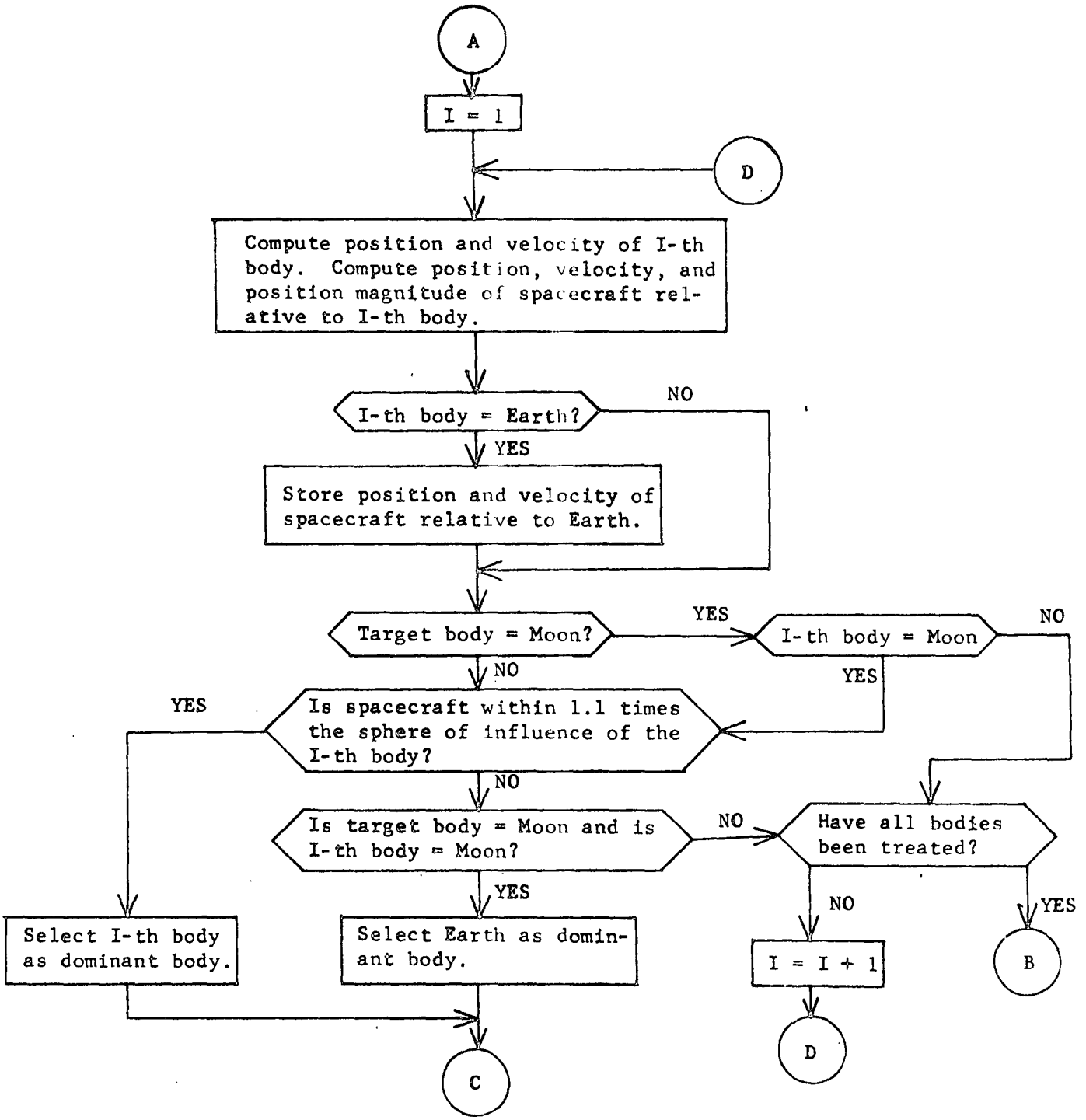
Subroutine PCTM does not actually compute the state transition matrix $\phi(t_k, t_{k-1})$ itself; this is accomplished by calling CØNC2 from within PCTM. The primary function of PCTM is to determine the dominant body at time t_{k-1} to be used in the computation of $\phi(t_k, t_{k-1})$ by means of the analytical patched conic technique.

On interplanetary trajectories we compute the distance separating the spacecraft from each of the celestial bodies included in the analysis. If the distance between the spacecraft and the i-th body is less than or equal to 1.1 times the sphere of influence of the i-th body, the i-th body is selected as the dominant body. Otherwise, the Sun is selected as the dominant body.

On lunar trajectories we compute the distance separating the spacecraft from the Moon. If this distance is less than or equal to 1.1 times the sphere of influence of the Moon, the moon is selected as the dominant body. If not, the Earth is selected as the dominant body.

PCTM Flow Chart




A
I = 1
D
Compute position and velocity of I-th body. Compute position, velocity, and position magnitude of spacecraft relative to I-th body.
I-th body = Earth?
NO
YES
Store position and velocity of spacecraft relative to Earth.
Target body = Moon?
YES
I-th body = Moon
NO
YES
NO
YES
Is spacecraft within 1.1 times the sphere of influence of the I-th body?
NO
Is target body = Moon and is I-th body = Moon?
NO
Have all bodies been treated?
YES
NO
YES
Select I-th body as dominant body.
Select Earth as dominant body.
I = I + 1
B
D
C

PECEQ Analysis

Subroutine PECEQ computes the coordinate transformation maxtrix A from planetocentric ecliptic to planetocentric equatorial coordinates for an arbitrary planet.

The derivation of A for a planet other than the earth or moon will be summarized. Matrix A is defined by

$$A = \left[\hat{X} \mid \hat{Y} \mid \hat{Z}\right]^T \tag{1}$$

where $\hat{X}$, $\hat{Y}$, and $\hat{Z}$ are unit vectors aligned with the planetocentric equatorial coordinate axes and referenced to the planetocentric ecliptic coordinate system. Unit vector $\hat{Z}$ is aligned with the planet pole. Unit vector $\hat{X}$ lies along the intersection of the of the planet equatorial and orbital planes and points at the planet vernal equinox. Unit vector $\hat{Y}$ completes the orthogonal triad and is given by

$$\hat{Y} = \hat{Z} \times \hat{X}. \tag{2}$$

It remains to obtain expressions for $\hat{X}$ and $\hat{Z}$. Let $\hat{N}$ denote the unit vector normal to the planet orbital plane, and let $\hat{P}$ denote the unit vector aligned with the planet pole. Then

$$\hat{Z} = \hat{P} \tag{3}$$

and

$$X = \frac{\hat{P} \times \hat{N}}{\left|\hat{P} \times \hat{N}\right|}. \tag{4}$$

The unit vector $\hat{N}$, referred to the ecliptic coordinate system, is given by

$$\hat{N} = \begin{bmatrix} \sin i \sin \Omega \\ -\sin i \cos \Omega \\ \cos i \end{bmatrix} \tag{5}$$

where i and Ω are the inclination and longitude of the ascending node, respectively, of the planet orbital plane. The unit vector $\hat{P}$, referred to the ecliptic system is given by

$$\hat{P} = \begin{bmatrix} \cos\alpha \cos\delta \\ \cos\varepsilon \sin\alpha \cos\delta + \sin\varepsilon \sin\delta \\ -\sin\varepsilon \sin\alpha \cos\delta + \cos\varepsilon \sin\delta \end{bmatrix} \quad (6)$$

where α and δ are the right ascension and declination, respectively, of the planet pole relative to the geocentric equatorial coordinate system, and ε is the obliquity of the ecliptic. Expressions for α and δ for each planet were obtained from JPL TR 32-1306, *Constants and Related Information for Astrodynamic Calculations*, 1968, by Melbourne, *et al.*

For the earth and the moon, the transformation matrix A is written as the produce of two transformation matrices

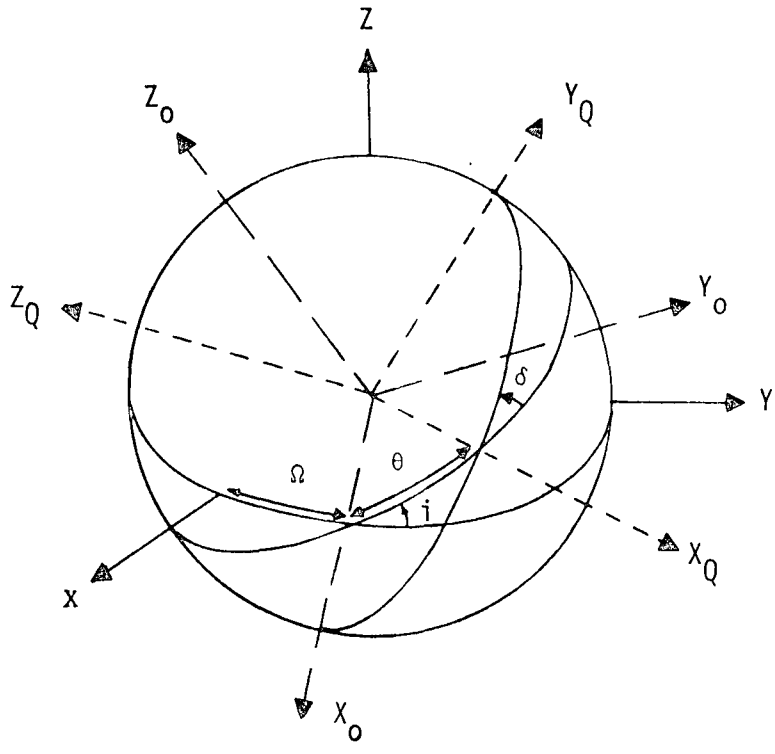
$$A = A_2 A_1. \quad (7)$$

For the earth A_2 is the identity matrix and A_1 is given by

$$A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varepsilon & -\sin\varepsilon \\ 0 & \sin\varepsilon & \cos\varepsilon \end{bmatrix}. \quad (8)$$

The following figure defines the transformations A_1 and A_2, using the definitions given.

XYZ	Ecliptic coordinate axes
$X_o Y_o Z_o$	Orbital plane coordinate axes
$X_Q Y_Q Z_Q$	Moon's equatorial coordinate axes
i	Inclination of moon's orbital plane to ecliptic plane
Ω	Right ascension of moon's orbital plane to ecliptic plane
δ	Inclination of moon's equatorial to orbital plane
θ	Right ascension of moon's equatorial to orbital plane



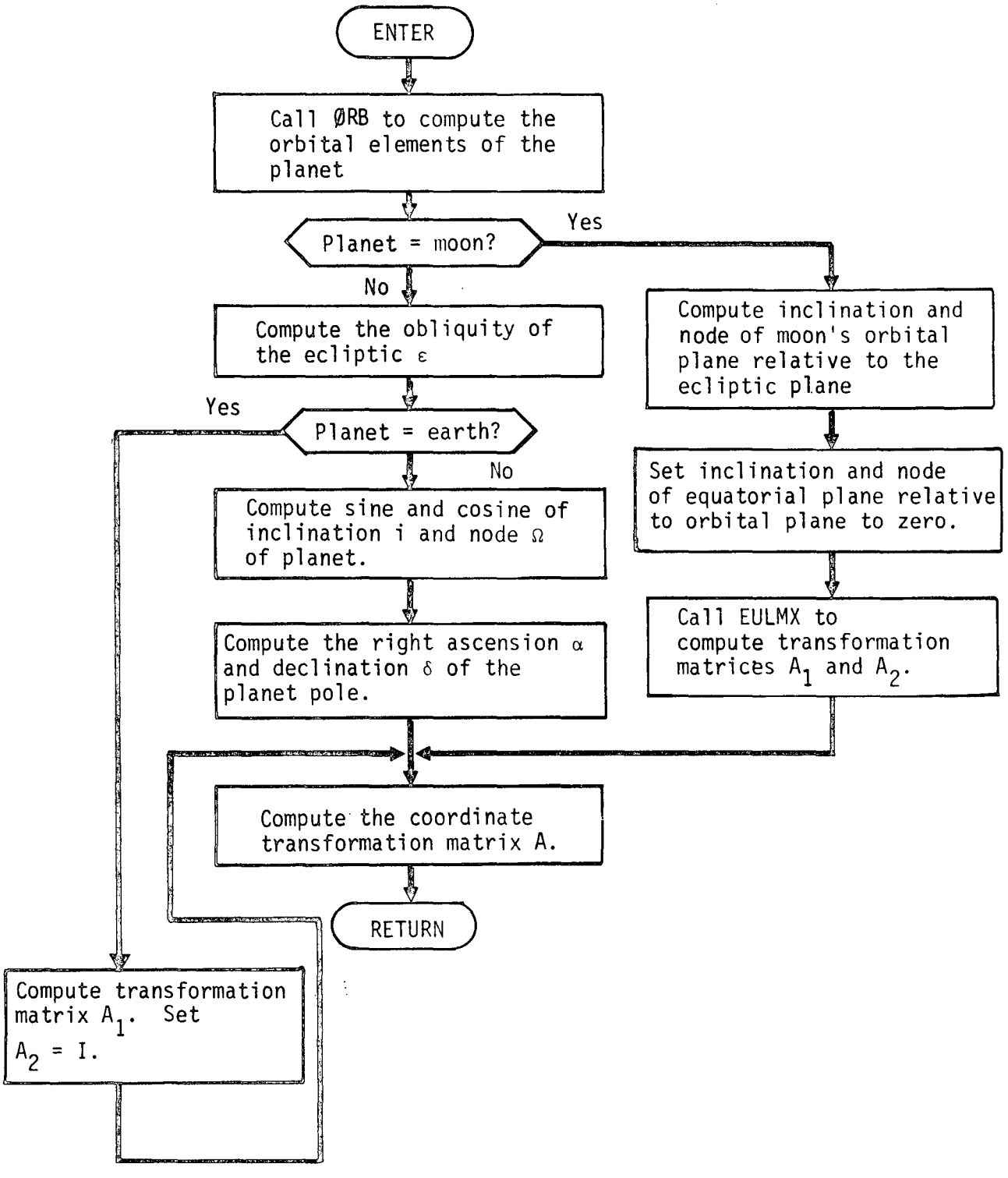
The transformation A_1 from ecliptic to orbital plane coordinates is performed by rotating about the z-axis through an angle Ω and then about the resulting x-axis through an angle i. Symbolically,

$$A_1 = (\Omega \text{ about } 3,\ i \text{ about } 1). \tag{9}$$

The transformation A_2 from orbital plane to equatorial coordinates can be written similarly as

$$A_2 = (\theta \text{ about } -3,\ \delta \text{ about } -1). \tag{10}$$

PECEQ Flow Chart



PERHEL Analysis

PERHEL is responsible for propagating a heliocentric trajectory considering the perturbations produced by both the launch and target bodies. The equations of motion of a body moving under the influence of the sun while perturbed by a smaller mass are

$$\ddot{\vec{r}} = -\frac{\mu_o \vec{r}}{r^3} - \frac{\mu(\vec{r} - \vec{r}_m)}{|\vec{r} - \vec{r}_m|^3} - \frac{\mu \vec{r}_m}{r_m^3} \tag{1}$$

where $\vec{r}$ is the vector radius from the sun to the spacecraft
$\vec{r}_m$ is the vector radius from the sun to the perturbative mass
μ_o, μ are the gravitational constants of the sun and mass respectively.

Assuming that the indirect term is small, attention may be directed to the first two terms only. Suppose that $(\vec{r}_o(t), \vec{v}_o(t))$ satisfy

$$\begin{aligned} \dot{\vec{r}}_o &= \vec{v}_o \\ \dot{\vec{v}}_o &= \frac{\mu_o \vec{r}_o}{r_o^3} \end{aligned} \tag{2}$$

Then $(r_o(t), v_o(t))$ are given by the familar equations of conic motion.

A first order corrected solution necessary to account for the direct term force must then satisfy

$$\dot{\vec{r}} = \dot{\vec{r}}_o + \dot{\vec{\delta r}} = \vec{v}$$

$$\dot{\vec{v}} = \dot{\vec{v}} + \dot{\vec{\delta v}} = -\frac{\mu_o \vec{r}_o}{r_o^3} - \mu \frac{(\vec{r}_o - \vec{r}_m)}{|\vec{r}_o - \vec{r}_m|^3} \tag{3}$$

Applying the conditions (2) leads to the equations defining the corrections

$$\begin{aligned} \dot{\vec{\delta r}} &= \vec{\delta v} \\ \dot{\vec{\delta v}} &= -\mu \frac{\vec{R}}{R^3} \end{aligned} \tag{4}$$

where $\vec{R} = \vec{r}_o(t) - \vec{r}_m(t)$ is the position vector of the spacecraft with respect to the perturbing mass.

One further assumption enables one to solve in closed form the perturbations produced by the third mass. Generally $\vec{R}(t)$ and $R(t)$ are nearly linear functions of time. Therefore suppose that the initial and final values of these variables are known to be $\vec{R}_1, \vec{R}_2, R_1, R_2$ over the interval Δt.

Introduce the definitions

$$\begin{aligned}
\vec{\Delta R} &= \vec{R}_2 - \vec{R}_1 \\
\Delta R &= R_2 - R_1 \qquad (\text{not } |\vec{\Delta R}|) \\
\langle R \rangle &= \tfrac{1}{2}(R_1 + R_2) \\
\hat{\Delta R} &= \frac{\vec{R}_2}{R_2} - \frac{\vec{R}_1}{R_1}
\end{aligned} \tag{5}$$

Then the equation defining the velocity perturbation would be

$$\dot{\vec{\delta v}} = -\mu \frac{\vec{a} + \vec{b}\,t}{(c + d\,t)^3} \qquad \vec{a} = \vec{R}_1 \qquad c = R_1 \qquad \vec{b} = \frac{\vec{\Delta R}}{\Delta t} \qquad d = \frac{\Delta R}{\Delta t} \tag{6}$$

It is more convenient however to transform from time t to position magnitude ρ as the independent variable. This may be done since the position magnitude is assumed to be linear in time with $\dot{\rho} = \frac{\Delta R}{\Delta t}$.

According to the assumptions, the position vector $\vec{R}$ is a linear function of ρ also

$$\vec{R} = \vec{A} + \vec{B}\,\rho \tag{7}$$

Since $\vec{R}(\rho_1) = \vec{R}_1$ and $\vec{R}(\rho_2) = \vec{R}_2$, the constants are

$$\begin{aligned}
\vec{A} &= \vec{R}_1 - \frac{\vec{\Delta R}}{\Delta R} R_1 = -\frac{R_1 R_2}{\Delta R} \hat{\Delta R} \\
\vec{B} &= \frac{\vec{\Delta R}}{\Delta R}
\end{aligned} \tag{8}$$

In terms of ρ the equations defining the perturbations may be written (with primes indicating differentiation with respect to ρ)

$$\begin{aligned}
\vec{\delta r}' &= \frac{\Delta t}{\Delta R} \vec{\delta v} \\
\vec{\delta v}' &= -\frac{\mu \Delta t}{\Delta R} \frac{\vec{A} + \vec{B}\rho}{\rho^3}
\end{aligned} \tag{9}$$

These equations are easily integrated to determine the perturbations caused as the spacecraft moves from $\vec{R}_1$ to $\vec{R}_2$ relative to the perturbative body:

$$\vec{\delta v} = -\frac{\mu \Delta t}{\Delta R} \int_{R_1}^{\rho} \frac{\vec{A} + \vec{B}\rho}{\rho^3} \, d\rho$$

$$= \frac{\mu \Delta t}{\Delta R} \left[\frac{\vec{A}}{2} \left(\frac{1}{\rho^2} - \frac{1}{R_1^2} \right) + \vec{B} \left(\frac{1}{\rho} - \frac{1}{R_1} \right) \right]$$

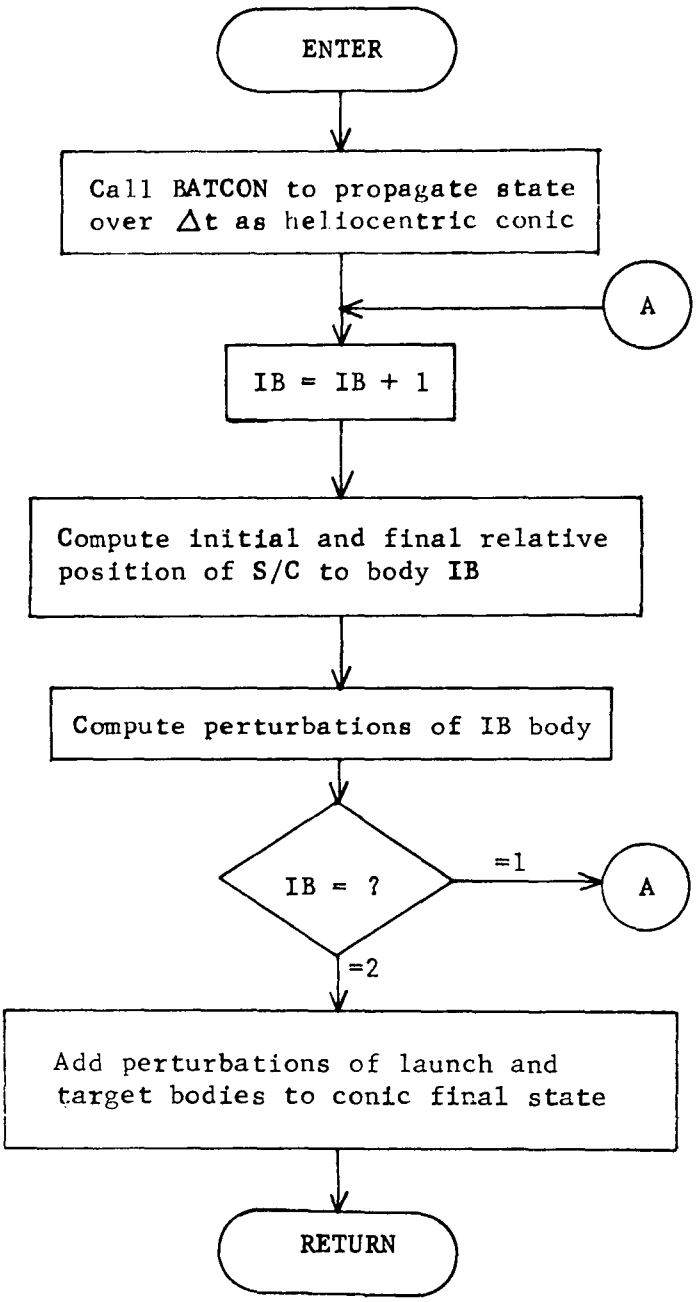
$$= \frac{\mu \Delta t}{R_1 R_2 \Delta R} \left[\langle R \rangle \hat{\Delta R} - \vec{\Delta R} \right], \qquad \rho = R_2 \tag{10}$$

$$\vec{\delta r} = \frac{\mu \Delta t^2}{\Delta R^2} \int_{R_1}^{R_2} \left[\frac{\vec{A}}{2} \left(\frac{1}{\rho^2} - \frac{1}{R_1^2} \right) + \vec{B} \left(\frac{1}{\rho} - \frac{1}{R_1} \right) \right] d\rho$$

$$= \frac{\mu \Delta t^2}{\Delta R} \left[\frac{1}{2} \frac{\hat{\Delta R}}{R_1} + \frac{\vec{\Delta R}}{\Delta R^2} \left(\ln\left(\frac{R_2}{R_1} \right) - \frac{\Delta R}{R_1} \right) \right] \tag{11}$$

PERHEL calls **BATCON** for the generation of the uncorrected heliocentric conic, computes the initial and final positions of the spacecraft relative to each of the launch and target planets, and computes the perturbations based on equations (10) and (11) above.

PERHEL Flow Chart



PLND Analysis

The nonlinear equations of motion of the spacecraft can be written symbolically as

$$\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{e}(t), t) \tag{1}$$

where $\vec{x}$ is the spacecraft position/velocity state and $\vec{e}(t)$ is a vector composed of the six orbital elements a, e, i, Ω, ω, and M of the target planet. The motion of the spacecraft is, of course, dependent on the positions of other celestial bodies, but this dependency need not be explicitly stated for the purposes of this analysis.

Suppose we wish to use numerical differencing to compute those columns of θ_{xx_s}, θ_{xu}, and θ_{xw} associated with target planet ephemeris biases included in the augmented state vector over the time interval $[t_{k-1}, t_k]$. Let $\vec{\theta}_j(t_k, t_{k-1})$ represent the column associated with the j-th ephermeris bias. We assume we have available the nominal states $\vec{x}^*(t_{k-1})$ and $\vec{x}^*(t_k)$, which, of course, were obtained by numerically solving equation (1) using nominal $\vec{e}(t)$. To obtain $\vec{\theta}_j(t_k, t_{k-1})$, we increment the j-th orbital element by the pertinent numerical differencing factor Δe_j and numerically integrate equation (1) over the interval $[t_{k-1}, t_k]$ to obtain the new spacecraft state $\vec{x}_j(t_k)$, where the j-subscript on the spacecraft state indicates that it was obtained by incrementing the j-th orbital element. Then

$$\vec{\theta}_j(t_k, t_{k-1}) = \frac{\vec{x}_j(t_k) - \vec{x}^*(t_k)}{\Delta e_j} \tag{2}$$

Ephemeris biases are defined as biases of the basic set of orbital elements a, e, i, Ω, ω, and M. However the ephemeris constants of a, e, i, Ω, $\tilde{\omega}$, and M for the planets and a, e, i, Ω, $\tilde{\omega}$, and L for the moon are stored in the program. Thus to increment certain of the basic elements, we must increment certain combinations of the stored ephemeris constants.

The elements ω and M are related to the longitude of perihelion $\tilde{\omega}$ and the mean longitude L as follows:

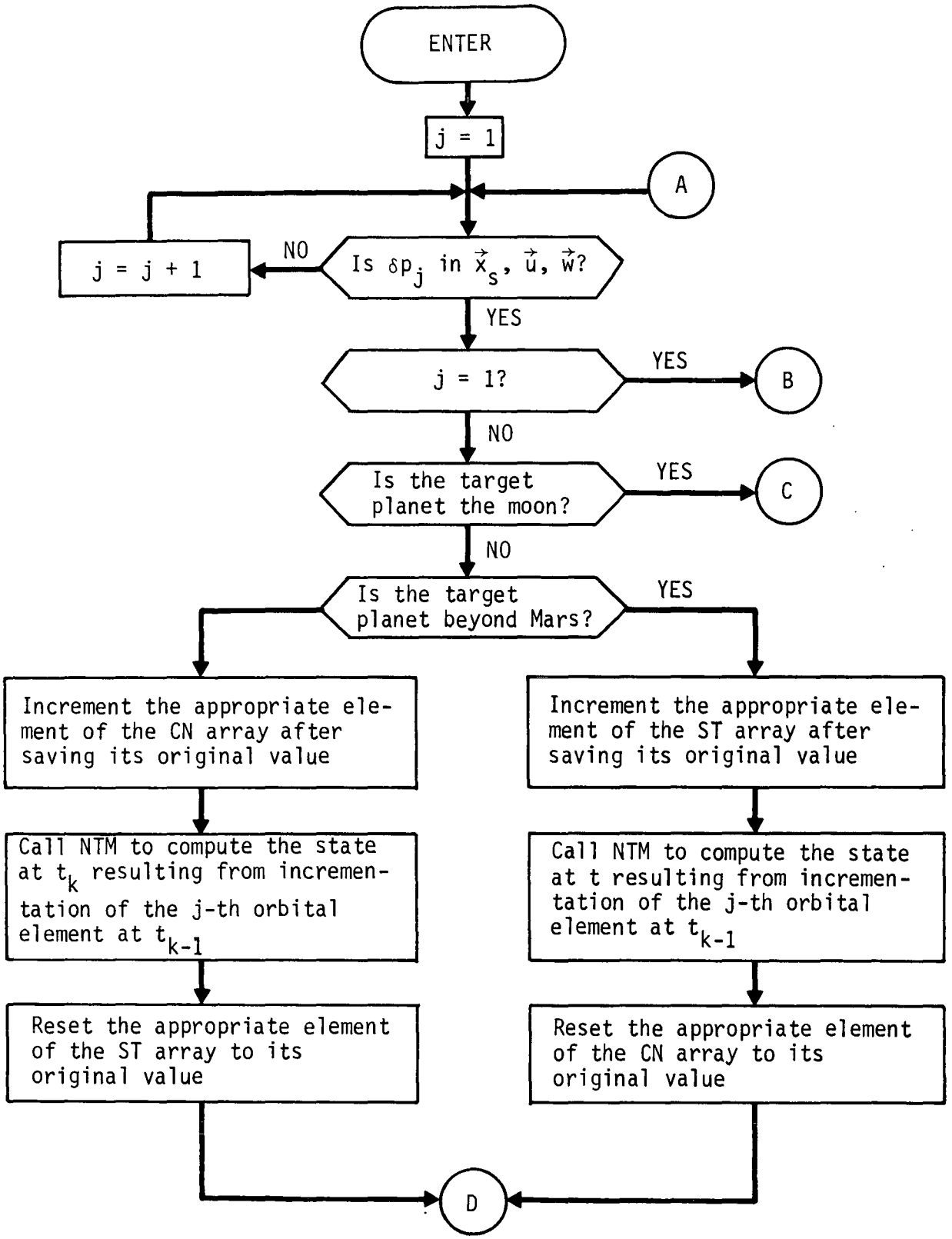
$$\omega = \tilde{\omega} - \Omega$$

$$M = L - \tilde{\omega} \quad .$$

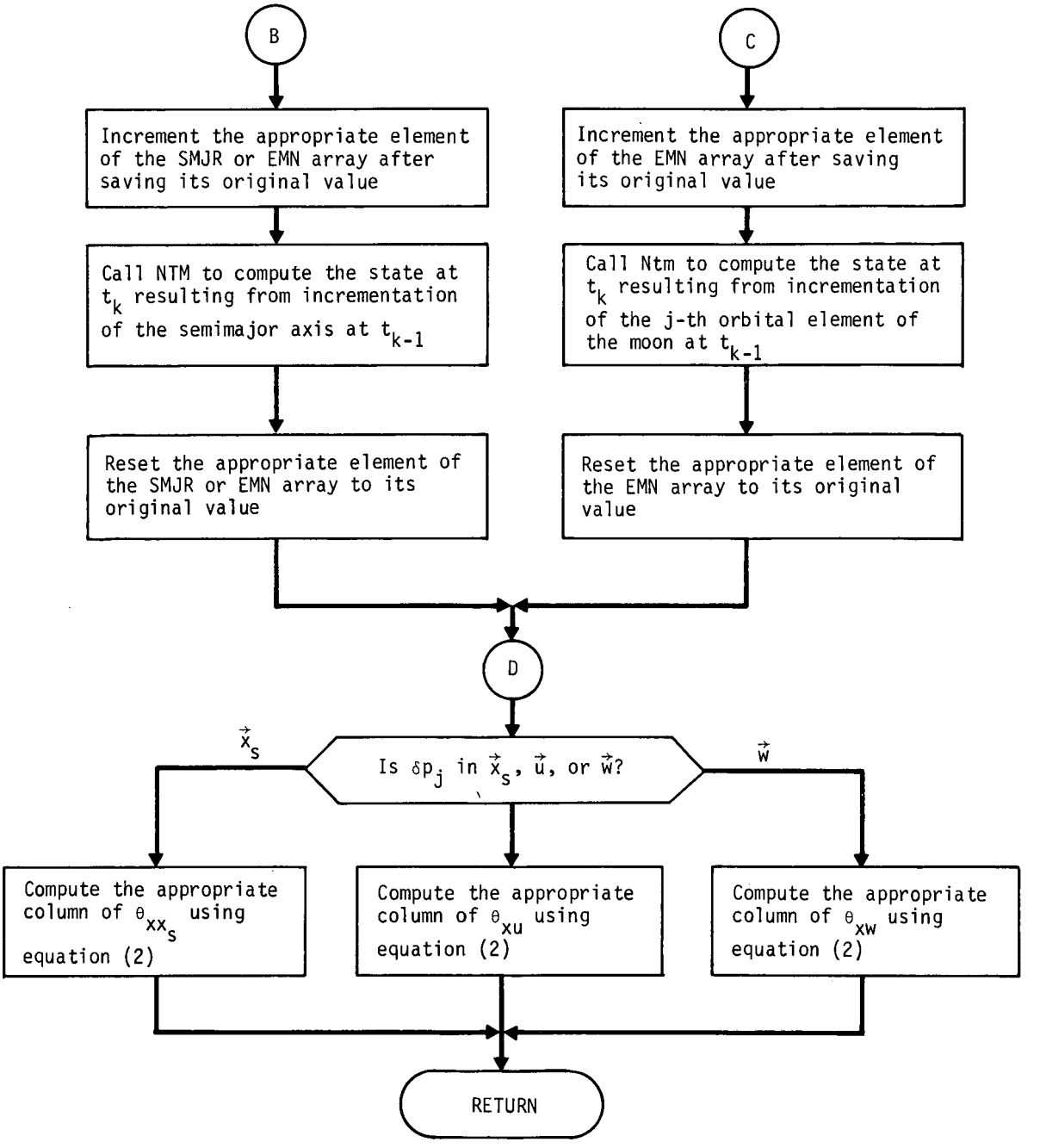
Thus, to increment Ω by $\Delta\Omega$ without changing the other five basic elements requires that we also increment $\tilde{\omega}$ by $\Delta\Omega$ for the case of a planet, and both $\tilde{\omega}$ and L by $\Delta\Omega$ for the case of the moon. To increment ω by $\Delta\omega$ we simply increment $\tilde{\omega}$ by $\Delta\omega$ for a planet, while for the moon we must increment both $\tilde{\omega}$ and L by $\Delta\omega$. To increment M by ΔM for the moon we simply increment L by ΔM.

In the PLND flow chart we employ the following definition:

$$P_j = \begin{cases} a & j = 1 \\ e & 2 \\ i & 3 \\ \Omega & 4 \\ \omega & 5 \\ M & 6 \end{cases} \quad .$$



PLND Flow Chart


B
C
Increment the appropriate element of the SMJR or EMN array after saving its original value
Increment the appropriate element of the EMN array after saving its original value
Call NTM to compute the state at t_k resulting from incrementation of the semimajor axis at t_{k-1}
Call Ntm to compute the state at t_k resulting from incrementation of the j-th orbital element of the moon at t_{k-1}
Reset the appropriate element of the SMJR or EMN array to its original value
Reset the appropriate element of the EMN array to its original value
D
$\vec{x}_s$
Is δp_j in $\vec{x}_s$, $\vec{u}$, or $\vec{w}$?
$\vec{w}$
Compute the appropriate column of θ_{xx_s} using equation (2)
Compute the appropriate column of θ_{xu} using equation (2)
Compute the appropriate column of θ_{xw} using equation (2)
RETURN

POICOM Analysis

Subroutine POICOM computes the target condition covariance W_j^+ after a guidance correction, the projection of W_j^+ into the impact plane, and the probability of impact of the spacecraft with the target planet.

The target condition covariance matrix W_j^+ is defined as

$$W_j^+ = \eta_j \, (P_{k_j}^- + M\tilde{Q}_j M^T) \, \eta_j^T$$

where η_j is the variation matrix for the appropriate guidance policy, $P_{k_j}^-$ is the knowledge covariance prior to the guidance correction, $\tilde{Q}_j$ is the execution error covariance, and M is defined as the following 6 x 3 matrix:

$$M = \left[0 \;\vdots\; I \right]^T$$

Before the probability of impact can be computed, it is necessary to compute the projection Λ_j of W_j^+ into the impact plane. The covariance Λ_j is computed as follows for each of the three available midcourse guidance policies.

a. Fixed-time-of-arrival:

$$\Lambda_j = A \, W_j^+ \, A^T$$

where transformation A is defined in the subroutine BIAIM analysis.

b. Two-variable B-plane:

$$\Lambda_j = W_j^+$$

c. Three-variable B-plane:

$$\Lambda_j = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} W_j^+ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Assuming the probability density function associated with Λ_j is Gaussian and nearly constant over the target planet capture area permits us to compute the probability of impact using the equation

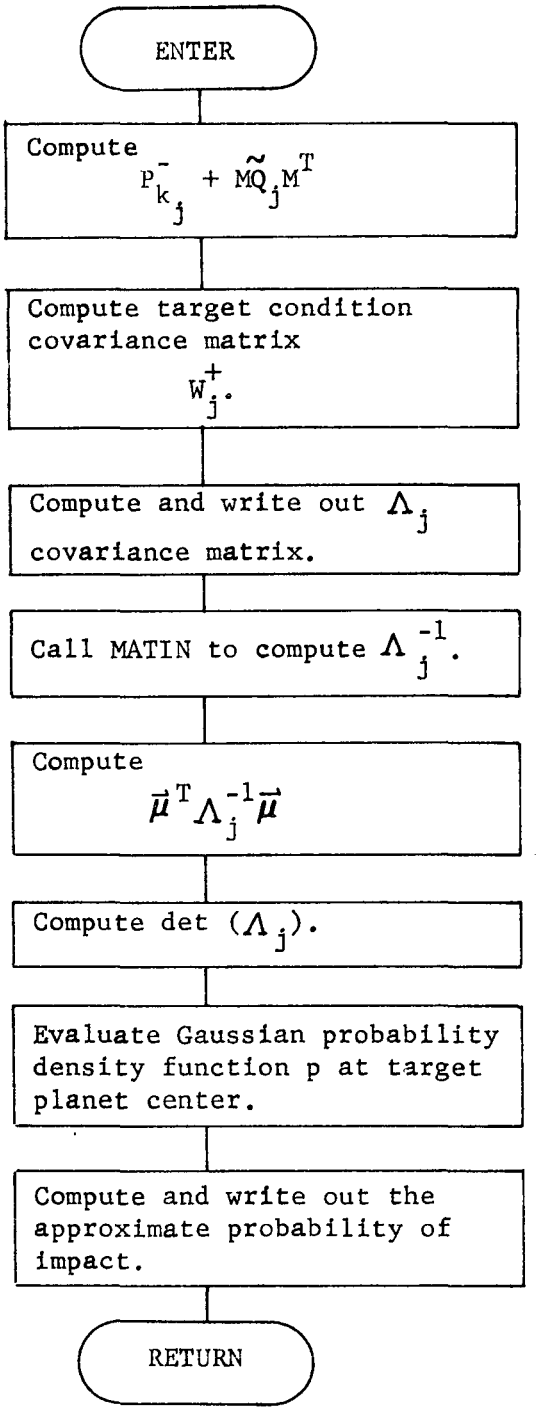
$$POI = \pi R_c^2 \, p$$

where R_c is the target planet capture radius and p is the Gaussian probability density function evaluated at the target planet center and given by

$$p = \frac{1}{2\pi |\Lambda_j|^{1/2}} \exp\left[-\frac{1}{2} \vec{\mu}^T \Lambda_j^{-1} \vec{\mu} \right]$$

where $\vec{\mu}$ is the aimpoint in the impact plane.

POICOM Flow Chart



PRED Analysis

Subroutine PRED executes a prediction event in the error analysis/ generalized covariance analysis program. Subroutine PRED differs from subroutine PRESIM in three respects. First, the propagated knowledge covariance matrix partitions are based on the (most recent) targeted nominal, rather than on the most recent nominal as in PRESIM. Second, estimated position/velocity deviations are not propagated in PRED since estimates are processed only in the simulation program and not in the error analysis program. And third, subroutine PRED treats both assumed and actual knowledge covariance matrix partitions, whereas subroutine PRESIM treats only assumed knowledge covariance matrix partitions. Subroutine PRED uses the propagation equations in subroutine GNAVM to propagate both assumed and actual covariances.

A flow chart for PRED is not presented here because of its similarity to the PRESIM flow chart (see PRESIM for further details).

PRELIM Analysis

PRELIM is responsible for the preliminary work required by NOMNAL including the initialization of variables, the reading of input, and the computation of zero iterate values for initial time, position, and velocity if necessary.

On the first call to PRELIM, PRELIM presets constants to be used on the entire series of runs. These constants include the double precision numbers and the launch profile parameters. On subsequent calls these variables are not reset.

PRELIM then presets constants for individual runs. These constants presently include most of the guidance event parameters. The user may easily change the two sets of constants for his particular needs.

PRELIM then accepts the input data. It reads data in the NAMELIST format.

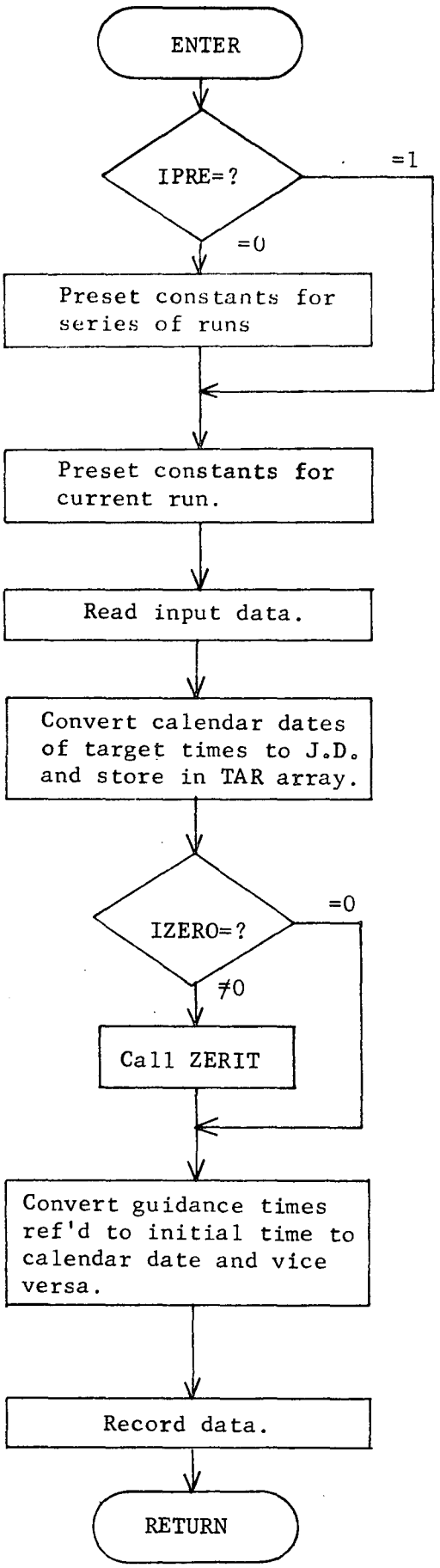
Target times must be read in as calendar dates. PRELIM next converts these to Julian date referenced 1900 and stores the converted values in the TAR array.

If the flag IZERO is nonzero, ZERIT is called for the computation of the zero iterate values of initial time, position, and velocity. ZERIT in turn calls HELIO for interplanetary trajectories and LUNA for lunar trajectories.

PRELIM then converts guidance event times referenced to initial time to calendar data and converts times read in as calendar dates to times referenced to the initial time. When the latter is done, it sets KTIM to acknowledge that conversion.

Finally PRELIM records all pertinent data.

PRELIM Flow Chart



PREPUL Analysis

PREPUL is responsible for performing the preliminary computations required for the pulsing arc model.

PREPUL first determines the nominal pulsing arc. Let the following definitions be made:

T	magnitude of pulsing engine thrust
m	nominal mass of spacecraft
Δt	duration of single pulse
Δt_i	time interval between pulses
$\overrightarrow{\Delta v}$	total velocity increment to be added

The velocity increment imparted by a single pulse is

$$\Delta v_i = \frac{T \, \Delta t}{m} \tag{1}$$

The number of pulses required is then

$$N_p = \left[\frac{\Delta v}{\Delta v_i} \right] + 1 \tag{2}$$

where $[\cdot]$ denotes the greatest integer function. The magnitude of the final pulse must be set to

$$\Delta v_f = \Delta v - (N_p - 1) \, \Delta v_i \tag{3}$$

The vector nominal pulse and final pulse are therefore given by

$$\begin{aligned} \overrightarrow{\Delta v_i} &= \Delta v_i \, \frac{\overrightarrow{\Delta v}}{\Delta v} \\ \overrightarrow{\Delta v_f} &= \Delta v_f \, \frac{\overrightarrow{\Delta v}}{\Delta v} \end{aligned} \tag{4}$$

The duration of the pulsing arc is then given by

$$\Delta T = (N_p - 1) \, \Delta t_i \tag{5}$$

Later computations require time histories of the position vectors of the launch and target bodies. An efficient means of obtaining this involves the f and g series. Given the state $\vec{r}_o$, $\vec{v}_o$ of body moving in a conic section about a central body of gravitational constant μ, the position vector as a function of t measured from the initial time is given by

$$\vec{r}(t) = f(t) \, \vec{r}_o + g(t) \, \vec{v}_o \tag{6}$$

where

$$f(t) = \sum_{k=0}^{n} f_k t^k \qquad g(t) = \sum_{k=1}^{n} g_k t^k \tag{7}$$

The constants f_k, g_k are computed in PREPUL as

$$f_o = 1$$

$$f_1 = 0$$

$$f_2 = \frac{-\mu}{2r_o^3}$$

$$f_3 = \frac{\mu \dot{r}_o}{2r_o^4}$$

$$f_4 = \frac{\mu^2}{24r_o^6}\left(4 - 15\frac{r_o \dot{r}_o^2}{\mu} - 3\frac{r_o}{a}\right)$$

$$f_5 = \frac{-\mu^2 \dot{r}_o}{8r_o^7}\left(4 - \frac{7r_o\dot{r}_o^2}{\mu} - 3\frac{r_o}{a}\right)$$

$$f_6 = \frac{\mu^3}{720\, r_o^9}\left[-70 + 114\frac{r_o}{a} + 840\frac{r_o\dot{r}_o^2}{\mu} - 630\frac{r_o^2\dot{r}_o^2}{\mu a} - 450\left(\frac{r_o\dot{r}_o^2}{\mu}\right)^2 - 45\frac{r_o^2}{a^2}\right]$$

$$g_1 = 1$$

$$g_2 = 0$$

$$g_3 = \frac{1}{3} f_2$$

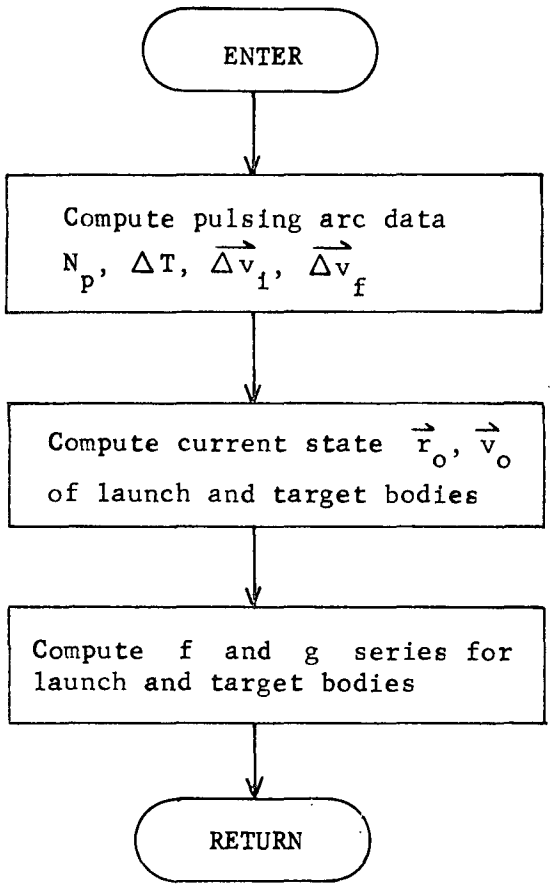
$$g_4 = \frac{1}{2} f_3$$

$$g_5 = \frac{3}{5} f_4 - \frac{1}{15} f_2^2$$

$$g_6 = \frac{2}{3} f_5 - \frac{1}{6} f_2 f_3$$

Reference: Baker, R. M. L. and Makemson, M. W., An Introduction to Astrodynamics, Academic Press, New York, 1967.

PREPUL Flow Chart



PRESIM Analysis

Subroutine PRESIM executes a prediction event in the simulation program SIMUL. At a prediction event, the knowledge covariance partitions, and the estimated position/velocity deviations from the most recent nominal trajectory are propagated forward to t_p, the time to which the prediction is to be made. The knowledge covariance partitions are propagated using the prediction equations found in the NAVM Analysis section. The estimate is propagated using the equation

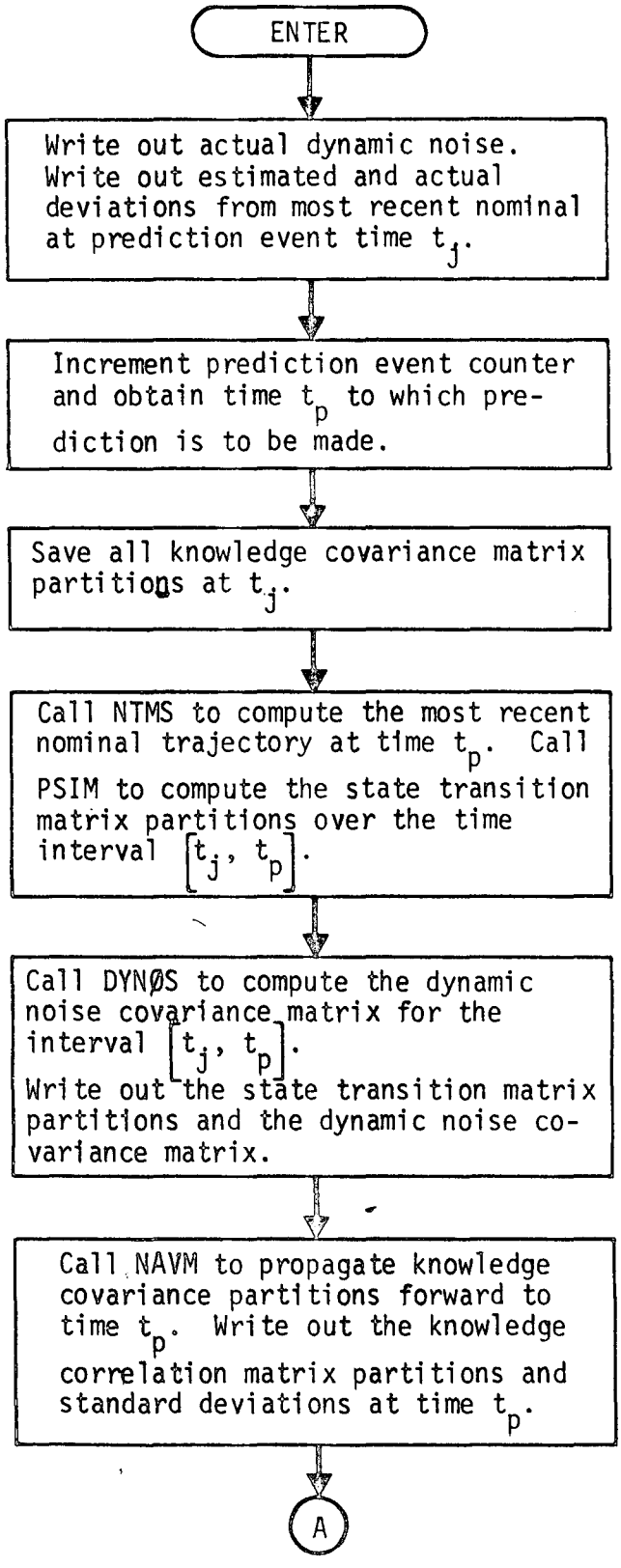
$$\delta\tilde{\hat{X}}_p = \Phi\ (t_p,\ t)\ \delta\tilde{\hat{X}}_j + \theta_{xx_s}\ (t_p,\ t_j)\ \delta\tilde{\hat{X}}_{s_j}$$

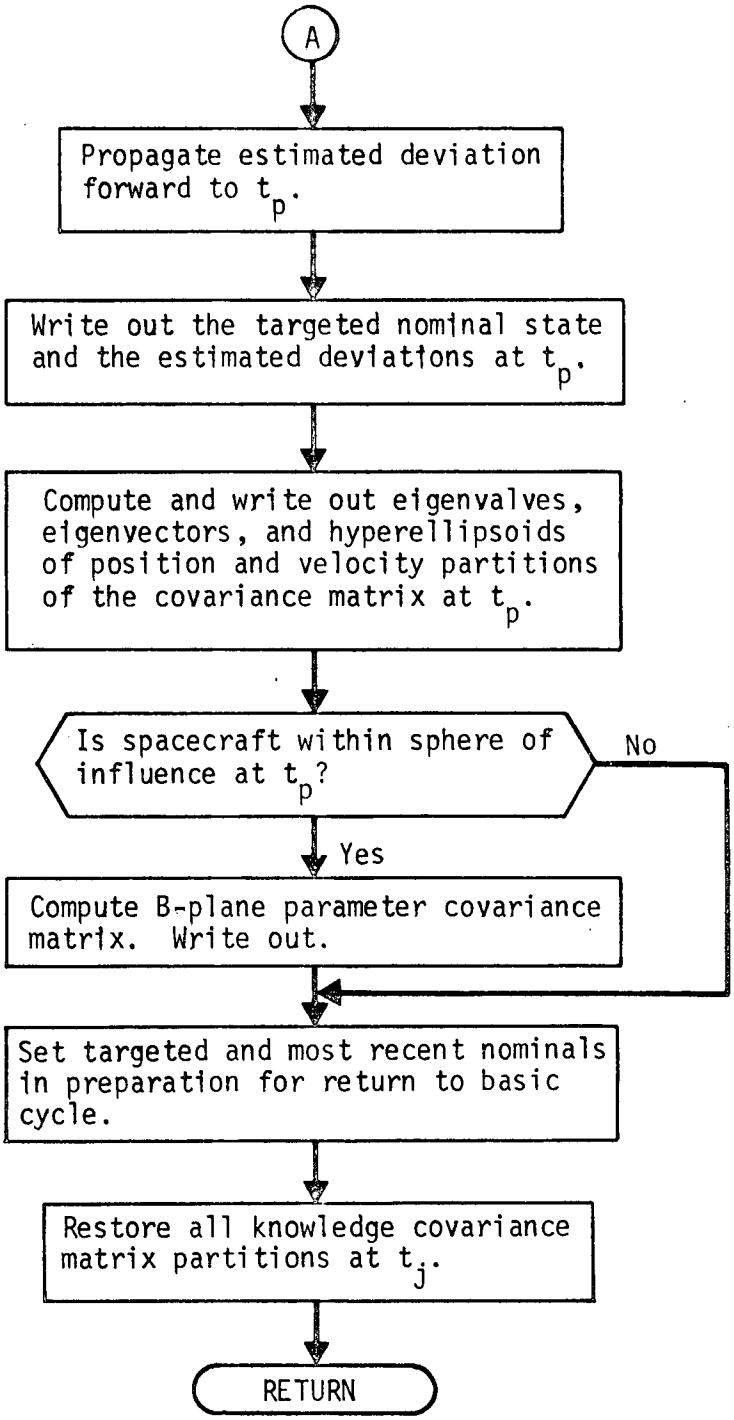
where Φ and θ_{xx_s} are the state transition matrix partitions over the time interval $[t_j,\ t_p]$.

The position and velocity partitions of the propagated knowledge covariance matrix are diagonalized at time t_p and the eigenvalues, eigenvectors, and hyperellipsoids are computed.

If t_p occurs within the target planet sphere of influence, the Cartesian position/velocity covariance matrix is transformed to a B-plane parameter covariance matrix. The B-plane parameters are B·T, B·R, time-of-flight, S·R, S·T, and C_3.

PRESIM Flow Chart




A
Propagate estimated deviation forward to t_p.
Write out the targeted nominal state and the estimated deviations at t_p.
Compute and write out eigenvalves, eigenvectors, and hyperellipsoids of position and velocity partitions of the covariance matrix at t_p.
Is spacecraft within sphere of influence at t_p?
No
Yes
Compute B-plane parameter covariance matrix. Write out.
Set targeted and most recent nominals in preparation for return to basic cycle.
Restore all knowledge covariance matrix partitions at t_j.
RETURN

PRØBE Analysis

Subroutine PRØBE controls the execution of both main probe and miniprobe release events. When a probe release event occurs, PRØBE saves all states, covariance matrices, etc relating to the bus and initializes all states, covariance matrices, etc for the probe under consideration. The probe state at release is then propagated forward to entry, along with the probe control covariance matrix partitions to obtain the probe state and control dispersions at entry. Next, the probe is tracked from release to entry and probe knowledge covariance matrix partitions are propagated and updated accordingly.

Let t_j be the time of probe release and let $\bar{X}_j$ denote the nominal bus state at release. Denote all main probe quantities with a superscript zero, and all miniprobe quantities with a superscript i (for the ith miniprobe where i = 1, 2, 3). Then, following release, the probe states are given by

$$\bar{X}_j^0 = \bar{X}_j \tag{1}$$

$$\bar{X}_j^i = \bar{X}_j + \begin{bmatrix} 0 \\ \hline \Delta V_j^{(i)} \end{bmatrix} \tag{2}$$

where $\Delta V_j^{(i)}$ is the velocity increment imparted to the ith miniprobe by the spin release at t_j. The velocity increment $\Delta V_j^{(i)}$ and the miniprobe release controls are used in subroutine MINIQ to compute the execution error covariance matrix $\tilde{Q}_j^{(i)}$ associated with the spin release of the ith miniprobe. Denote the bus position/velocity knowledge and control covariance matrices at release by P_{k_j} and P_{c_j}, respectively. Then, immediately following release, the probe position/velocity knowledge and control covariance matrices are given by

$$P_{k_j}^0 = P_{k_j} \tag{3}$$

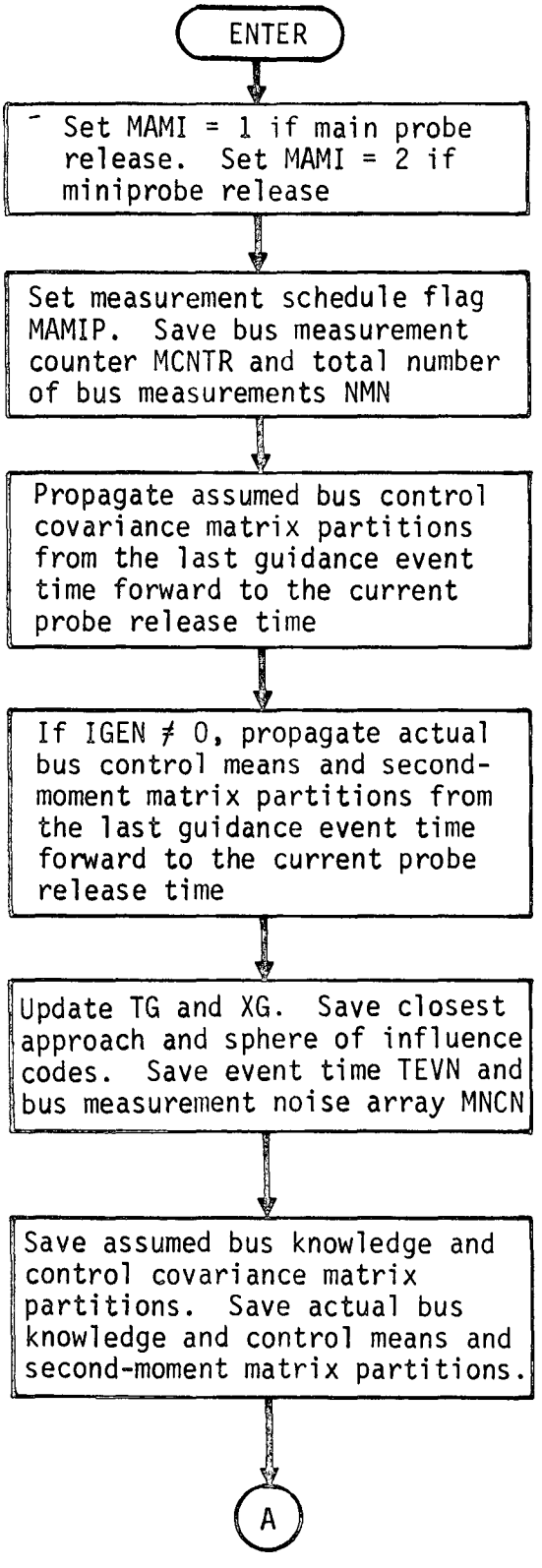
$$P_{c_j}^0 = P_{c_j} \tag{4}$$

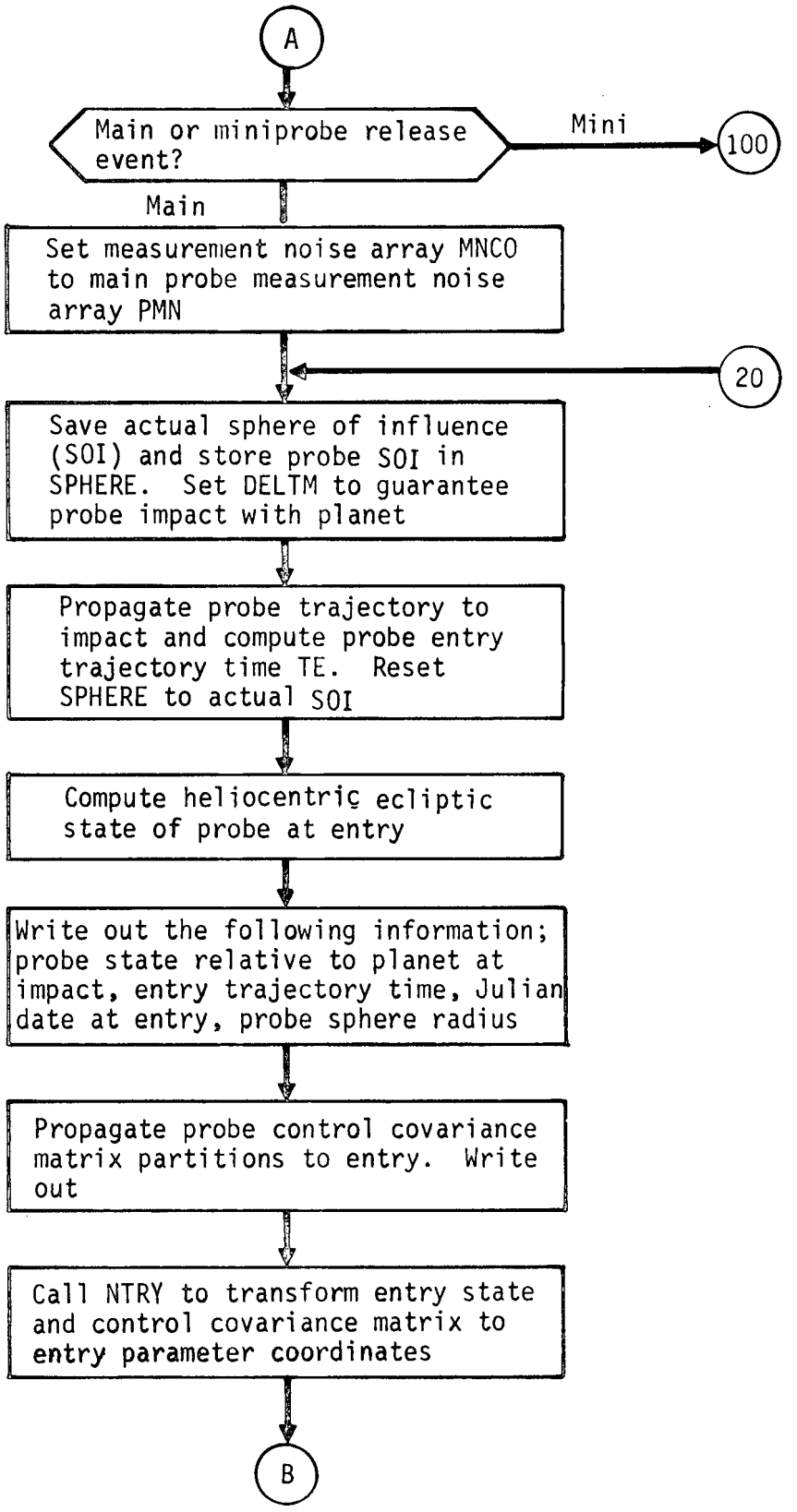
$$P_{k_j}^{(i)} = P_{k_j} + \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \tilde{Q}_j^{(i)} \end{array}\right] \tag{5}$$

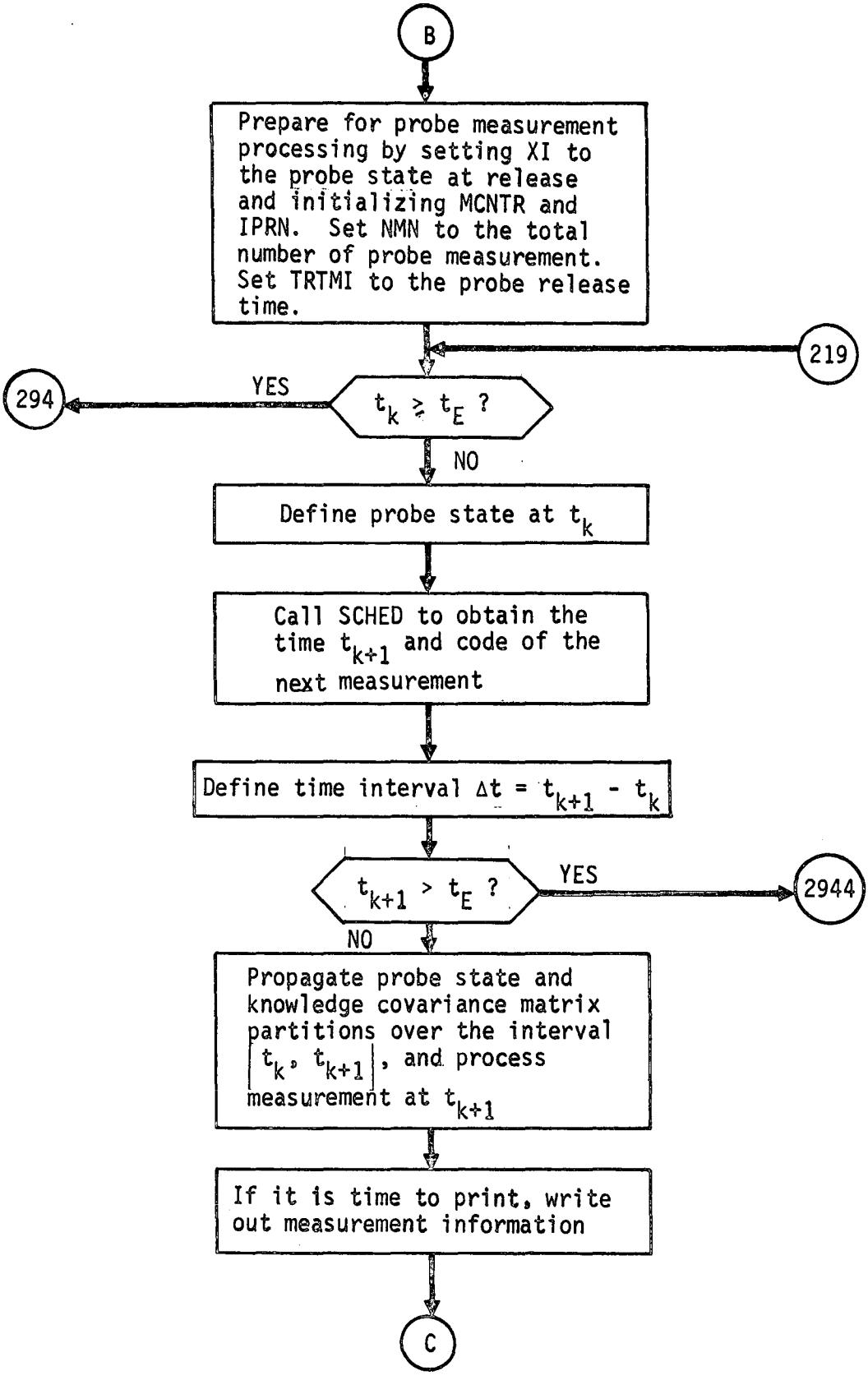
$$P_{c_j}^{(i)} = P_{k_j}^{(i)} \quad . \tag{6}$$

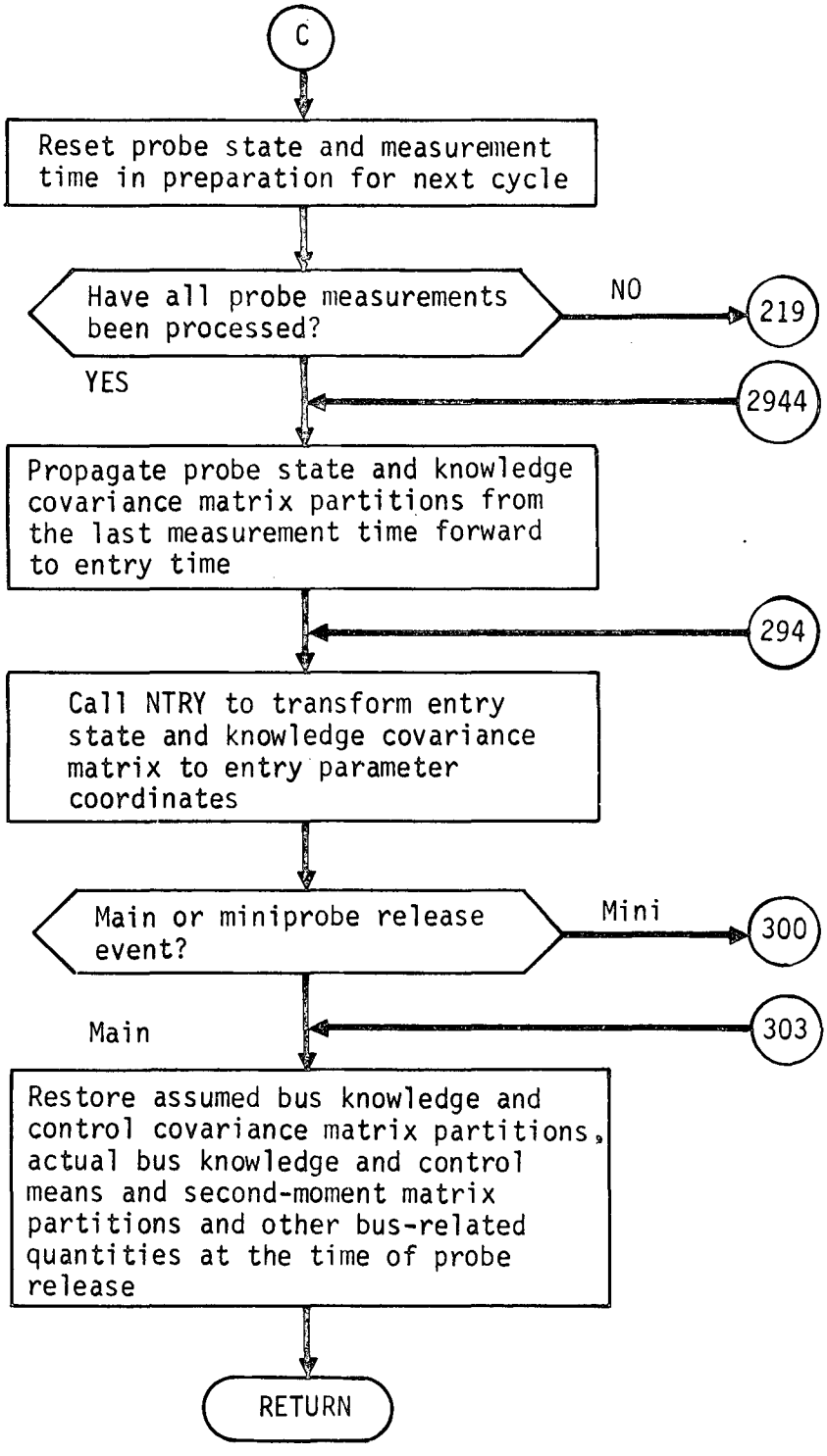
The above probe control matrices, along with all related partitions, are propagated forward to entry time t_E using the propagation equations appearing in subroutine GNAVM to obtain the probe entry dispersions. Beginning with the above knowledge covariance matrix of the probe under consideration and all related partitions, the probe knowledge covariance matrix partitions are propagated and updated as each probe measurement is processed using the update equations appearing in subroutine GNAVM.

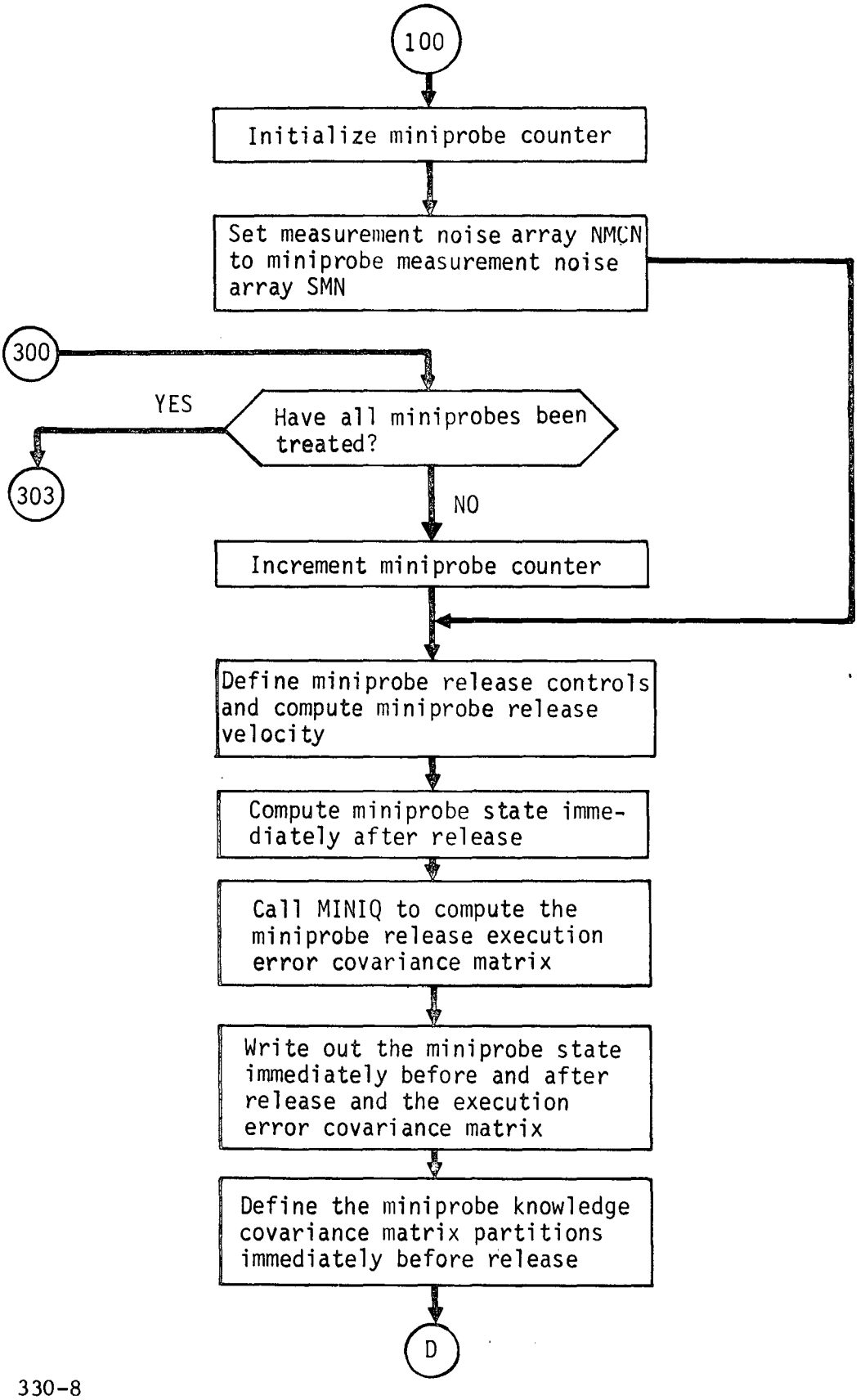
PRØBE Flow Chart

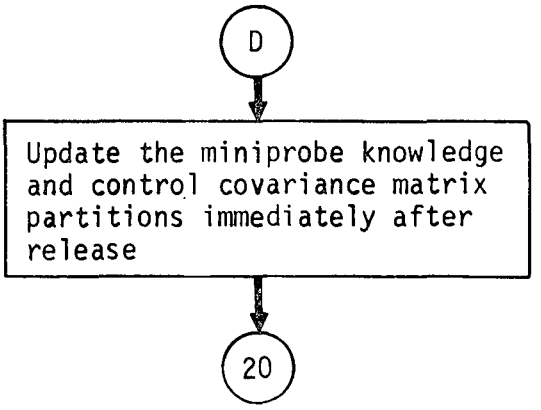



A
Main or miniprobe release event?
Mini
100
Main
Set measurement noise array MNCO to main probe measurement noise array PMN
20
Save actual sphere of influence (SOI) and store probe SOI in SPHERE. Set DELTM to guarantee probe impact with planet
Propagate probe trajectory to impact and compute probe entry trajectory time TE. Reset SPHERE to actual SOI
Compute heliocentric ecliptic state of probe at entry
Write out the following information; probe state relative to planet at impact, entry trajectory time, Julian date at entry, probe sphere radius
Propagate probe control covariance matrix partitions to entry. Write out
Call NTRY to transform entry state and control covariance matrix to entry parameter coordinates
B


B
Prepare for probe measurement processing by setting XI to the probe state at release and initializing MCNTR and IPRN. Set NMN to the total number of probe measurement. Set TRTMI to the probe release time.
219
294
YES
$t_k \geq t_E$?
NO
Define probe state at t_k
Call SCHED to obtain the time t_{k+1} and code of the next measurement
Define time interval $\Delta t = t_{k+1} - t_k$
$t_{k+1} > t_E$?
YES
2944
NO
Propagate probe state and knowledge covariance matrix partitions over the interval $[t_k, t_{k+1}]$, and process measurement at t_{k+1}
If it is time to print, write out measurement information
C


C
Reset probe state and measurement time in preparation for next cycle
Have all probe measurements been processed?
NO
219
YES
2944
Propagate probe state and knowledge covariance matrix partitions from the last measurement time forward to entry time
294
Call NTRY to transform entry state and knowledge covariance matrix to entry parameter coordinates
Main or miniprobe release event?
Mini
300
Main
303
Restore assumed bus knowledge and control covariance matrix partitions, actual bus knowledge and control means and second-moment matrix partitions and other bus-related quantities at the time of probe release
RETURN


100
Initialize miniprobe counter
Set measurement noise array NMCN to miniprobe measurement noise array SMN
300
YES
Have all miniprobes been treated?
303
NO
Increment miniprobe counter
Define miniprobe release controls and compute miniprobe release velocity
Compute miniprobe state immediately after release
Call MINIQ to compute the miniprobe release execution error covariance matrix
Write out the miniprobe state immediately before and after release and the execution error covariance matrix
Define the miniprobe knowledge covariance matrix partitions immediately before release
D


D
Update the miniprobe knowledge and control covariance matrix partitions immediately after release
20

PRØBS Analysis

Subroutine PRØBS controls the execution of both main probe and miniprobe release events in the simulation program SIMUL. When a probe release event occurs, PRØBS saves all nominal and actual states, deviation estimates, covariance matrices, etc relating to the bus and initializes all nominal and actual states, deviation estimates, covariance matrices, etc for the probe under consideration. The probe state at release is the propagated forward to entry, along with the probe control covariance matrix partitions, to obtain the probe state and control dispersions at entry. Next the probe is tracked from release to entry and probe knowledge covariance matrix partitions and deviation estimates are propagated and updated accordingly.

Let t_j denote the time of probe release. Denote the bus targeted nominal and most recent nominal states at release by $\bar{X}_j$ and $\tilde{X}_j$, respectively. Denote actual and estimated deviations of the bus state from the most recent nominal by $\delta\tilde{X}_j$ and $\delta\hat{\tilde{X}}_j$, respectively. All main probe quantities will be denoted with a superscript zero and all miniprobe quantities with a superscript i (for the ith miniprobe, where i = 1,2,3). Then following release, the probe nominal states are given by

$$\bar{X}_j^0 = \bar{X}_j \tag{1}$$

$$\tilde{X}_j^0 = \tilde{X}_j \tag{2}$$

$$\bar{X}_j^i = \tilde{X}_j + \delta\hat{\tilde{X}}_j + \left[\begin{array}{c} -\frac{0}{\Delta V_j^i}- \end{array}\right] \tag{3}$$

$$\tilde{X}_j^i = \bar{X}_j^i \tag{4}$$

where ΔV_j^i is the velocity increment imparted to the ith miniprobe by the spin-release at t_j. The velocity increment ΔV_j^i and the miniprobe release controls are used in subroutine MINIQ to compute the execution error covariance matrix $\tilde{Q}_j^i$ and the actual execution $\delta\Delta V_j^i$.

The actual and estimated deviations of the probe states from the most recent nominals are given by

$$\delta\tilde{\hat{X}}_j^0 = \delta\tilde{\hat{X}}_j \tag{5}$$

$$\delta\tilde{X}_j^0 = \delta\tilde{X}_j \tag{6}$$

$$\delta\tilde{\hat{X}}_j^i = 0 \tag{7}$$

$$\delta\tilde{X}_j^i = \delta\tilde{X}_j - \delta\tilde{\hat{X}}_j + \begin{bmatrix} -\frac{0}{} - \\ \delta\Delta V_j^i \end{bmatrix} . \tag{8}$$

Denote the bus position/velocity knowledge and control covariance matrices at release by P_{K_j} and P_{c_j}, respectively. Then, immediately following release, the probe position/velocity knowledge and control covariance matrices are given by

$$P_{K_j}^0 = P_{K_j} \tag{9}$$

$$P_{c_j}^0 = P_{c_j} \tag{10}$$

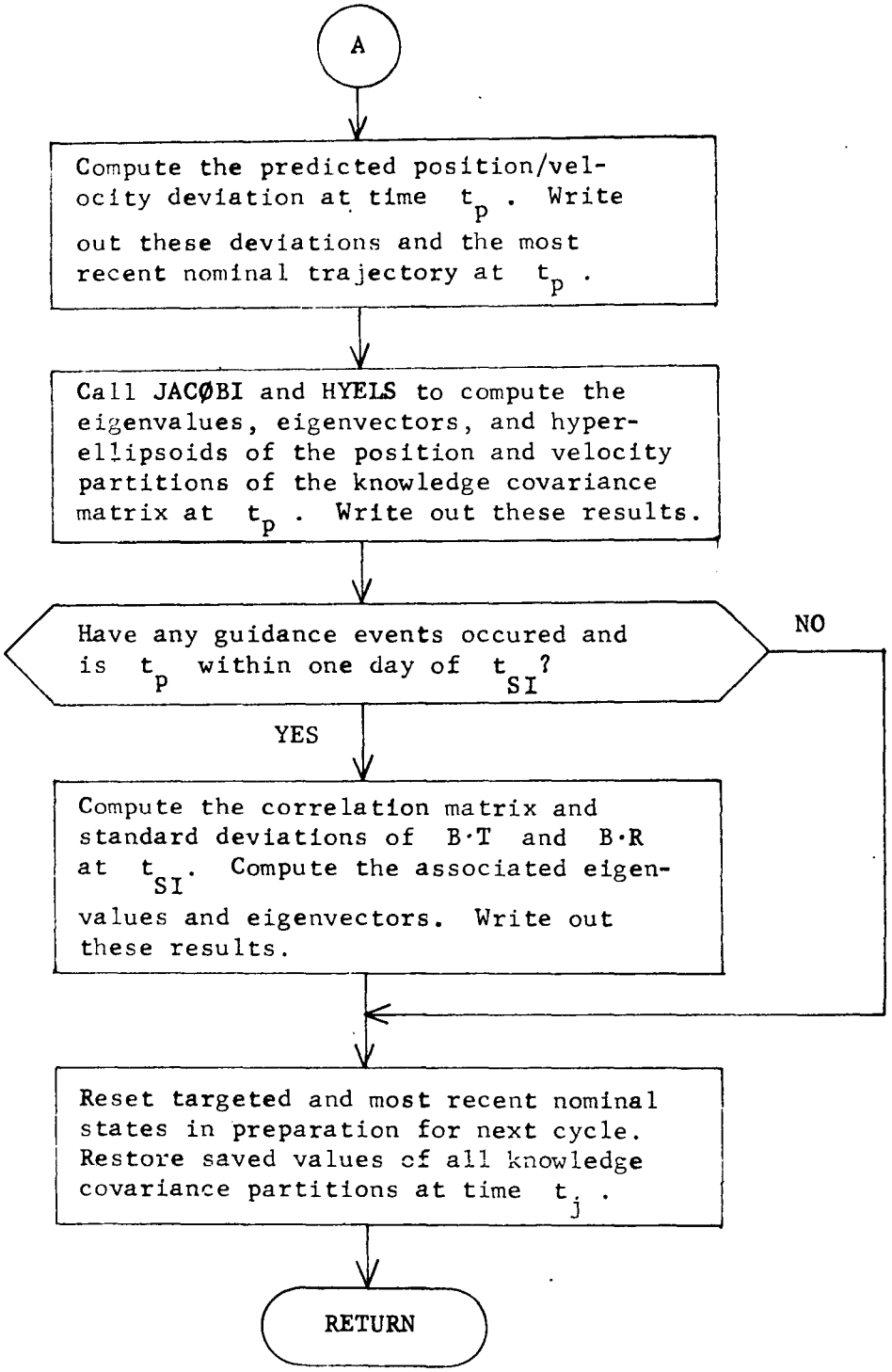
$$P_{K_j}^i = P_{K_j} + \left[\begin{array}{c|c} -\frac{0}{} - & -\frac{0}{} - \\ 0 & \tilde{Q}_j^i \end{array}\right] \tag{11}$$

$$P_{c_j}^i = P_{K_j}^i . \tag{12}$$

The above probe control covariance matrices, along with all related partitions, are propagated forward to entry time t_E using the propagation equations appearing in subroutine NAVM to obtain the probe entry dispersions. Beginning with the above knowledge covariance matrix of the probe under consideration and all related partitions, the probe knowledge covariance matrix partitions are

propagated and updated as each probe measurement is processed using the equations appearing in subroutine NAVM. The probe estimates are propagated and updated using the equations appearing in subroutine SIMUL.

A flow chart for subroutine PRØBS is not presented since it would be quite similar to the flow chart for subroutine PRØBE (see subroutine PRØBE for details).


A
Compute the predicted position/velocity deviation at time t_p. Write out these deviations and the most recent nominal trajectory at t_p.
Call JACØBI and HYELS to compute the eigenvalues, eigenvectors, and hyperellipsoids of the position and velocity partitions of the knowledge covariance matrix at t_p. Write out these results.
Have any guidance events occured and is t_p within one day of t_{SI}?
NO
YES
Compute the correlation matrix and standard deviations of B·T and B·R at t_{SI}. Compute the associated eigenvalues and eigenvectors. Write out these results.
Reset targeted and most recent nominal states in preparation for next cycle. Restore saved values of all knowledge covariance partitions at time t_j.
RETURN

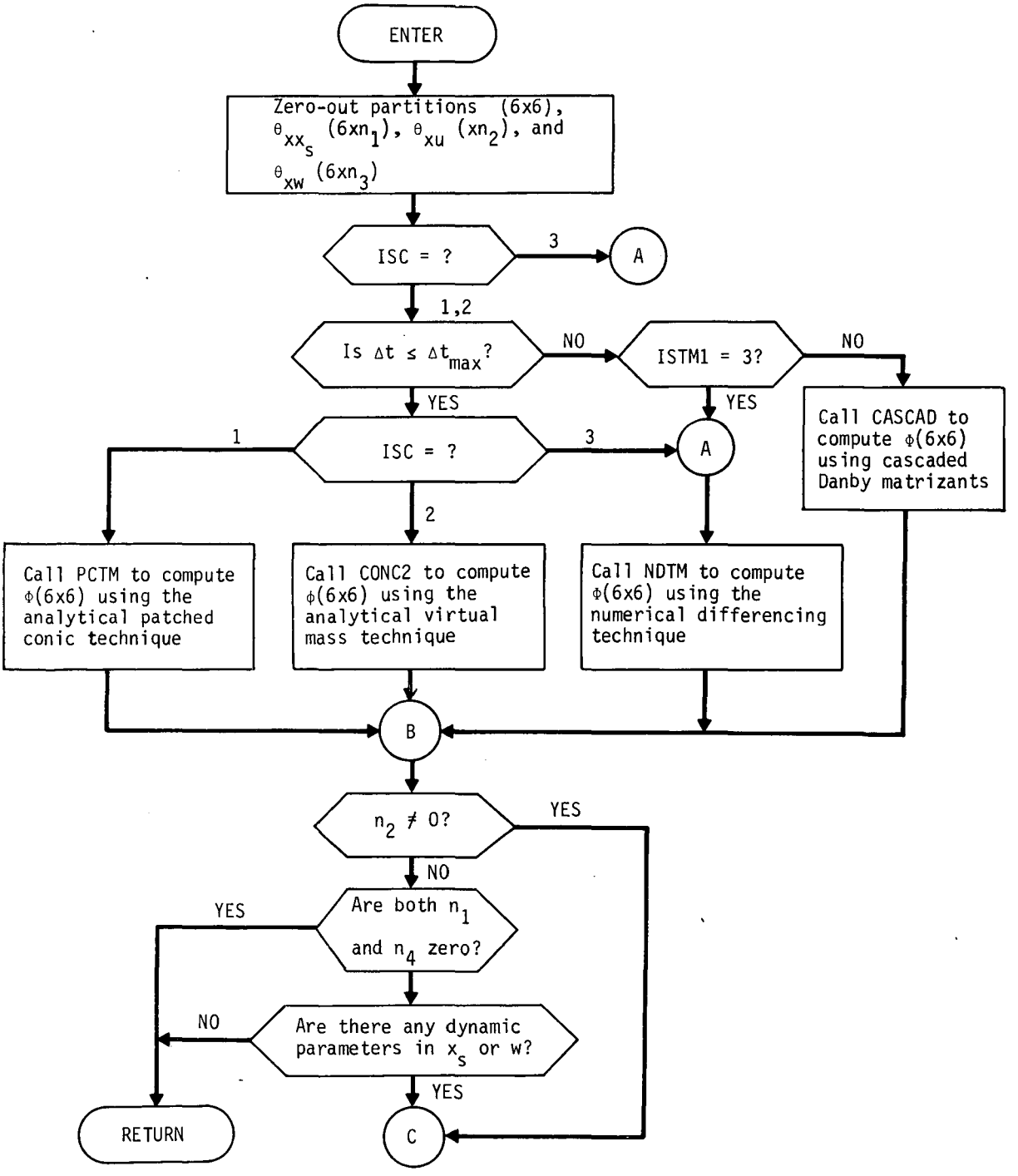
PSIM Analysis

Subroutine PSIM controls the computation of each partition appearing in the augmented state transition matrix

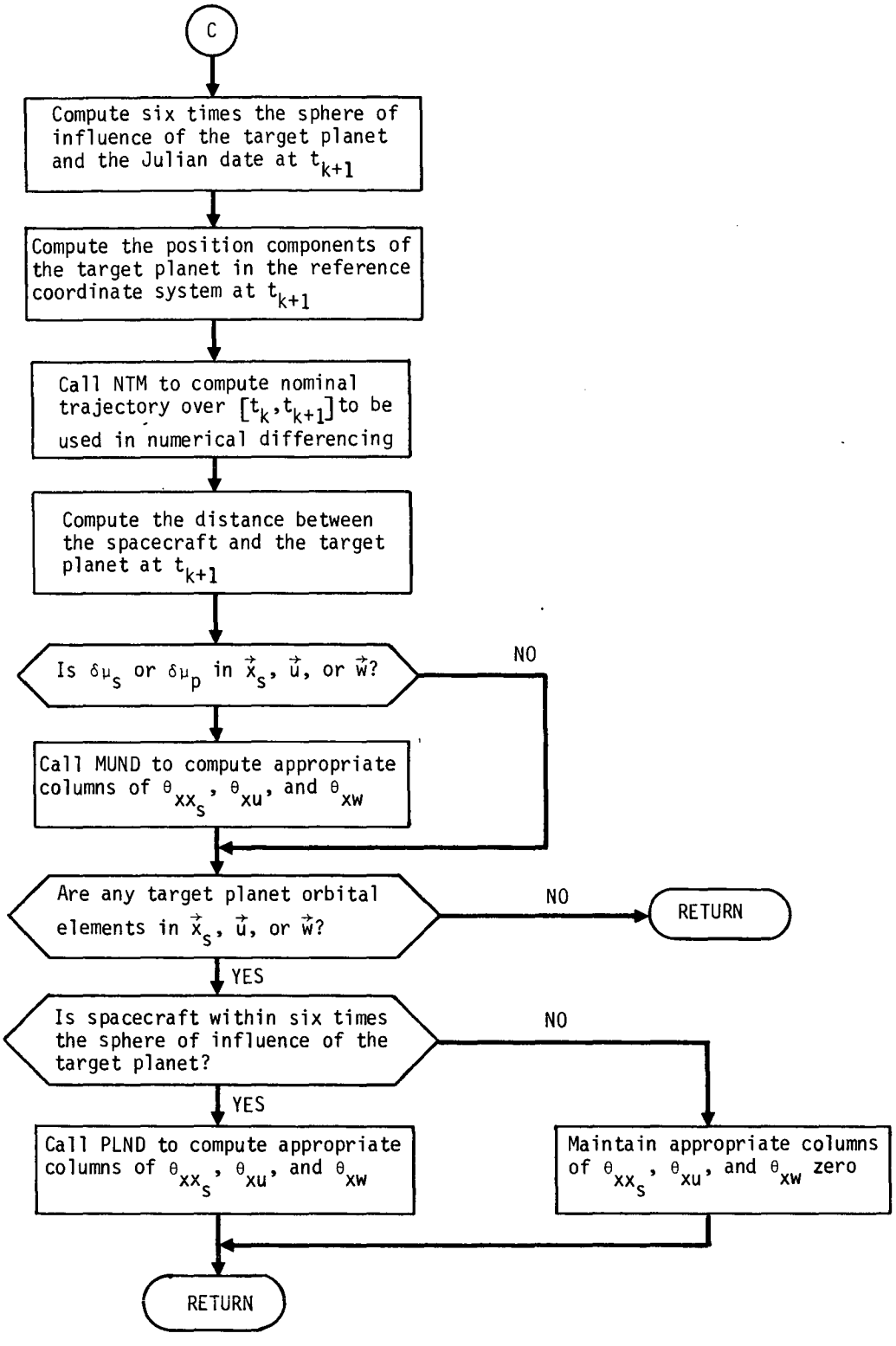
$$\Phi^{A}(k+1,k) = \begin{bmatrix} \Phi(k+1,k) & \theta_{xx_s}(k+1,k) & \theta_{xu}(k+1,k) & 0 & \theta_{xw}(k+1,k) \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}$$

The first part of the subroutine deals solely with the computation of $\Phi(k+1,k)$ by one of the three techniques -- analytical patched conic, analytical virtual mass, or numerical differencing. If an analytical technique is selected for computing $\Phi(k+1,k)$ over an interval of time greater than the maximum time interval for which the analytical technique is considered valid, we compute $\Phi(k+1,k)$ using numerical differencing or by cascading Danby matrizants.

The remaining partitions, θ_{xx_s}, θ_{xu}, and θ_{xw}, are always computed by numerical differencing. Columns in these partitions associated with target planet gravitational constant or orbital elements are computed only if the spacecraft is within six times the sphere of influence of the target planet at t_{k+1}. Otherwise, these columns are set to zero.



PSIM Flow Chart


C
Compute six times the sphere of influence of the target planet and the Julian date at t_{k+1}
Compute the position components of the target planet in the reference coordinate system at t_{k+1}
Call NTM to compute nominal trajectory over $[t_k, t_{k+1}]$ to be used in numerical differencing
Compute the distance between the spacecraft and the target planet at t_{k+1}
Is $\delta\mu_s$ or $\delta\mu_p$ in $\vec{x}_s$, $\vec{u}$, or $\vec{w}$?
NO
Call MUND to compute appropriate columns of θ_{xx_s}, θ_{xu}, and θ_{xw}
Are any target planet orbital elements in $\vec{x}_s$, $\vec{u}$, or $\vec{w}$?
NO
RETURN
YES
Is spacecraft within six times the sphere of influence of the target planet?
NO
YES
Call PLND to compute appropriate columns of θ_{xx_s}, θ_{xu}, and θ_{xw}
Maintain appropriate columns of θ_{xx_s}, θ_{xu}, and θ_{xw} zero
RETURN

PULCØV Analysis

PULCØV processes the control covariance through the pulsing arc to determine a measure of the probabilistic deviation of the corrected trajectory from the desired trajectory resulting from execution errors.

The pulsing arc itself is computed in PREPUL. It consists of N_p-1 pulses $\overrightarrow{\Delta v}_i$ and a final pulse $\overrightarrow{\Delta v}_f$ satisfying

$$(N_p-1)\ \overrightarrow{\Delta v}_i + \overrightarrow{\Delta v}_f = \overrightarrow{\Delta v} \qquad (1)$$

where $\overrightarrow{\Delta v}$ is the equivalent single impulse. The pulses are separated by a time interval Δt_i . The duration of the entire sequence of pulses is given by $\Delta T = (N_p-1)\ \Delta t_i$.

PULCØV must compute the execution error matrices Q, Q_f corresponding to the nominal pulse $\overrightarrow{\Delta v}_i$ and the final pulse $\overrightarrow{\Delta v}_f$ respectively. The error model for the engine is defined by the input specifications

σ_k^2 = proportionality error

σ_k^2 = resolution error

σ_α^2 = first pointing error

σ_β^2 = second pointing error

The execution error matrix measuring the probabilistic deviation of the actual velocity increment from the desired velocity increment is computed by QCØMP.

The exact equations defining the propagation of the covariance matrix are recursive in nature. If P_k^+ is the control covariance immediately after the k^{th} pulse, the covariance will propagate to the time of the next pulse t_{k+1} by the formula

$$P_{k+1}^- = \Phi_{k+1,k}\ P_k^+\ \Phi_{k+1,k}^T \qquad (2)$$

where $\Phi_{k+1,k}$ is the 6x6 state transition matrix relating perturbations at t_{k+1} to perturbations at t_k . Adding the pulse at t_{k+1} expands the covariance by

$$P^{+}_{k+1} = P^{-}_{k+1} + \begin{bmatrix} 0 & 0 \\ - & - \\ 0 & Q \end{bmatrix} \qquad (3)$$

where Q is set equal either to the nominal or final form of Q.

To start the process the control covariance following the first pulse is given by

$$P^{+}_{1} = \begin{bmatrix} 0 & 0 \\ - & - \\ 0 & Q \end{bmatrix} \qquad (4)$$

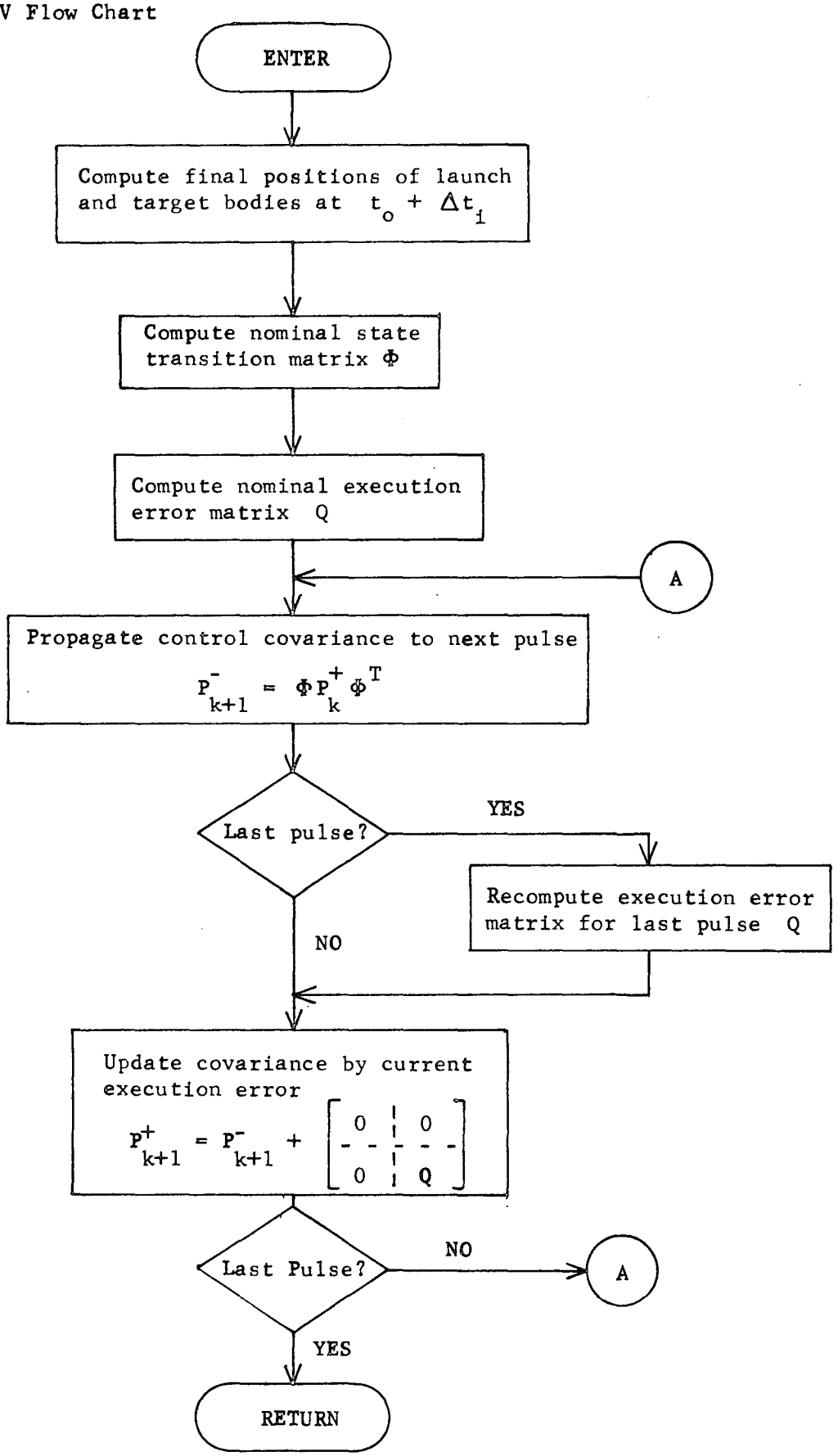
For efficiency one simplification is made in the process. Instead of recomputing the state transition matrix over each interval, the value of that matrix is held constant at the value corresponding to the "average interval". To explain this, let the state of the spacecraft at the time t_o of the impulsive $\vec{\Delta v}$ computation be denoted $\vec{r}_o$, $\vec{v}_o$. Then the "average interval" is defined to be the perturbed heliocentric trajectory (PERHEL) resulting from the propagation of the state $(\vec{r}_o, \vec{v}_o + \frac{1}{2}\vec{\Delta v})$ over the interval $(t_o, t_o + \Delta t_i)$.

The constant state transition matrix Φ is computed by numerical differencing. The initial state $(\vec{r}_o, \vec{v}_o + \frac{1}{2}\vec{\Delta v})$ is first propagated over the Δt_i time interval (using PERHEL) resulting in the state $(\vec{r}_f, \vec{v}_f)$. Then the x-component of initial position is perturbed by Δx, leading to a final state of $(\vec{r}_{Pf}, \vec{v}_{Pf})$ upon propagation. The first column of the matrix is then computed by

$$\Phi_1 = \left[\frac{\vec{r}_{Pf} - \vec{r}_f}{\Delta x} , \frac{\vec{v}_{Pf} - \vec{v}_f}{\Delta x} \right]^T \qquad (5)$$

The other columns of Φ are computed by similar computations using the remaining components of position and velocity $(y, z, \dot{x}, \dot{y}, \dot{z})$.

PULCØV Flow Chart



PULSEX Analysis

PULSEX is responsible for the actual execution of the pulsing arc. Experiments have shown that adding an impulsive $\vec{\Delta v}$ at time t_o may be approximated quite closely by centering an equivalent sequence of smaller impulses about the nominal time t_o.

This equivalent sequence of thrusts is computed by PREPUL. It consists of $N_p - 1$ pulses $\vec{\Delta v}_i$ and a final pulse $\vec{\Delta v}_f$ satisfying

$$(N_p - 1)\ \vec{\Delta v}_i + \vec{\Delta v}_f = \vec{\Delta v} \tag{1}$$

The pulses are separated by a time interval Δt_i. The duration of the entire sequence of pulses is given by $\Delta T = (N_p - 1)\ \Delta t_i$.

For efficiency the perturbed heliocentric conic propagator PERHEL is used to propagate the trajectory between pulses. PERHEL requires the positions of the launch and target bodies at the beginning and end of each propagation interval. PREPUL stores the position and velocity of the launch and target bodies at the reference time t_o : ($\vec{r}_{LO}$, $\vec{v}_{LO}$) and ($\vec{r}_{TO}$, $\vec{v}_{TO}$) and stores the constants of the f and g series for those states (f_{Lk}, g_{Lk}, f_{Tk}, g_{Tk}, k=1,6). The position of the launch body at some time t relative to the reference time t_o is then given by

$$\vec{r}_L(t) = f_L(t)\ \vec{r}_{LO} + g_L(t)\ \vec{v}_{LO} \tag{2}$$

where

$$f_L(t) = \sum_{k=0}^{6} f_{Lk}\ t^k \tag{3}$$

$$g_L(t) = \sum_{k=1}^{6} g_{Lk}\ t^k$$

with similar equations holding for the target body.

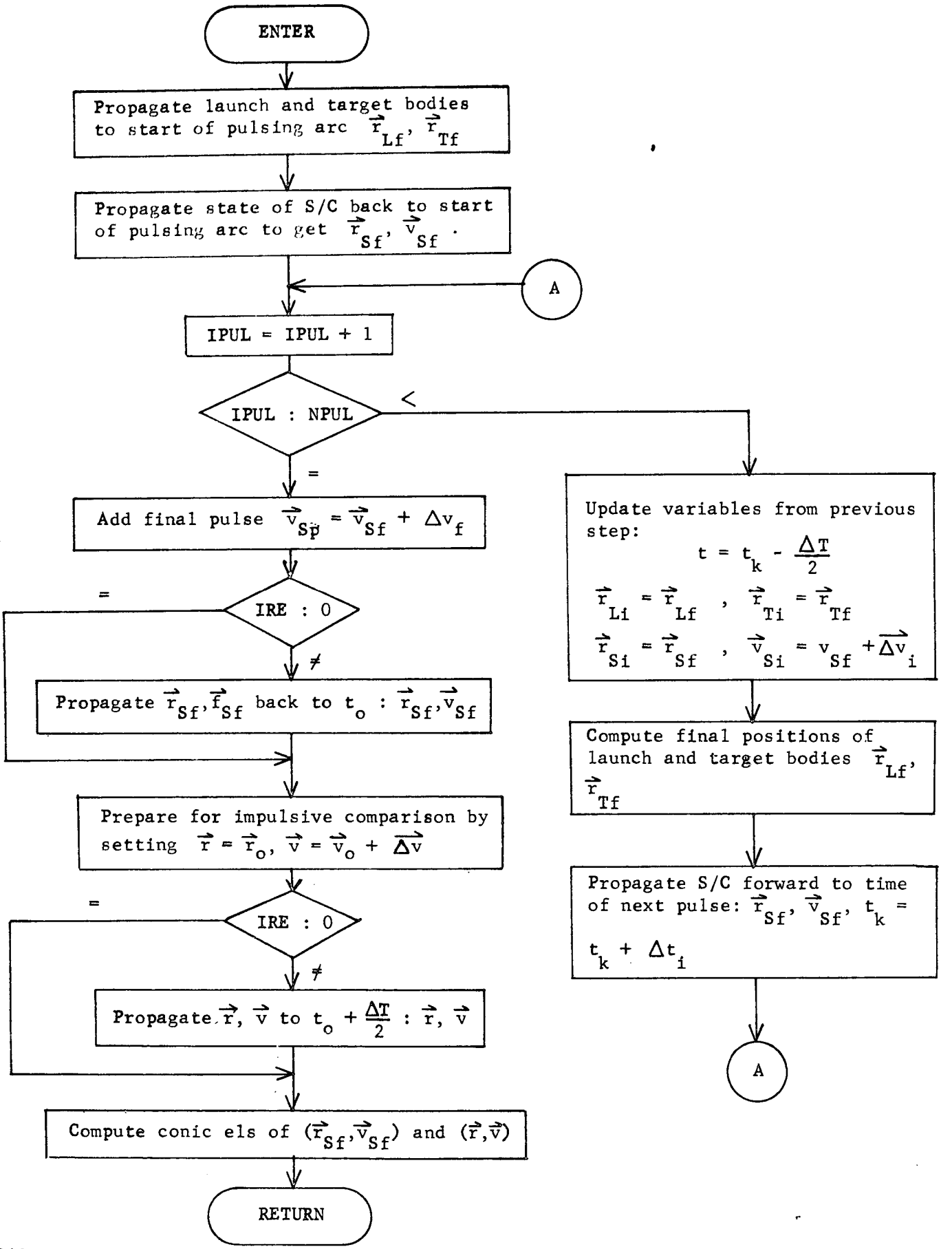
The procedure of PULSEX is straightforward. The positions of the launch and target bodies are computed at the time the pulsing arc should begin: $t_B = t_o - \Delta T/2$. PERHEL is then called to propagate the spacecraft from t_o backwards to t_B. The actual pulsing arc cycle is now entered. The nominal velocity increment $\vec{\Delta v}_i$ is added to the current velocity impulsively

$$\vec{v} = \vec{v} + \vec{\Delta v}_i \tag{4}$$

and the resulting state $(\vec{r}, \vec{v})$ is propagated forward over the time interval Δt_i by PERHEL. Another pulse is added and the process repeated until $N_p - 1$ pulses have been added. Finally a pulse of $\vec{\Delta v}_f$ is added.

Two options are now permitted. If IRE = 0 , the final state is not altered (NOMNAL). If IRE = 1, the final state is propagated backwards back to t_o for use in ERRAN and SIMUL.

Finally CAREL is called to compute the conic elements of the final state. For comparison purposes, the impulsive $\vec{\Delta v}$ is added to the state at t_o, propagated to the final time $t_E = t_o + \Delta T/2$ by PERHEL, and those elements computed.


ENTER
Propagate launch and target bodies to start of pulsing arc $\vec{r}_{Lf}$, $\vec{r}_{Tf}$
Propagate state of S/C back to start of pulsing arc to get $\vec{r}_{Sf}$, $\vec{v}_{Sf}$.
A
IPUL = IPUL + 1
IPUL : NPUL
<
=
Add final pulse $\vec{v}_{Sp} = \vec{v}_{Sf} + \Delta v_f$
=
IRE : 0
≠
Propagate $\vec{r}_{Sf}, \vec{f}_{Sf}$ back to t_o : $\vec{r}_{Sf}, \vec{v}_{Sf}$
Prepare for impulsive comparison by setting $\vec{r} = \vec{r}_o$, $\vec{v} = \vec{v}_o + \vec{\Delta v}$
=
IRE : 0
≠
Propagate $\vec{r}$, $\vec{v}$ to $t_o + \frac{\Delta T}{2}$: $\vec{r}$, $\vec{v}$
Compute conic els of $(\vec{r}_{Sf}, \vec{v}_{Sf})$ and $(\vec{r}, \vec{v})$
RETURN
Update variables from previous step:
$t = t_k - \frac{\Delta T}{2}$
$\vec{r}_{Li} = \vec{r}_{Lf}$, $\vec{r}_{Ti} = \vec{r}_{Tf}$
$\vec{r}_{Si} = \vec{r}_{Sf}$, $\vec{v}_{Si} = v_{Sf} + \vec{\Delta v}_i$
Compute final positions of launch and target bodies $\vec{r}_{Lf}$, $\vec{r}_{Tf}$
Propagate S/C forward to time of next pulse: $\vec{r}_{Sf}$, $\vec{v}_{Sf}$, $t_k = t_k + \Delta t_i$
A

QCØMP Analysis

Subroutine QCØMP computes the execution error covariance matrix $\tilde{Q}_j$ for a velocity correction $\vec{\Delta V} = (\Delta V_x, \Delta V_y, \Delta V_z)$ occurring at time t_j. If the execution error is assumed to have form

$$\vec{\delta \Delta V} = k \vec{\Delta V} + s \frac{\vec{\Delta V}}{\Delta V} + \vec{\delta \Delta V}_{pointing}$$

where k is the proportionality error and s is the resolution error, then the elements of the $\tilde{Q}_j$ matrix are given by

$$\tilde{Q}_{11} = \Delta V_x^2 \left[\sigma_k^2 + \frac{\sigma_s^2}{\rho^2} \right] + \frac{\Delta V_y^2 \rho^2 \sigma_{\delta\alpha}^2}{\mu^2} + \frac{\Delta V_x^2 \Delta V_z^2 \sigma_{\delta\beta}^2}{\mu^2}$$

$$\tilde{Q}_{12} = \tilde{Q}_{21} = \Delta V_x \Delta V_y \left[\sigma_k^2 + \frac{\sigma_s^2}{\rho^2} - \frac{\rho^2 \sigma_{\delta\alpha}^2}{\mu^2} + \frac{\Delta V_z^2 \sigma_{\delta\beta}^2}{\mu^2} \right]$$

$$\tilde{Q}_{13} = \tilde{Q}_{31} = \Delta V_x \Delta V_z \left[\sigma_k^2 + \frac{\sigma_s^2}{\rho^2} - \sigma_{\delta\beta}^2 \right]$$

$$\tilde{Q}_{22} = \Delta V_y^2 \left[\sigma_k^2 + \frac{\sigma_s^2}{\rho^2} \right] + \frac{\Delta V_x^2 \rho^2 \sigma_{\delta\alpha}^2}{\mu^2} + \frac{\Delta V_y^2 \Delta V_z^2 \sigma_{\delta\beta}^2}{\mu^2}$$

$$\tilde{Q}_{23} = \tilde{Q}_{32} = \Delta V_y \Delta V_z \left[\sigma_k^2 + \frac{\sigma_s^2}{\rho^2} - \sigma_{\delta\beta}^2 \right]$$

$$Q_{33} = \Delta V_z^2 \left[\sigma_k^2 + \frac{\sigma_s^2}{\rho^2} \right] + \mu^2 \sigma_{\delta\beta}^2$$

where $\mu^2 = \Delta V_x^2 + \Delta V_y^2$, $\rho^2 = \mu^2 + \Delta V_z^2$, and σ_s^2, σ_k^2, $\sigma_{\delta\alpha}^2$, and $\sigma_{\delta\alpha}^2$ are the variances associated with the resolution, proportionality, and two pointing errors, respectively.

QUASI Analysis

At a quasi-linear filtering event the most recent nominal trajectory is updated by using the most recent state deviation estimate. If $\tilde{X}_j^-$ is the most recent nominal position/velocity state immediately preceding the event at time t_j, and if $\delta\hat{\tilde{X}}_j^-$ is the position/velocity deviation estimate, then immediately following the quasi-linear filtering event, the most recent nominal position/velocity state is given by

$$\tilde{X}_j^+ = \tilde{X}_j^- + \delta\hat{\tilde{X}}_j^-$$

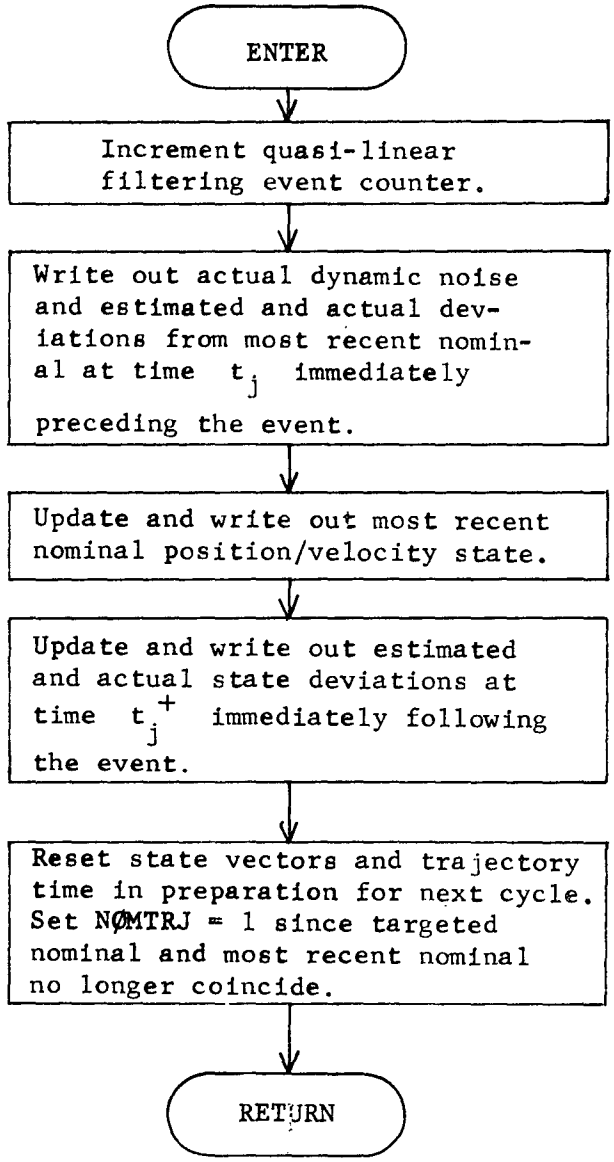
The estimated and actual deviations from the most recent nominal trajectory must also be updated:

$$\delta\hat{\tilde{X}}_j^+ = 0$$

$$\delta\tilde{X}_j^+ = \delta\tilde{X}_j^- - \delta\hat{\tilde{X}}_j^-$$

A quasi-linear filtering event in no way alters the knowledge and control uncertainties at time t_j. Thus knowledge covariance P_{k_j} and control covariance P_{c_j} remain constant across a quasi-linear filtering event. Furthermore, since no velocity correction is performed, the (most recent) targeted nominal $\bar{X}_j$ is unchanged. Neither is the solve-for parameter state updated at a quasi-linear filtering event.

QUASI Flow Chart

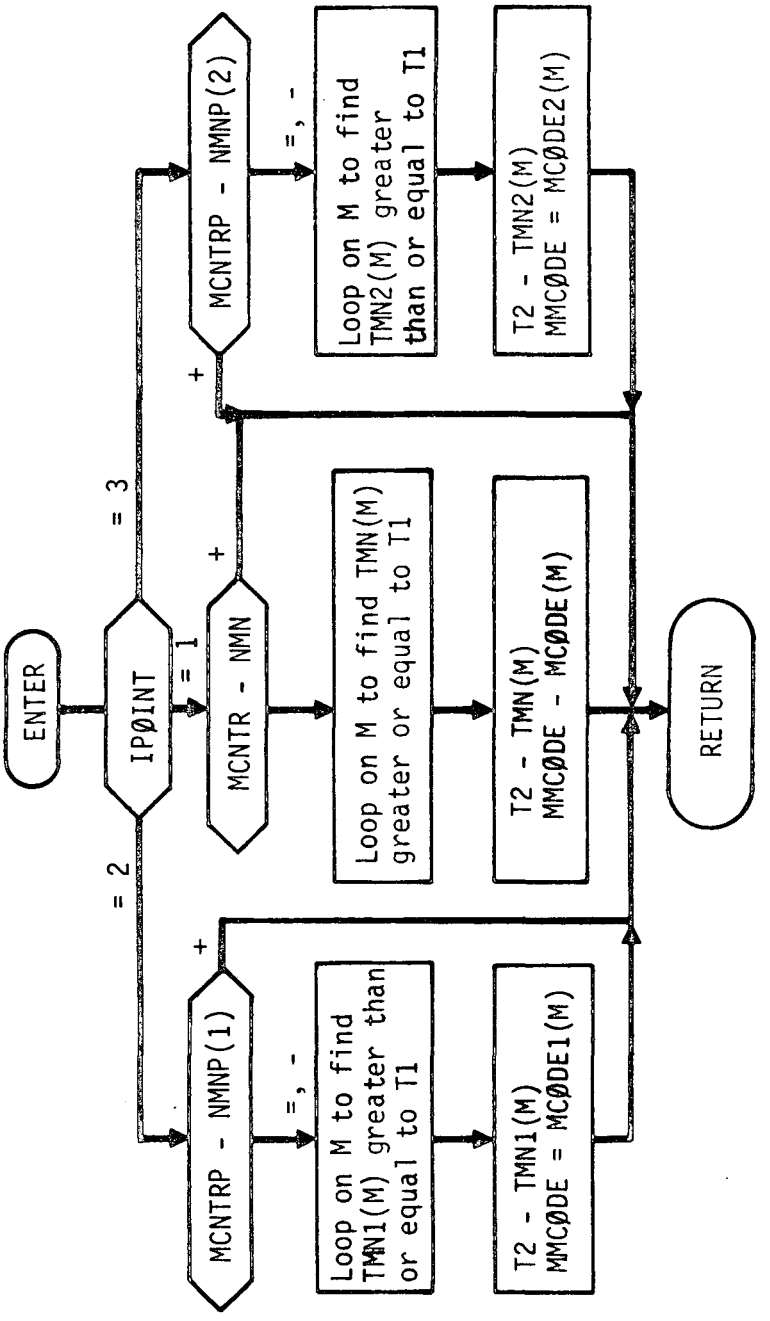


RNUM Analysis

Function subprogram RNUM supplies random numbers on a normal distribution with near zero and standard deviation σ .

Twelve random numbers X_i between 0 and 1 are computed, which are then used to compute the returned random number RNUM using the following equation:

$$\text{RNUM} = \left[\sum_{i=1}^{12} X_i - 6 \right] \cdot \sigma$$


ENTER
IPØINT
= 1
= 2
= 3
MCNTR - NMN
+
Loop on M to find TMN(M) greater or equal to T1
T2 - TMN(M)
MMCØDE - MCØDE(M)
MCNTRP - NMNP(1)
=, -
Loop on M to find TMN1(M) greater than or equal to T1
T2 - TMN1(M)
MMCØDE = MCØDE1(M)
MCNTRP - NMNP(2)
Loop on M to find TMN2(M) greater than or equal to T1
T2 - TMN2(M)
MMCØDE = MCØDE2(M)
RETURN

SERIE Analysis

SERIE computes the transcendental functions S(x) and C(x) used in the FLITE program in the solution of Lambert's theorem.

The functions S(x) and C(x) are defined by

$$S(x) = \frac{\sqrt{x} - \sin\sqrt{x}}{\sqrt{x}^3} \qquad x > 0$$

$$= \frac{\sinh\sqrt{-x} - \sqrt{-x}}{\sqrt{-x}^3} \qquad x < 0$$

$$= \frac{1}{6} \qquad x = 0 \qquad (1)$$

$$C(x) = \frac{1 - \cos\sqrt{x}}{x} \qquad x > 0$$

$$= \frac{\cosh\sqrt{-x} - 1}{-x} \qquad x < 0$$

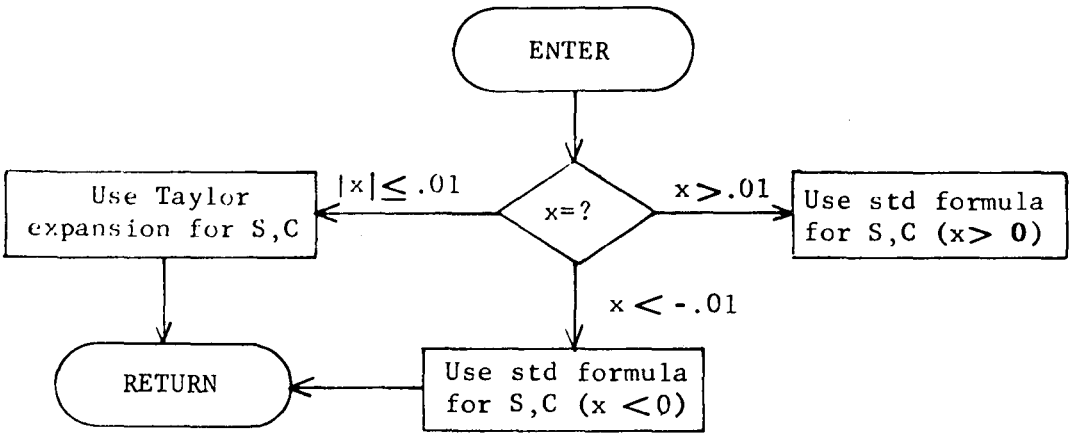
$$= \frac{1}{2} \qquad x = 0 \qquad (2)$$

For small values of $|x|$ the Taylor series expansions are used

$$S(x) = \frac{1}{3!} - \frac{x}{4!} + \frac{x^2}{5!} + \ldots$$

$$C(x) = \frac{1}{2!} - \frac{x}{3!} + \frac{x^2}{4!} + \ldots \qquad (3)$$

SERIE Flow Chart



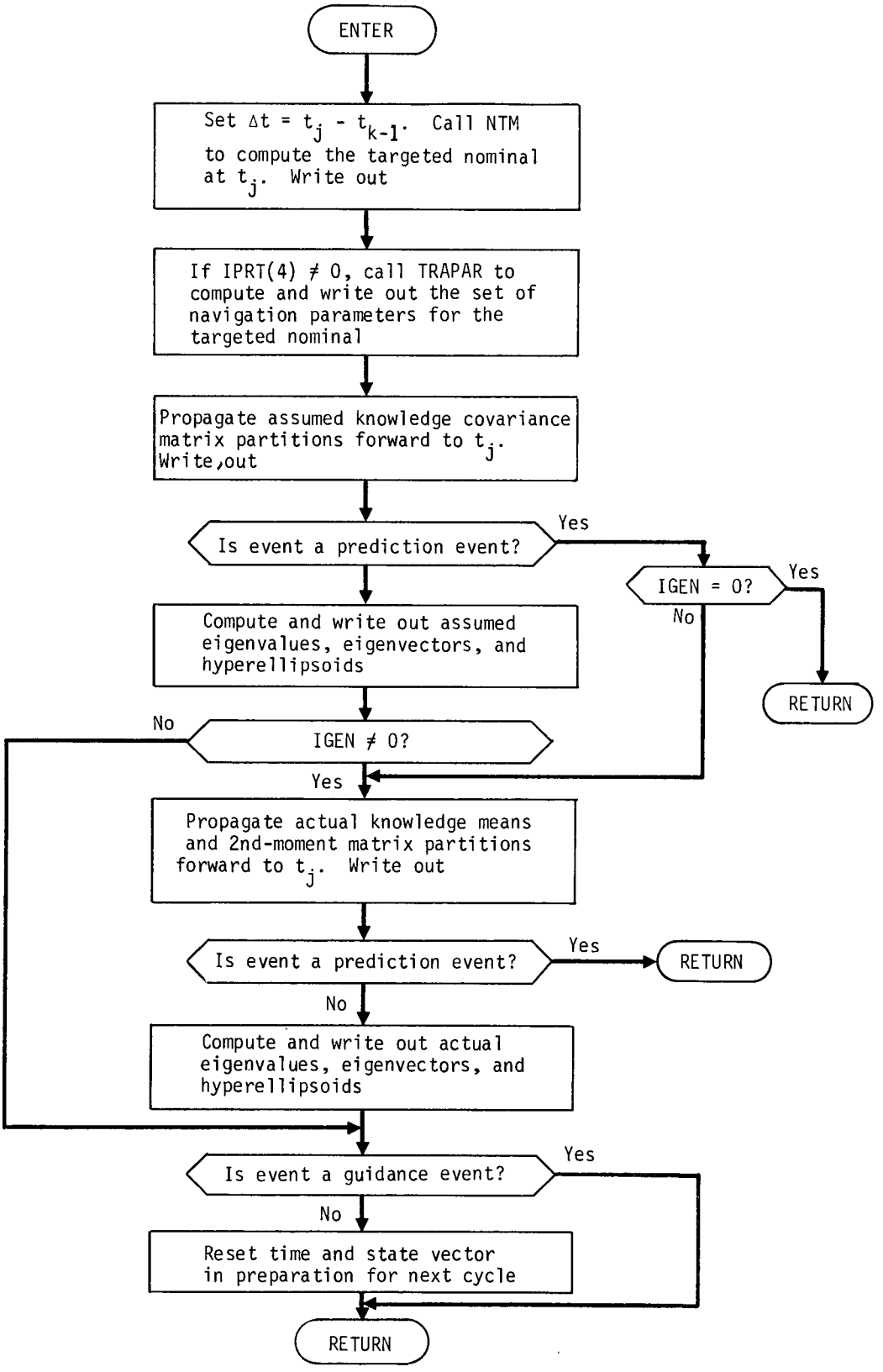
4-5

SETEVN Analysis

Before executing any event in the error analysis/generalized covariance analysis program subroutine SETEVN is called to perform a series of computations that are common to all events. Subroutine SETEVN computes the targeted nominal trajectory at t_j and propagates the assumed and actual knowledge covariance partitions at t_{k-1} -- the time of the previous event or measurement -- forward to time t_j using the propagation equations found in subroutine GNAVM. The actual estimation error means are also propagated forward to t_j using the propagation equations found in subroutine MEAN.

For any event other than a prediction event, subroutine SETEVN also computes eigenvalues, eigenvectors, and hyperellipsoids of the position and velocity partitions of the assumed and actual knowledge covariance at t_j.

SETEVN Flow Chart



SETEVS Analysis

Prior to executing any event in the simulation mode, subroutine SETEVS is called to perform a series of computations which are common to all events. After computing the targeted nominal and most recent nominal states at the time of the event t_j, knowledge covariance partitions are propagated forward to time t_j from time t_{k-1} of the previous event or measurement using the prediction equations found in the NAVM Analysis section. The actual trajectory state at t_j is computed using

$$X_j = Z_j + \omega_j$$

where Z_j is the actual trajectory state assuming no unmodeled acceleration has been acting on the spacecraft, and ω_j is the contribution of the actual unmodeled acceleration to the actual trajectory state at t_j. The actual and predicted position/velocity deviations from the most recent nominal at t_j are given by

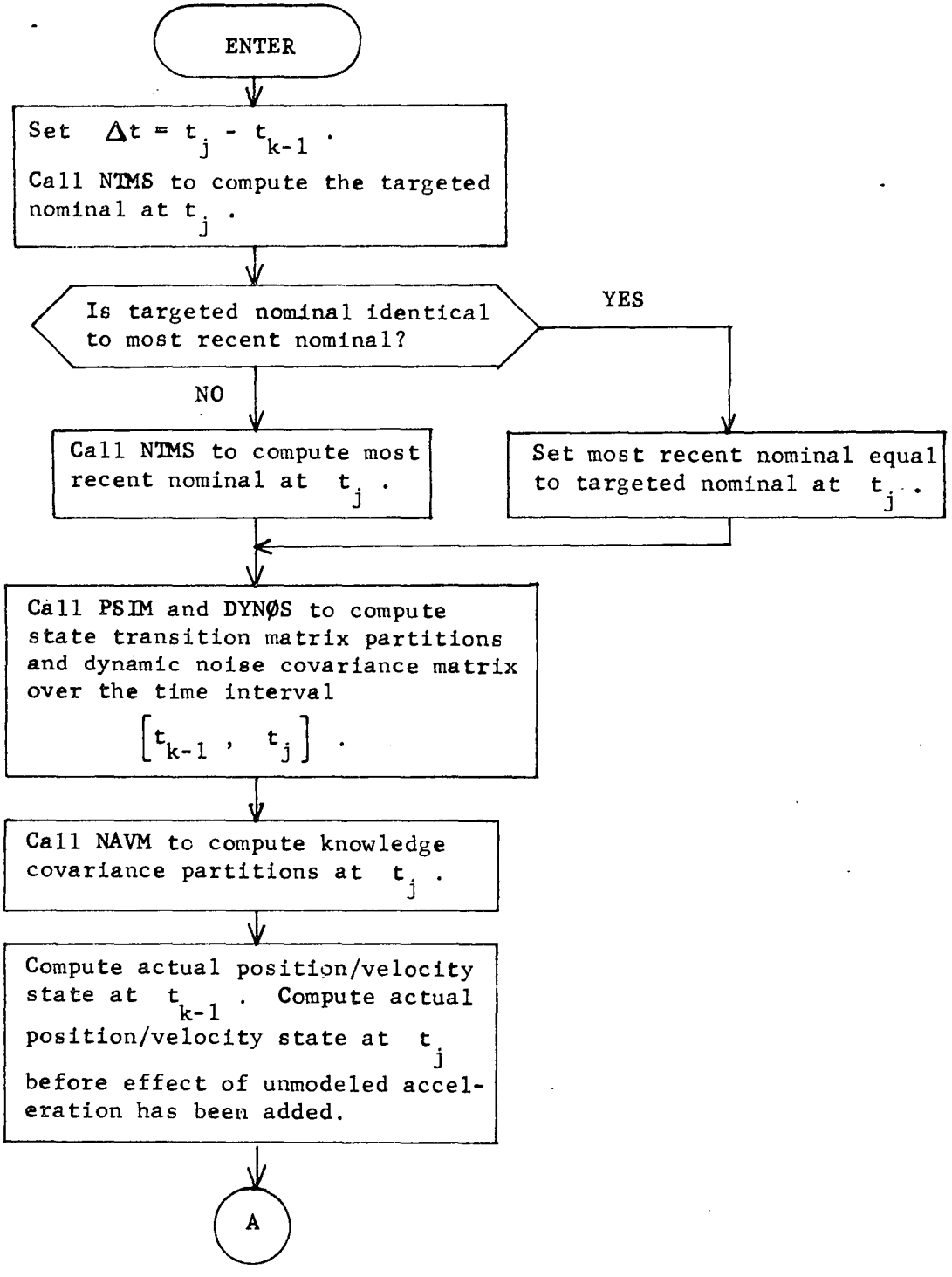
$$\delta\tilde{X}_j = X_j - \tilde{X}_j$$

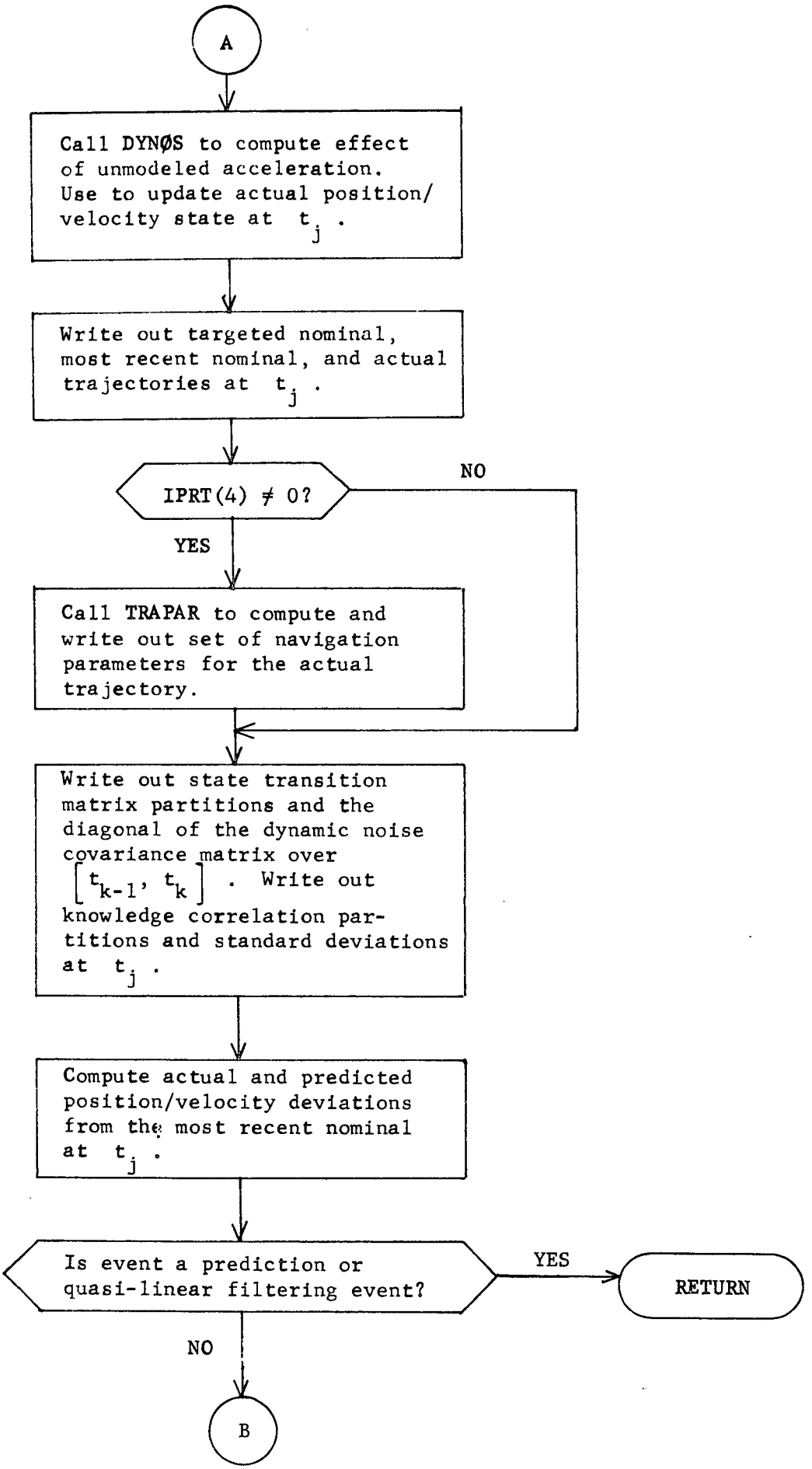
and $$\delta\tilde{\hat{X}}_j = \Phi(t_j, t_{k-1})\, \delta\tilde{\hat{X}}_{k-1} + \theta_{xx_s}(t_j, t_{k-1})\, \delta\tilde{\hat{X}}_{s_j} ,$$

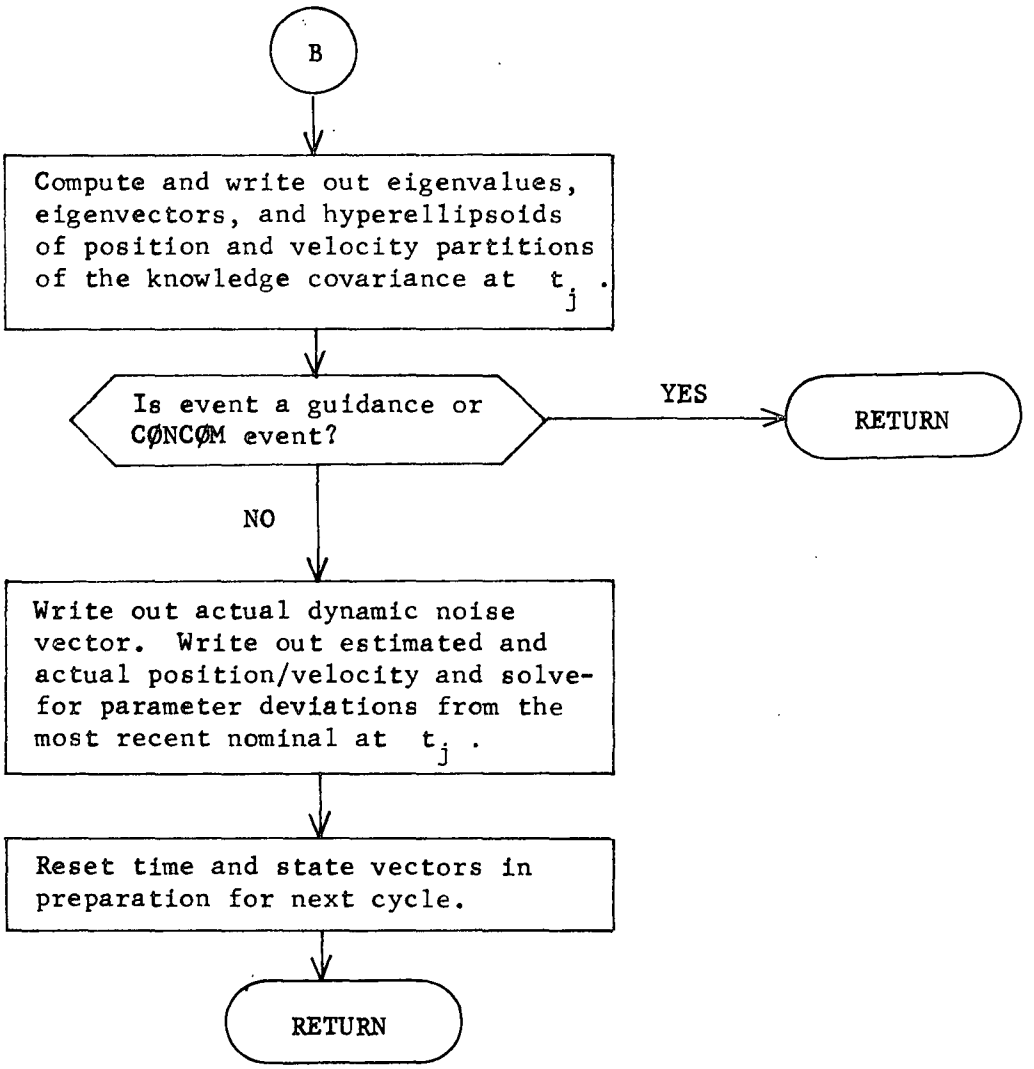
respectively, where Φ and θ_{xx_s} are the state transition matrix partitions over $\left[t_{k-1}, t_j \right]$.

For any event other than prediction and quasi-linear filtering events, subroutines SETEVS also computes eigenvalues, eigenvectors, and hyperellipsoids of the position and velocity partitions of the knowledge covariance at t_j.

SETEVS Flow Chart




A
Call DYNØS to compute effect of unmodeled acceleration. Use to update actual position/velocity state at t_j .
Write out targeted nominal, most recent nominal, and actual trajectories at t_j .
IPRT(4) ≠ 0?
NO
YES
Call TRAPAR to compute and write out set of navigation parameters for the actual trajectory.
Write out state transition matrix partitions and the diagonal of the dynamic noise covariance matrix over $[t_{k-1}, t_k]$. Write out knowledge correlation partitions and standard deviations at t_j .
Compute actual and predicted position/velocity deviations from the most recent nominal at t_j .
Is event a prediction or quasi-linear filtering event?
YES
RETURN
NO
B


B
Compute and write out eigenvalues, eigenvectors, and hyperellipsoids of position and velocity partitions of the knowledge covariance at t_j .
Is event a guidance or CØNCØM event?
YES
RETURN
NO
Write out actual dynamic noise vector. Write out estimated and actual position/velocity and solve-for parameter deviations from the most recent nominal at t_j .
Reset time and state vectors in preparation for next cycle.
RETURN

SIMUL Analysis

The primary function of subroutine SIMUL is to control the computational flow through the basic cycle (measurement processing) and all events in the simulation mode. Subroutine SIMUL also performs some computations in the basic cycle. All event-related analysis is presented in the event subroutines themselves and will not be treated below.

In the basic cycle the first task of SIMUL is to control the generation of targeted nominal and most recent nominal spacecraft states, $\bar{X}_{k+1}$ and $\tilde{X}_{k+1}$, respectively, at time t_k, given states $\bar{X}_k$ and $\tilde{X}_k$ at time t_k. Then, calling PSIM, DYNØS, TRAKS, and MENØS, successively, SIMUL controls the computation of all matrix information required by subroutine NAVM in order to compute the covariance matrix partitions at time t_{k+1}^+ immediately following the measurement.

After computing the actual state X_k at time t_k from

$$X_k = \tilde{X}_k + \delta\tilde{X}_k \tag{1}$$

where $\delta\tilde{X}_k$ is the actual spacecraft state deviation from the most recent nominal, SIMUL controls the generation of the actual state Z_{k+1} at time t_k before the effect of unmodeled acceleration has been added. Then, having called DYNØS to compute the effect of unmodeled acceleration ω_{k+1}, SIMUL computes the actual state and actual state deviation at time t_{k+1}:

$$X_{k+1} = Z_{k+1} + \omega_{k+1} \tag{2}$$

$$\delta\tilde{X}_{k+1} = X_{k+1} - \tilde{X}_{k+1} \tag{3}$$

With both the most recent nominal and actual spacecraft states available at t_{k+1}, SIMUL calls TRAKS twice in succession to compute the ideal measurements $\tilde{Y}_{k+1}$ and $\underline{Y}_{k+1}$, respectively, which would be made at each of these trajectory states. Calling MENØS, RNUM, and BIAS to compute the actual measurement noise and bias corrupting the ideal measurement associated with the actual state, SIMUL computes the actual measurement at time t_{k+1} using

$$Y_{k+1}^a = \underline{Y}_{k+1} + b_{k+1} + \nu_{k+1} \tag{4}$$

where b_{k+1} and ν_{k+1} represent the actual measurement bias and noise, respectively.

All information required for computing both predicted and filtered state deviations from the most recent nominal at t_{k+1} is now available. With Φ and θ_{xx_s} denoting state transition matrix partitions over the time interval $\left[t_k, t_{k+1} \right]$, SIMUL computes the predicted spacecraft state deviations and solve-for parameter deviations at t_{k+1} using

$$\delta\tilde{\hat{X}}^{-}_{k+1} = \Phi\, \delta\tilde{\hat{X}}^{+}_{k} + \theta_{xx_s}\, \delta\tilde{\hat{X}}^{+}_{s_k} \tag{5}$$

$$\delta\tilde{\hat{X}}^{-}_{s_{k+1}} = \delta\tilde{\hat{X}}^{+}_{s_k} \tag{6}$$

Prior to computing filtered deviations, SIMUL computes the measurement residual from

$$\epsilon_{k+1} = \left(Y^{a}_{k+1} - \tilde{Y}_{k+1} \right) - H_{k+1}\, \delta\tilde{\hat{X}}^{-}_{k+1} - M_{k+1}\, \delta\tilde{\hat{X}}^{-}_{s_{k+1}} \tag{7}$$

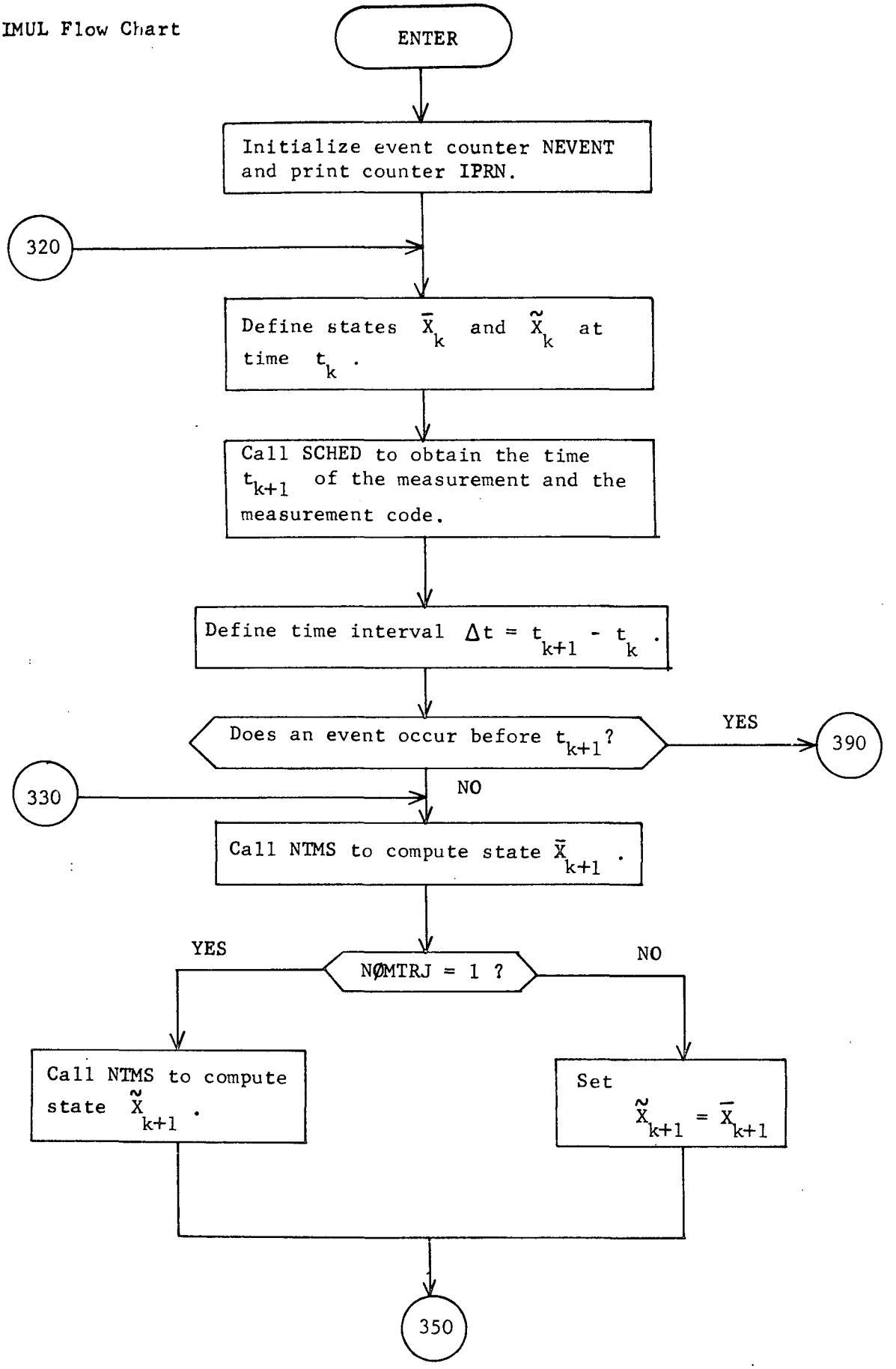
where H_{k+1} and M_{k+1} are observation matrix partitions. Filtered spacecraft state deviations and solve-for parameter deviations are then computed from

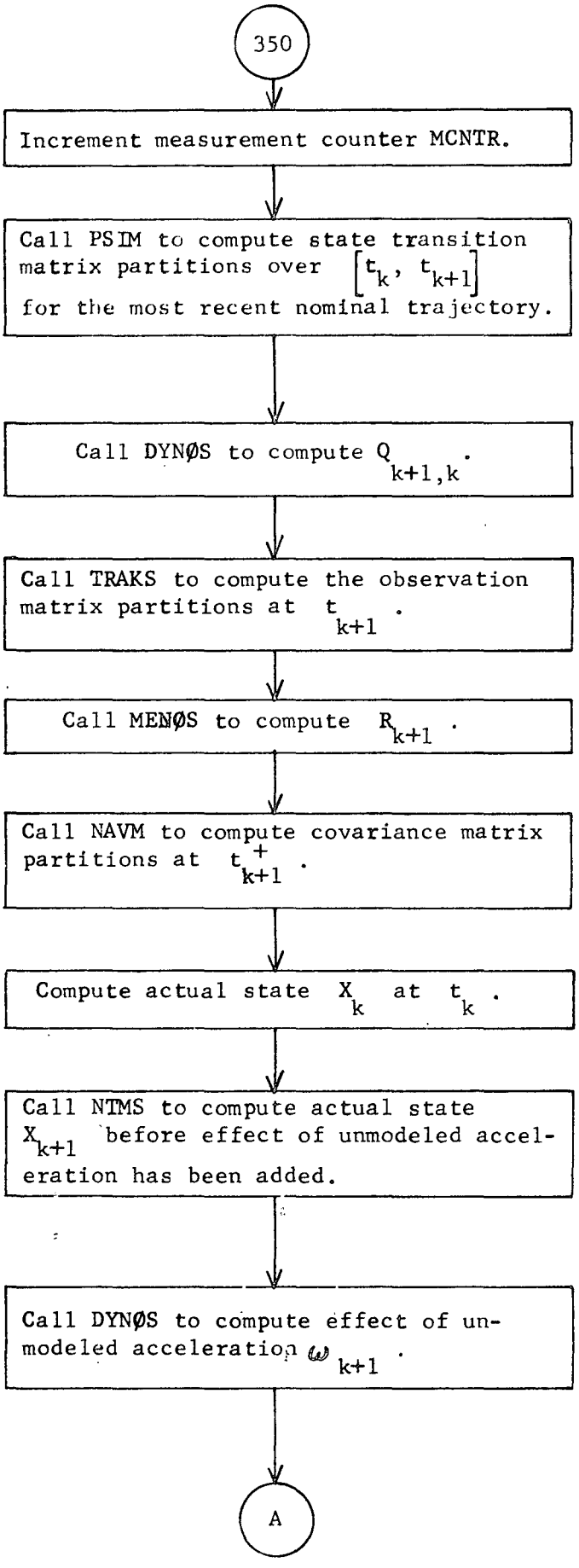
$$\delta\tilde{\hat{X}}^{+}_{k+1} = \delta\tilde{\hat{X}}^{-}_{k+1} + K_{k+1}\, \epsilon_{k+1} \tag{8}$$

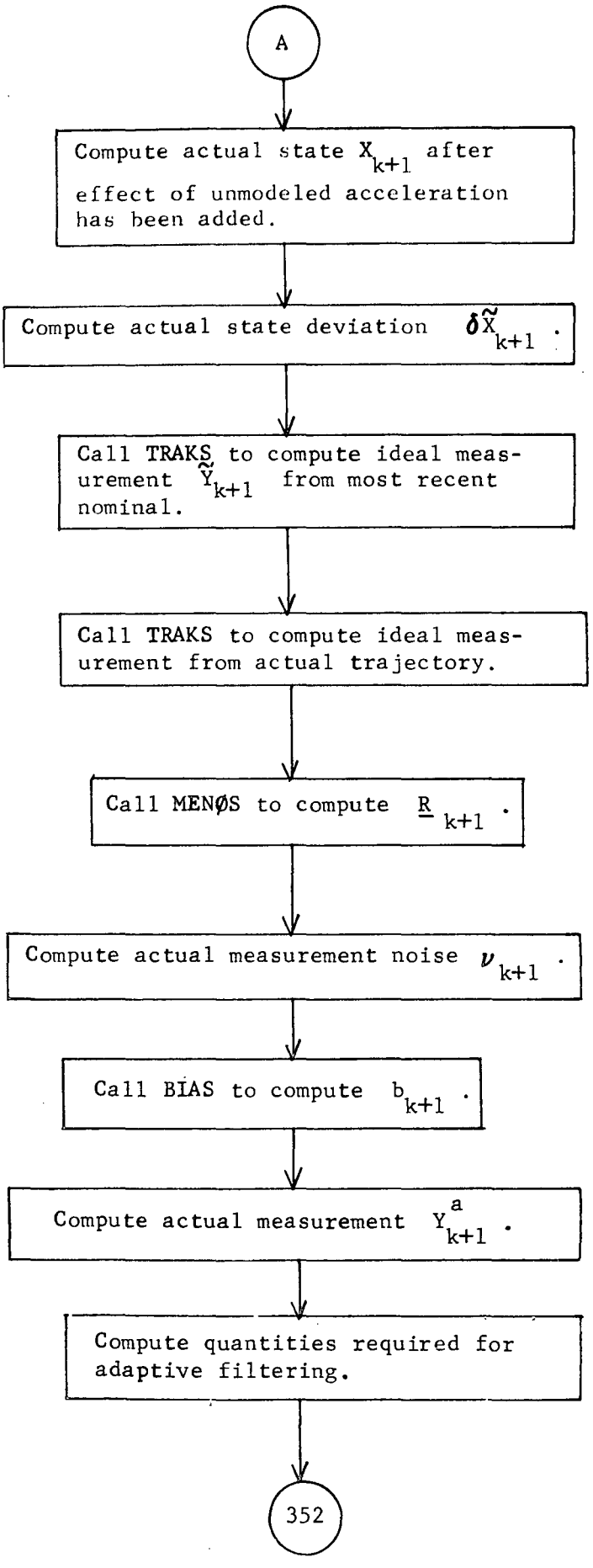
$$\delta\tilde{\hat{X}}^{+}_{s_{k+1}} = \delta\tilde{\hat{X}}^{-}_{s_{k+1}} + S_{k+1}\, \epsilon_{k+1} \tag{9}$$

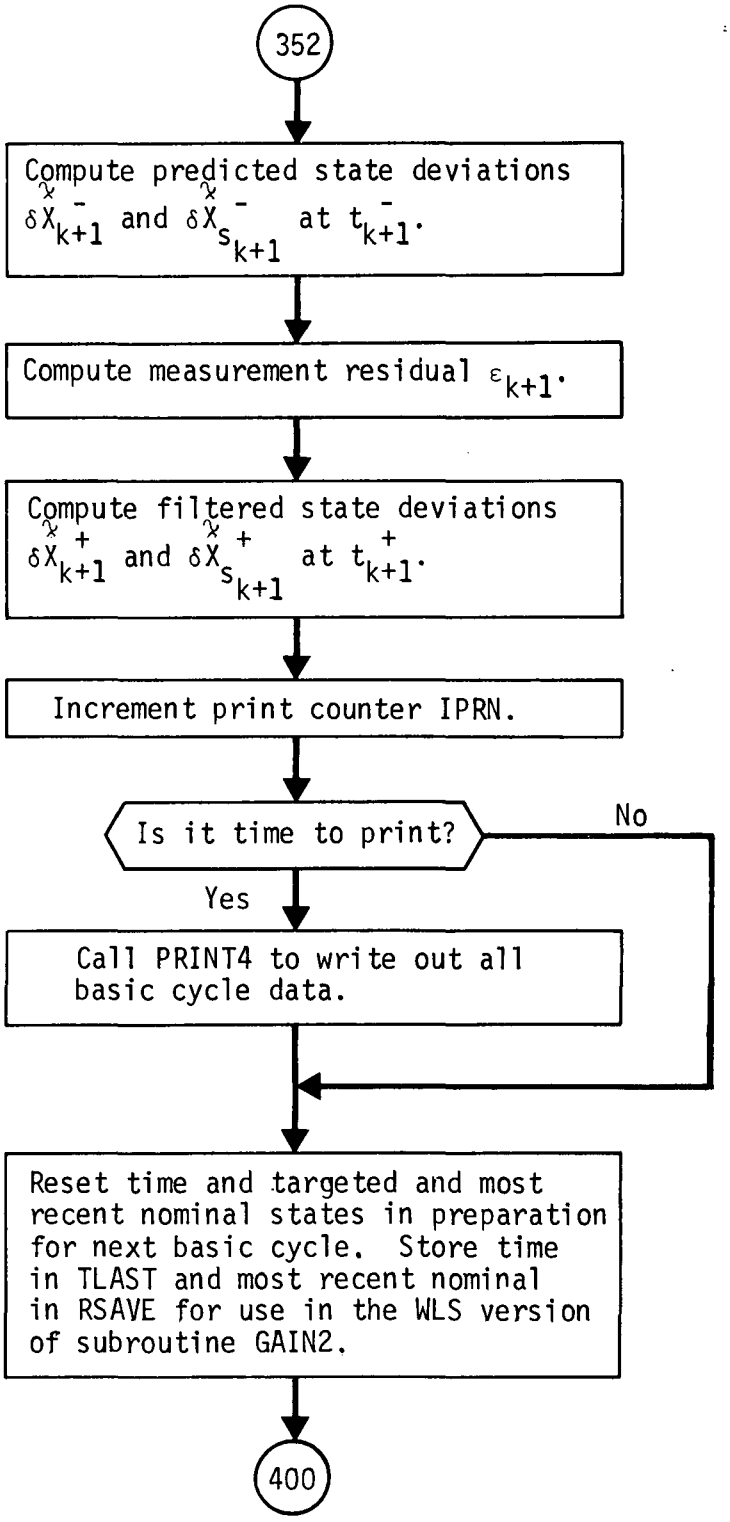
where K_{k+1} and S_{k+1} are the filter gain constants.

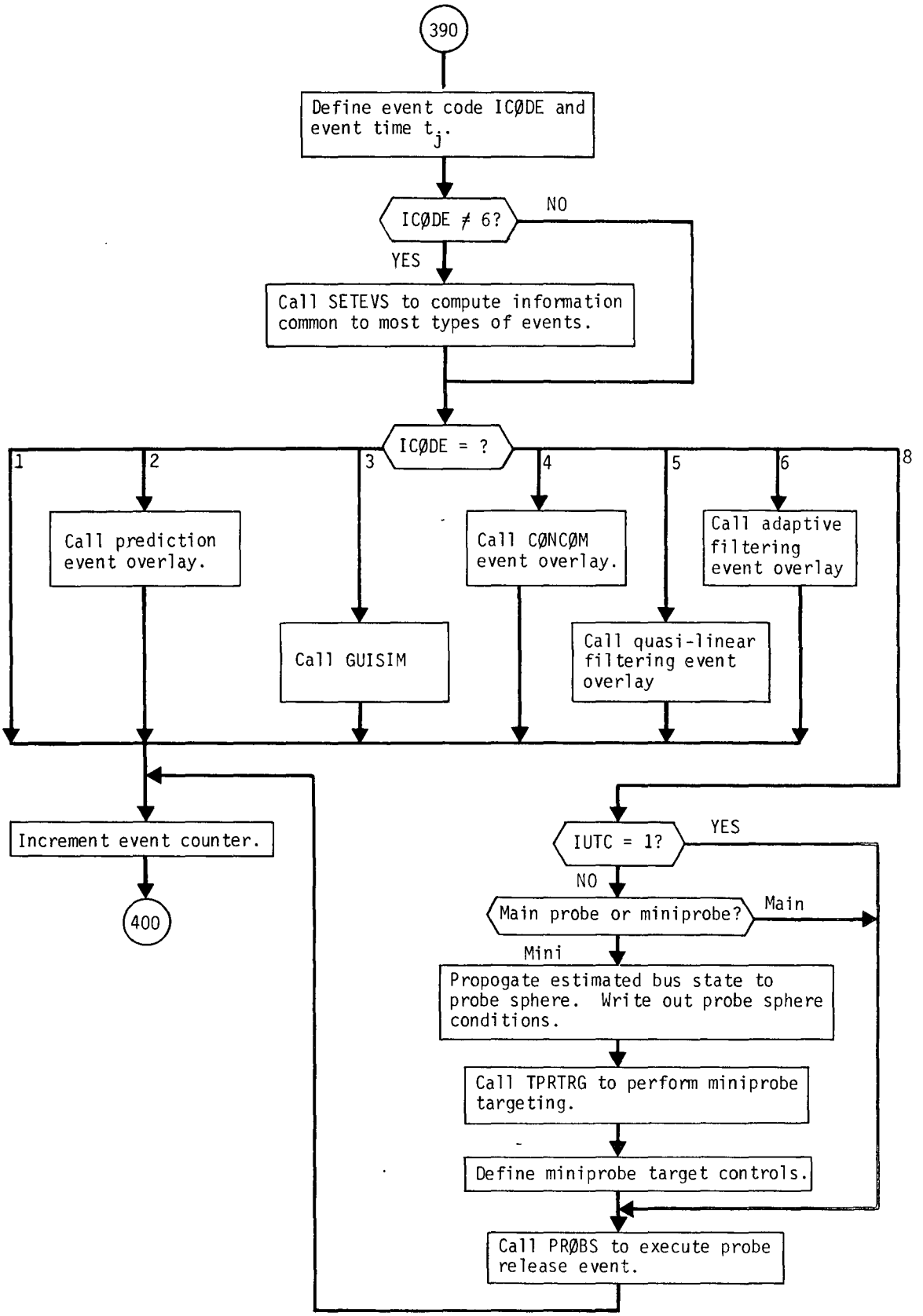
SIMUL Flow Chart

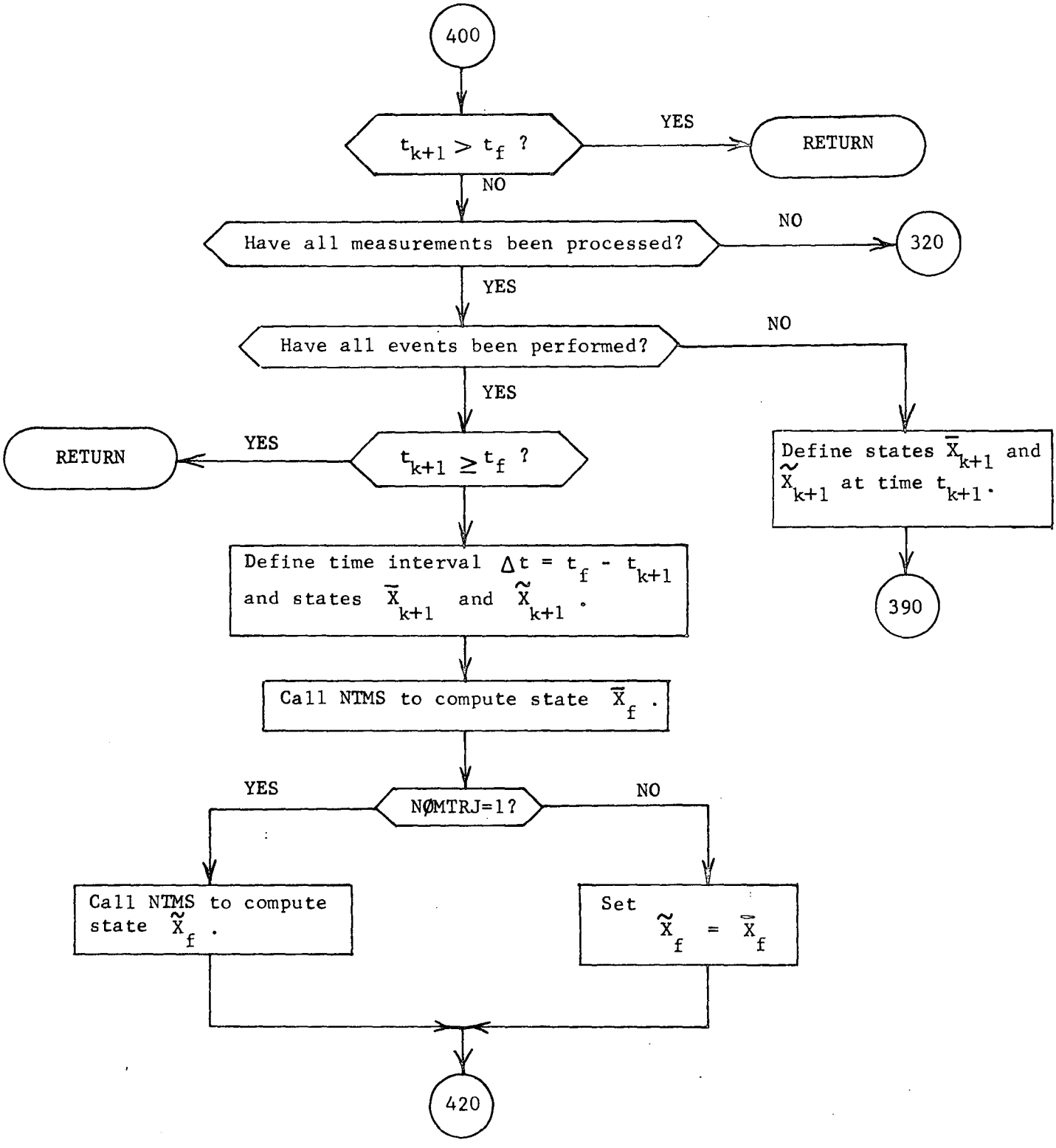


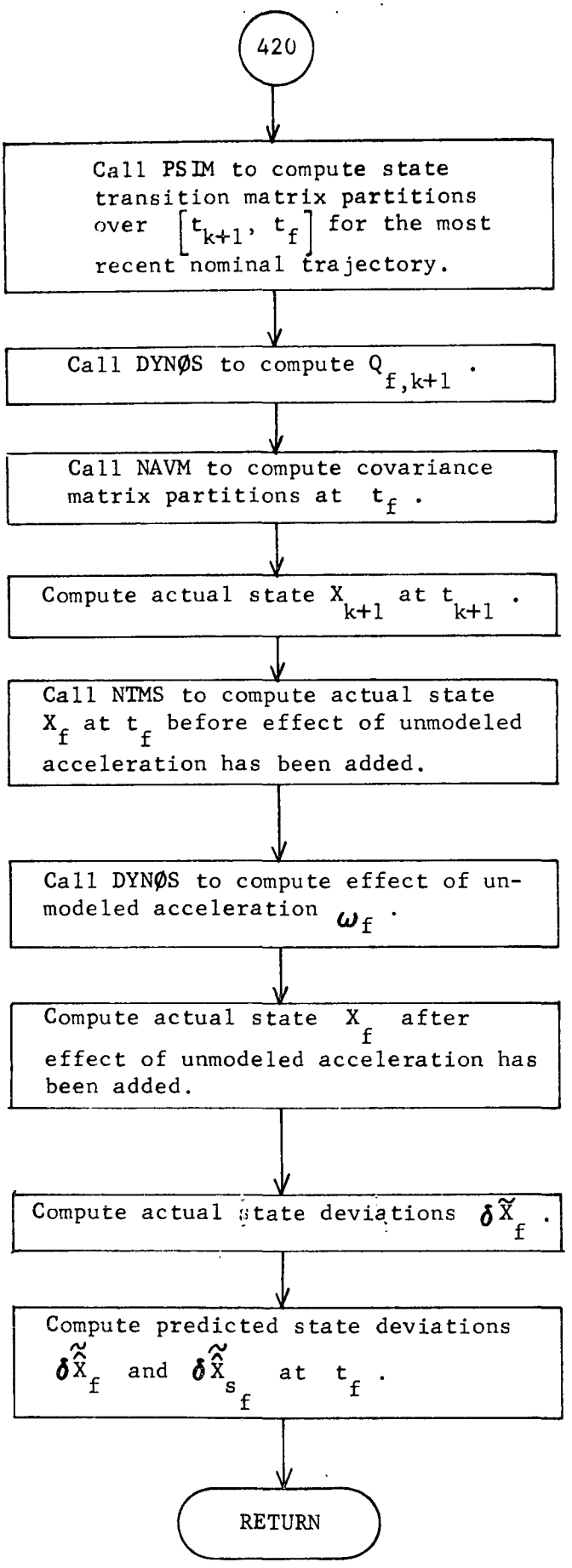

350
Increment measurement counter MCNTR.
Call PSIM to compute state transition matrix partitions over $[t_k, t_{k+1}]$ for the most recent nominal trajectory.
Call DYNØS to compute $Q_{k+1,k}$.
Call TRAKS to compute the observation matrix partitions at t_{k+1}.
Call MENØS to compute R_{k+1}.
Call NAVM to compute covariance matrix partitions at t^+_{k+1}.
Compute actual state X_k at t_k.
Call NTMS to compute actual state X_{k+1} before effect of unmodeled acceleration has been added.
Call DYNØS to compute effect of unmodeled acceleration ω_{k+1}.
A


A
Compute actual state X_{k+1} after effect of unmodeled acceleration has been added.
Compute actual state deviation $\delta\tilde{X}_{k+1}$.
Call TRAKS to compute ideal measurement $\tilde{Y}_{k+1}$ from most recent nominal.
Call TRAKS to compute ideal measurement from actual trajectory.
Call MENØS to compute $\underline{R}_{k+1}$.
Compute actual measurement noise ν_{k+1}.
Call BIAS to compute b_{k+1}.
Compute actual measurement Y^{a}_{k+1}.
Compute quantities required for adaptive filtering.
352


352
Compute predicted state deviations $\delta\hat{X}_{k+1}^{-}$ and $\delta\hat{X}_{s_{k+1}}^{-}$ at t_{k+1}^{-}.
Compute measurement residual ε_{k+1}.
Compute filtered state deviations $\delta\hat{X}_{k+1}^{+}$ and $\delta\hat{X}_{s_{k+1}}^{+}$ at t_{k+1}^{+}.
Increment print counter IPRN.
Is it time to print?
No
Yes
Call PRINT4 to write out all basic cycle data.
Reset time and targeted and most recent nominal states in preparation for next basic cycle. Store time in TLAST and most recent nominal in RSAVE for use in the WLS version of subroutine GAIN2.
400


390
Define event code ICØDE and event time t_j.
ICØDE ≠ 6?
NO
YES
Call SETEVS to compute information common to most types of events.
ICØDE = ?
1
2
3
4
5
6
8
Call prediction event overlay.
Call GUISIM
Call CØNCØM event overlay.
Call quasi-linear filtering event overlay
Call adaptive filtering event overlay
Increment event counter.
400
IUTC = 1?
YES
NO
Main probe or miniprobe?
Main
Mini
Propogate estimated bus state to probe sphere. Write out probe sphere conditions.
Call TPRTRG to perform miniprobe targeting.
Define miniprobe target controls.
Call PRØBS to execute probe release event.


400
$t_{k+1} > t_f$?
YES
RETURN
NO
Have all measurements been processed?
NO
320
YES
Have all events been performed?
NO
YES
RETURN
YES
$t_{k+1} \geq t_f$?
Define states $\bar{X}_{k+1}$ and $\tilde{X}_{k+1}$ at time t_{k+1}.
390
Define time interval $\Delta t = t_f - t_{k+1}$ and states $\bar{X}_{k+1}$ and $\tilde{X}_{k+1}$.
Call NTMS to compute state $\bar{X}_f$.
YES
NØMTRJ=1?
NO
Call NTMS to compute state $\tilde{X}_f$.
Set $\tilde{X}_f = \bar{X}_f$
420


420
Call PSIM to compute state transition matrix partitions over $[t_{k+1}, t_f]$ for the most recent nominal trajectory.
Call DYNØS to compute $Q_{f,k+1}$.
Call NAVM to compute covariance matrix partitions at t_f.
Compute actual state X_{k+1} at t_{k+1}.
Call NTMS to compute actual state X_f at t_f before effect of unmodeled acceleration has been added.
Call DYNØS to compute effect of unmodeled acceleration ω_f.
Compute actual state X_f after effect of unmodeled acceleration has been added.
Compute actual state deviations $\delta\tilde{X}_f$.
Compute predicted state deviations $\delta\tilde{\hat{X}}_f$ and $\delta\tilde{\hat{X}}_{s_f}$ at t_f.
RETURN

STAPRL Analysis

The ecliptic components of the position and velocity of a tracking station relative to the Earth are related to station location parameters R, θ, and ϕ through the following set of equations:

$$X_s = R \cos\theta \cos G$$

$$Y_s = R \cos\theta \cos\epsilon \sin G + R \sin\theta \sin\epsilon$$

$$Z_s = -R \cos\theta \sin\epsilon \sin G + R \sin\theta \cos\epsilon$$

$$\dot{X}_s = -\omega R \cos\theta \sin G$$

$$\dot{Y}_s = \omega R \cos\theta \cos\epsilon \cos G$$

$$\dot{Z}_s = -\omega R \cos\theta \sin\epsilon \cos G$$

where $G = \phi + \omega (t - T)$, and T is the universal time at some epoch(usually launch time).

Subroutine STAPRL computes the negative of the partials of the previous quantities with respect to the station location parameters R, θ, and ϕ. These partials are summarized below:

$$-\frac{\partial X_s}{\partial R} = -\cos\theta \cos G$$

$$-\frac{\partial X_s}{\partial \theta} = R \sin\theta \cos G$$

$$-\frac{\partial X_s}{\partial \phi} = R \cos\theta \sin G$$

$$-\frac{\partial Y_s}{\partial R} = -\left[\sin\epsilon \sin\theta + \cos\epsilon \cos\theta \sin G\right]$$

$$-\frac{\partial Y_s}{\partial \theta} = R \cos\epsilon \sin\theta \sin G - R \sin\epsilon \cos\theta$$

$$-\frac{\partial Y_s}{\partial \phi} = -R \cos\epsilon \cos\theta \cos G$$

$$-\frac{\partial Z_s}{\partial R} = \sin\epsilon \cos\theta \sin G - \cos\epsilon \sin\theta$$

$$-\frac{\partial Z_s}{\partial \theta} = -\left[R \sin \epsilon \sin \theta \sin G + R \cos \epsilon \cos \theta\right]$$

$$-\frac{\partial Z_s}{\partial \phi} = R \sin \epsilon \cos \theta \cos G$$

$$-\frac{\partial \dot{X}_s}{\partial R} = \omega \cos \theta \sin G$$

$$-\frac{\partial \dot{X}_s}{\partial \theta} = -\omega R \sin \theta \sin G$$

$$-\frac{\partial \dot{X}_s}{\partial \phi} = \omega R \cos \theta \cos G$$

$$-\frac{\partial \dot{Y}_s}{\partial R} = -\omega \cos \theta \cos \epsilon \cos G$$

$$-\frac{\partial \dot{Y}_s}{\partial \theta} = \omega R \cos \epsilon \sin \theta \cos G$$

$$-\frac{\partial \dot{Y}_s}{\partial \phi} = \omega R \cos \epsilon \cos \theta \sin G$$

$$-\frac{\partial \dot{Z}_s}{\partial R} = \omega \sin \epsilon \cos \theta \cos G$$

$$-\frac{\partial \dot{Z}_s}{\partial \theta} = -\omega R \sin \epsilon \sin \theta \cos G$$

$$-\frac{\partial \dot{Z}_s}{\partial \phi} = -\omega R \sin \epsilon \cos \theta \sin G$$

STIMP Analysis

Subroutine STIMP converts a planetocentric ecliptic state vector $(\underline{r}, \underline{v})^T$ to the more readily targetable impact plane coorinates B·T and B·R. These coordinates are preferred as target variables for two basic reasons: (1) they generally exhibit reasonably linear dependence on the targeting controls, and (2) in probe targeting they obviate the need for defining a pseudoimpact point when the probe misses the planet in an early interation.

The impact coordinates are defined in terms of the direction of the trajectory hyperbolic excess velocity, $\underline{v}_\infty$, and the north ecliptic pole vector, $\underline{K}$. Let

$$\underline{S} = \underline{v}_\infty \, v_\infty \tag{1}$$

$$\underline{T} = \frac{\underline{S} \times \underline{K}}{\|\underline{S} \times \underline{K}\|} \tag{2}$$

$$\underline{R} = \underline{S} \times \underline{T} \tag{3}$$

where all the vectors are assumed to originate at the planet center. Thus $\underline{T}$, $\underline{R}$ and $\underline{S}$ form a right-handed Cartesian frame with $\underline{S}$ pointing in the direction of the hyperbolic excess velocity and $\underline{T}$ lying in the ecliptic plane. The plane containing the vector vector $\underline{T}$ and $\underline{R}$ is known as the impact plane. $\underline{B}$ is defined as that unique vector from the planet center to the point where the trajectory asymptote pierces the impact plane. B·T and B·R are then simply the components of $\underline{B}$ along the T and R axes, respectively. Figure 1 illustrates all of these terms for the case of a single probe trajectory.

In addition to calculating B·T and B.R, STIMP also makes available other approach trajectory parameters useful in the STEAP auxiliary targeting scheme. These are $\underline{S}$, $\underline{T}$. $\underline{R}$, and $\underline{B}$ in the inertial ecliptic frame as well as the approach hyperbola semimajor axis, a.

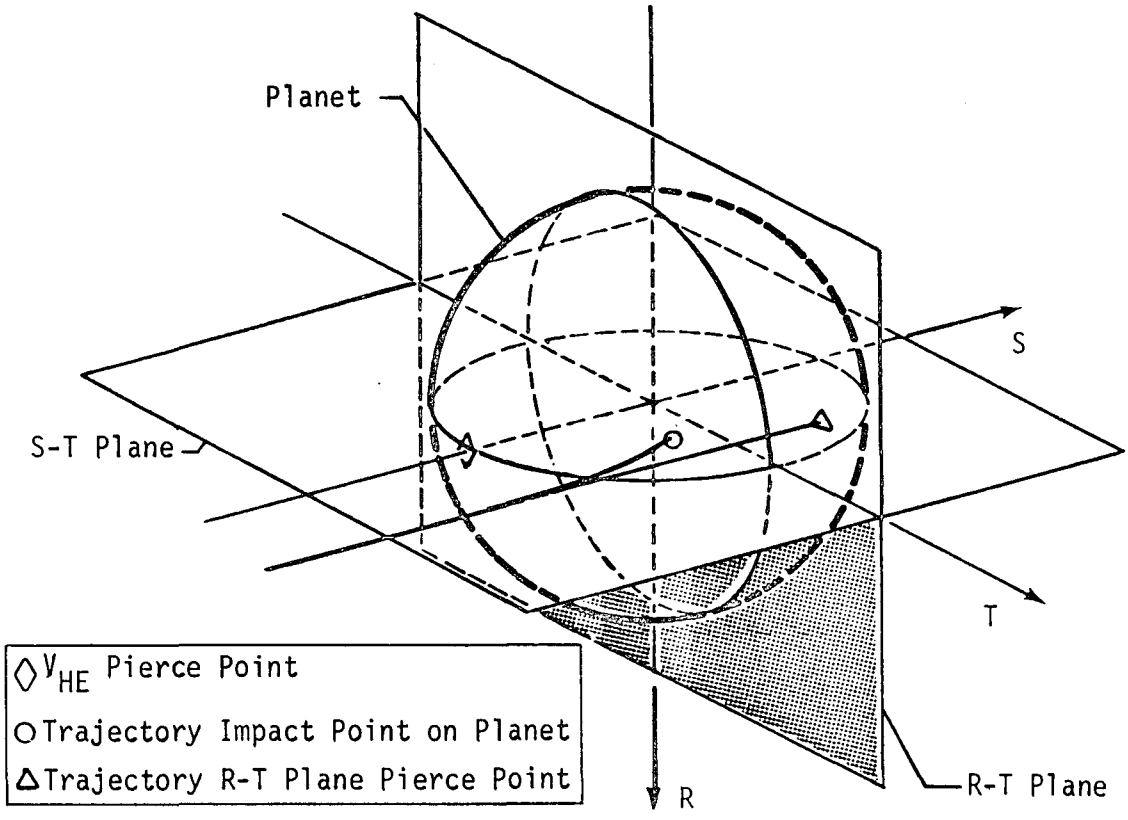


Figure 1 Single-Probe Auxiliary Targeting Illustration

The formulae used in calculating these elements are well known results derived in most engineering treatments of two-body motion (see, for example, Ref 2). Hence they will simply be listed here in the order they are used. The standard conic symbolism is used throughout:

$$\underline{h} = \underline{r} \times \underline{v} \tag{4}$$

$$\underline{W} = \underline{h}/h \tag{5}$$

$$\dot{r} = \underline{r} \cdot \underline{v}/r \tag{6}$$

$$p = h^2/\mu \tag{7}$$

$$a = r/(2 - rv^2/\mu) \tag{8}$$

$$e = \sqrt{1 - p/a} \tag{9}$$

$$\cos \theta = (p - r)/er \tag{10}$$

$$\sin\theta = \dot{r}h/e\mu \tag{11}$$

$$B = \sqrt{p|a|} \tag{12}$$

$$\underline{Z} = (r\underline{v} - \dot{r}\underline{r})/h \tag{13}$$

$$\underline{P} = \frac{r}{r}\cos\theta - \underline{Z}\sin\theta \tag{14}$$

$$\underline{Q} = \frac{r}{r}\sin\theta + \underline{Z}\cos\theta \tag{15}$$

$$\underline{S} = \underline{P}/e + \frac{Q}{e}\sqrt{e^2 - 1} \tag{16}$$

$$\underline{B} = p\underline{P}/e + \frac{aQ}{e}\sqrt{e^2 - 1} \tag{17}$$

$$\underline{T} = \left(S_2^2, -S_1^2, 0\right)^T / \sqrt{S_1^2 + S_2^2} \tag{18}$$

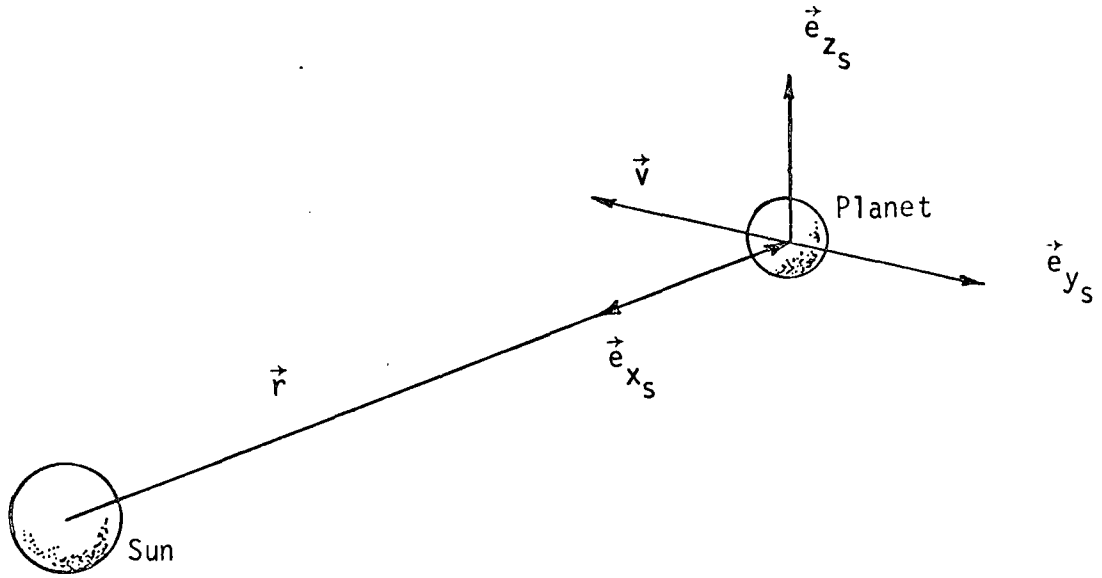
$$\underline{R} = \left(-S_3T_2, S_3T_1, S_1T_2 - S_2T_1\right)^T \tag{19}$$

$$B\cdot T = B_1T_1 + B_2T_2 \tag{20}$$

$$B\cdot R = B_1R_1 + B_2R_2 + B_3R_3 \tag{21}$$

SUBSØL Analysis

Subroutine SUBSØL computes the transformation from planetocentric ecliptic coordinates to subsolar planet orbital plane coordinates for an arbitrary planet. The subsolar planet orbital plane coordinate system is defined as the planetocentric system whose x-axis points directly at the sun, whose z-axis is normal to the planet's orbital plane, and whose y-axis is normal to the xz-plane and lies in the planet's orbital plane. In the figure below $\vec{r}$ and $\vec{v}$ denote the position and velocity vectors, respectively, of the planet relative to the sun. Unit vectors $\vec{e}_x$, $\vec{e}_y$, and $\vec{e}_z$ are aligned with the axes of the subsolar planet orbital plane system.



These unit vectors are defined as

$$\vec{e}_{x_s} = -\frac{\vec{r}}{r}$$

$$\vec{e}_{y_s} = \vec{e}_{z_s} \times \vec{e}_{x_s}$$

$$\vec{e}_{z_s} = \frac{\vec{r} \times \vec{v}}{|\vec{r} \times \vec{v}|}$$

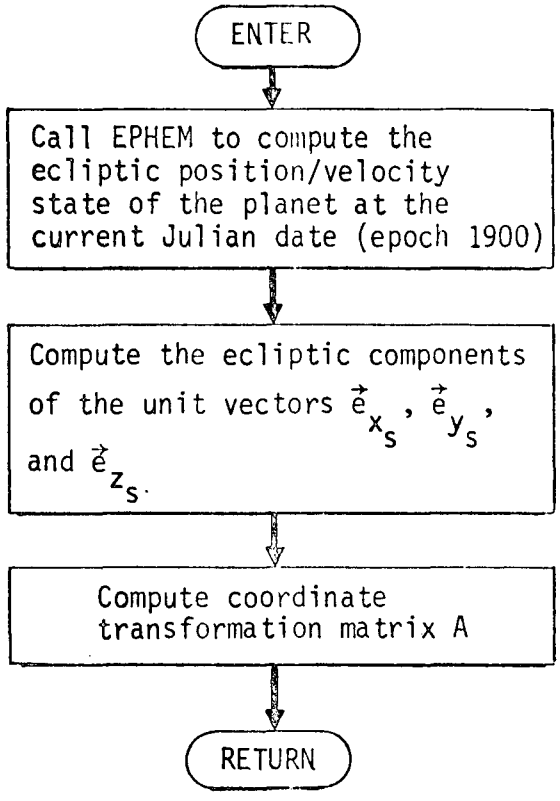
If these unit vectors are referred to the ecliptic coordinate system, the coordinate transformation A from planetocentric ecliptic to subsolar planet orbital plane coordinates is given by

$$A = \begin{bmatrix} \vec{e}_{x_s}^{\,T} \\ - - - \\ \vec{e}_{y_s}^{\,T} \\ - - - \\ \vec{e}_{z_s}^{\,T} \end{bmatrix}$$

Thus

$$\vec{x}_{\text{subsolar}} = A\,\vec{x}_{\text{ecliptic}}$$

SUBSØL Flow Chart



TARGET Analysis

TARGET is responsible for the control of any targeting (nonlinear guidance) event. The targeting is done either by the Newton-Raphson technique or by a steepest descent-conjugate gradient algorithm, the method being specified by the user. In either case numerical differencing is used to compute the required sensitivities.

I. Preliminaries

The current inertial state of the spacecraft upon entering TARGET is first saved (RIS=RIN) along with the original SOI radius (OSPH=SPHERE) since both variables may be changed during the course of the targeting. Before exiting from TARGET these values are restored.

The index of the current event KUR has been computed by TRJTRY. This enables the specific targeting parameters for the current event to be set:

Parameter	Definition
METHOD	Triggers Newton Raphson (=0) or Steepest Descent (≠0) technique
MATX	Determines whether Newton-Raphson matrix is computed always (=2) or only at low level (=1)
IBAST	Determines whether bad step checks are made never (=1), high level only (=2) or always (=3)
LEVELS	Number of integration accuracy levels to be used
NOPAR	Number of target parameters to be used
ACC	Actual accuracy levels used in targeting

The following flags are then initialized to zero

Flag	Definition
ITDS	Counter for steepest descent iterations
LOWHI	Flag indicating whether first phase complete (=1) or not (=0)
NOMORE	Flag indicating whether outer targeting has been done (=1) or not (=0)

The target time is computed and using that time the transformation matrix ϕ_{ECEQ} from ecliptic to target planet equatorial coordinates is calculated (PECEQ).

II. Phase Preparations

TARGET performs the targeting in one phase unless targeting to TCA (time of closest approach). In that case the trajectory is targeted in two phases: the first phase targets to the target planet SOI (sphere of influence), the second phase to the closest approach conditions. IPHASE is the phase counter, NOPHAS is the number of phases needed.

If all the phases have been completed, the program prepares to exit. If the last iterate satisfied the target tolerances ITOL will have been set to a 1. If it did not, ITOL will be zero and this requires that KWIT be set to 1 to terminate the program upon return to the basic cycle.

If the last phase has not yet been completed TAROPT is now called with an argument 1 to compute the following phase parameters:

Parameter	Definition
KEYTAR(3)	Vector of codes of target parameters
KAXTAR(3)	Vector of codes of auxiliary parameters
DTAR(3)	Vector of desired values of target parameters
DAUX(3)	Vector of desired values of auxiliary parameters
FAC(3)	Weighting factors for loss function for auxiliary parameters
ISTOP	Flag indicating integration stopping conditions with ISTOP = 1,2,3 indicating fixed final time, SOI, or CA encounter

The target parameters are the parameters actually desired; the auxiliary parameters are the parameters used to do the targeting. The target and auxiliary parameters are identical except when i_{CA} and r_{CA} are targets. In that case the corresponding auxiliary parameters are $B \cdot T$ and $B \cdot R$ which are much more linear variables. The codes of the target and auxiliary parameters are as follows:

Code	1	2	3	4	5	6	7	8	9	10	11	12
Parameter	TRF*	TSI	TCS	TCA	BDT	BDR	RCA	INC	SMA	XRF	YRF	ZRF

* not currently available

III. Level Preparations

Within any phase TARGET operates through a series of integration accuracy levels prescribed by the user. After completing each level TARGET checks to see if the maximum number of levels LEVELS has been exceeded. If it has the program cycles to the beginning of the "phase loop" to go to the next phase. If the current level LEV is less than LEVELS the following computations are made.

The flag ITARM controls whether the previous targeting matrix is to be used (=1) or whether the matrix is to be recomputed (=2) during the current level. ITARM is set according to the current values of MATX, ISTART, and LEV.

The flag IBAD controls the bad step logic. If IBAD=1 no bad step check will be made during the current level; if IBAD=2 the bad step check will be in effect. TARGET sets IBAD according to the values of IBAST and LEV.

The flags ITOL, ITER, ITBAD are set to 0 to begin the iterations. The allowable iterations NITS and bad iterations MAXBAD are also set at this time.

IV. Iterate Calculations

Within each level the program makes one or more iterations. After each iteration the program updates the iteration counter ITER. If the maximum number of iterations for this level NITS has been exceeded, the program sets KWIT to 1 and prepares for the return from TARGET. Otherwise TARGET computes the target and auxiliary values corresponding to the current iterate values of state (position and velocity) RIN.

The integration parameters are first set. VMP is then called to propagate the initial state to the final stopping conditions. Checks are made to insure that the target planet SOI was intersected if the stopping conditions were SOI or CA. If it was not intersected and this is the first iteration, the "outer targeting" phase is entered (see below). If "outer targeting" has already been performed, the bad-step check is entered to reduce the previous correction by REDUC.

Otherwise TAROPT is called with the argument 2 to compute the desired and actual target (DTAR, ATAR) and auxiliary (DAUX, AAUX) parameter values. The absolute error in target values AER and the error in auxiliary values DEV are then computed.

If the current iterate is the first integration at the low level during the second phase of targeting (LOWHI=1) TARMAX is now called to compute the phase 2 targeting matrix. Then the state RIN is reset to the targeted velocity at the high level TVH to prepare for the second phase targeting. The program then returns to the level loop.

Otherwise the program now checks the actual target variables to determine whether they satisfy the input tolerances or not.

V. Tolerances Satisfied

If the tolerances are satisfied, the program first checks to see if the current targeting phase is outer targeting. If it is TARGET restores the original target parameters and initiates the normal targeting (see Outer Targeting below).

If the current targeting is already normal targeting, TARGET sets ITOL=1 to indicate the satisfaction of the tolerances. If the problem is a 2-phase and the current level is the highest level in phase 1 targeting, the targeted high level velocity TVH=RIN is saved, LOWHI is set to 1 and the targeted low level velocity is recalled RIN=TVL for the construct of the phase 2 targeting matrix. Then the level loop is reentered.

VI. Bad Step Reduction

If the target parameter values of any iterate are not within the acceptable tolerances TARGET now assigns a scalar error ϵ to the iterate using the weighting factors $\vec{W}$

$$\epsilon = \vec{W} \cdot \Delta \vec{T}$$

If the bad-step check is to be made on this iterate the current error ϵ is compared to the previous error ϵ_p. If $\epsilon > \epsilon_p$ and the maximum number of bad steps has not been exceeded, the previous correction $\Delta\vec{v}$ is reduced by REDUC (usually 1/4). The initial state RIN is adjusted by this and the iterate loop is reentered. If the maximum number of bad steps has been made, KWIT is set to 1 and the preparations for return are made.

VII. Generation of Next Iterate

The correction $\Delta\vec{v}$ to any iterate may be computed from either of two techniques selected by the flag METHOD. If METHOD $\neq$ 0, subroutine DESCENT is called for the computation of the $\Delta\vec{v}$ by a steepest descent algorithm. The numerical value of METHOD determines the number n of conjugate gradient steps between each straight gradient step where n = METHOD-1. Thus if METHOD=1, every step is in the direction of the gradient. But if METHOD=5, four steps are taken following the conjugate gradient direction before rectification by the gradient direction.

If METHOD=0, the Newton-Raphson correction is used. If ITARM=0, TARMAX is called for the computation of the targeting matrix ϕ by numerical differencing. If any of the integrations made in constructing that matrix satisfy the tolerances in τ, the flag IEND is set to 1 before returning to TARGET. Thus a check must be made on IEND. If ITARM=1 the previous targeting matrix is used. The correction is then given by

$$\Delta\vec{v} = \phi \cdot \Delta\vec{\alpha}$$

where $\Delta\vec{\alpha}$ are the deviations in the iterate auxiliary values. The $\Delta\vec{v}$ is checked to insure that the maximum step size DVMAX is not violated: if it is, the $\Delta\vec{v}$ is reduced proportionately to satisfy it. The next iterate is then set to

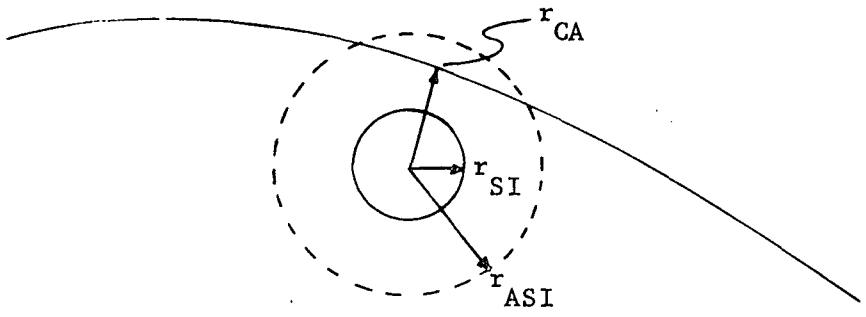
$$(\vec{r}, \vec{v}) = (\vec{r}, \vec{v} + \Delta\vec{v})$$

and the return is made to the iterate loop.

VIII. Outer Targeting

Occasionally the zero iterate initial state leads to a trajectory missing the target body SOI. Since all target options except one (targeting to a specified position, i.e., KTAR = 10,11,12) require the trajectory to intersect the target body SOI steps must be taken to correct this.

Let the initial state propagated forward lead to a trajectory with a closest approach to the target body of r_{CA} with $r_{CA} > r_{SI}$ where r_{SI} is the radius of the SOI.



Until the initial trajectory intersects the SOI the usual targeting can not be done. Therefore an "artificial" SOI is introduced having a radius of

$$r_{ASI} = 1.2 \times r_{CA}$$

The initial trajectory obviously intersects the artificial SOI and hence may be targeted to conditions on the ASOI. If the target conditions are established as $B \cdot T_A = B \cdot R_A = 0$, when this artificial targeting is completed, the refined trajectory will be headed straight for the target body when it hits the ASOI. Thus the refined trajectory should automatically hit the normal SOI when propagated past the ASOI. To insure that the time of intersection with the normal SOI is consistent with the target time, an artificial target time is also used. Let the speed of the spacecraft with respect to the target body at r_{CA} be v_{CA}. Make the approximation that this speed will be roughly the same for the refined trajectory. Then the time that the spacecraft should intersect the ASOI is

$$t_{ASI} = t_{CA} - \frac{r_{ASI}}{v_{CA}}$$

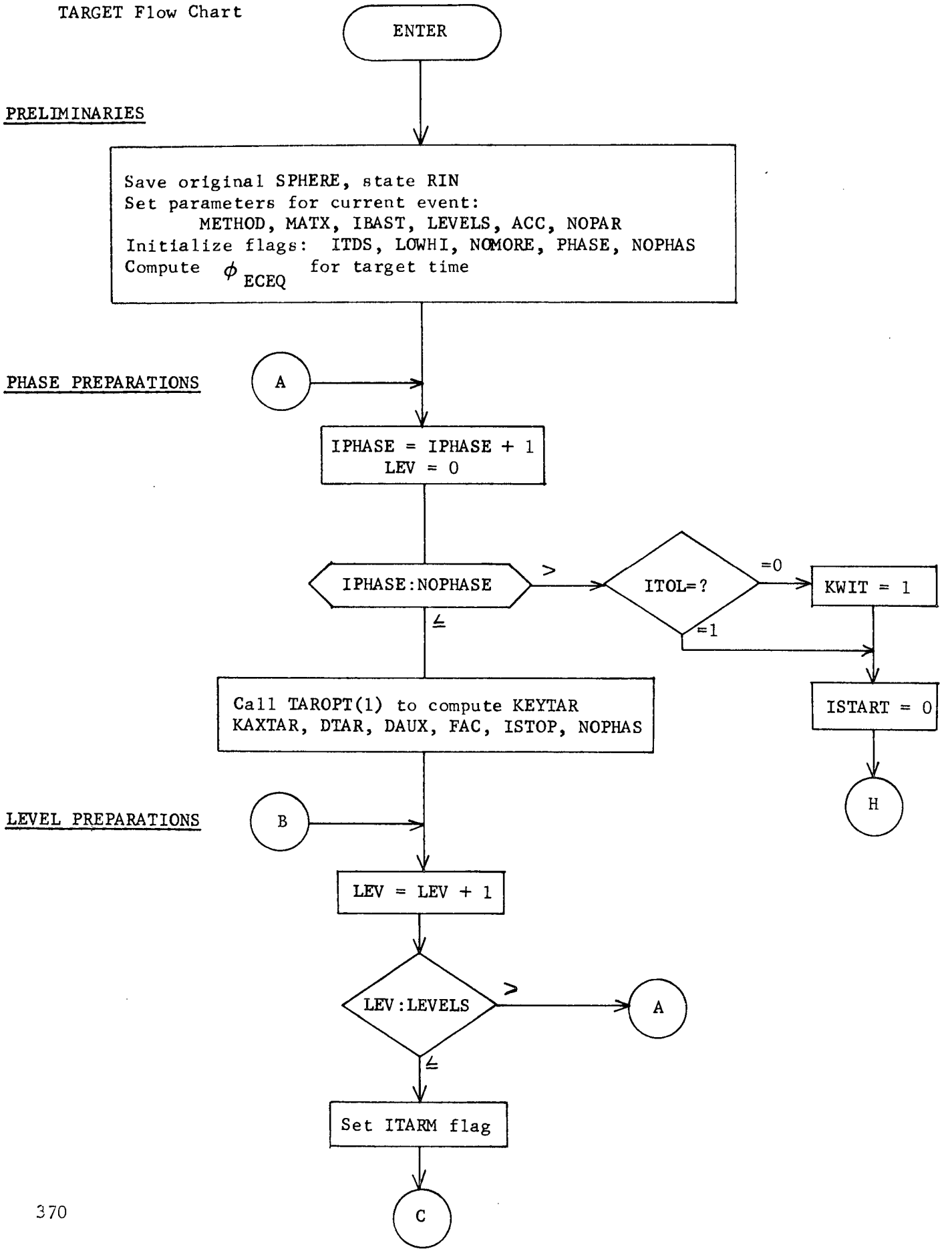
or

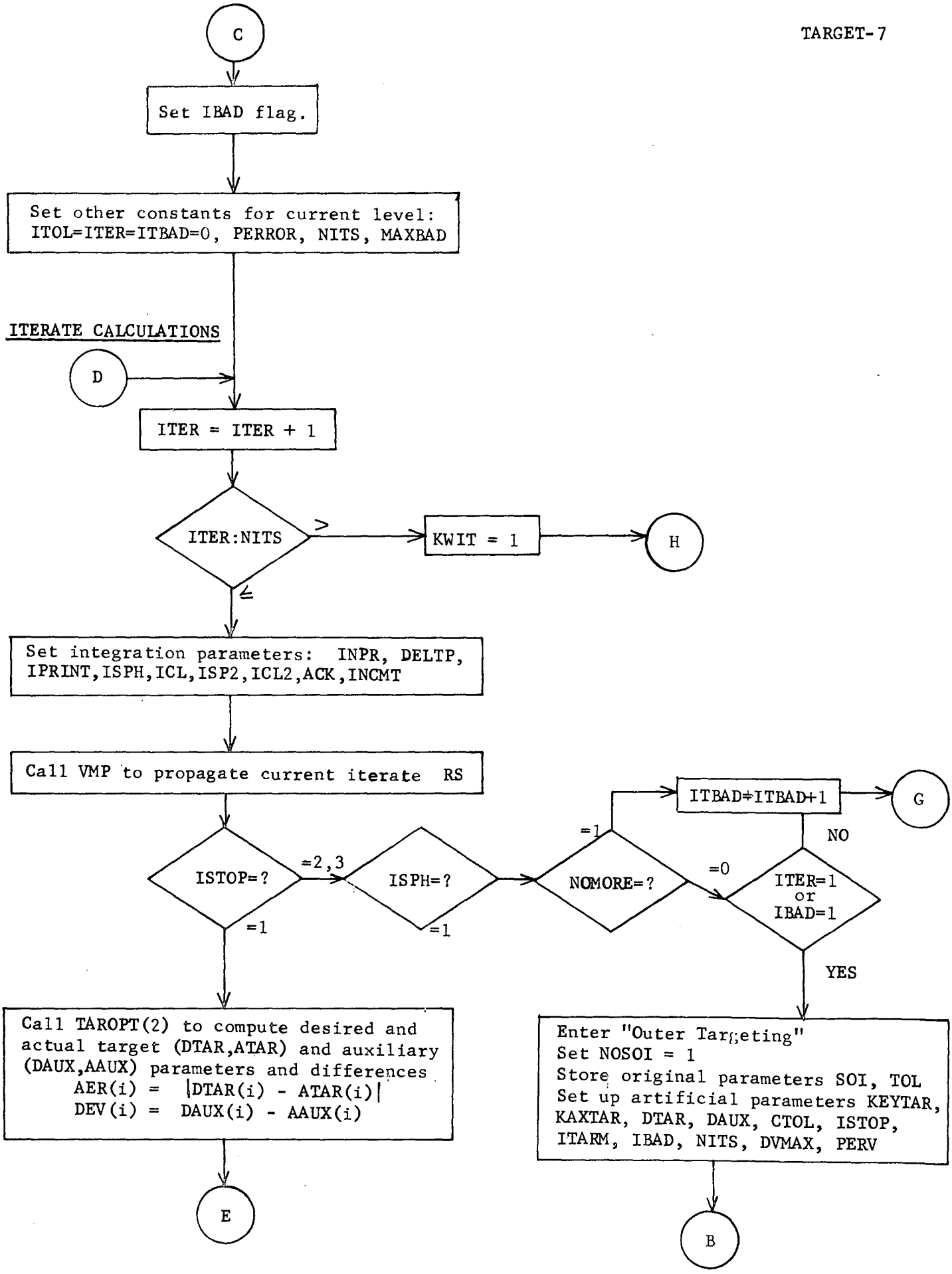
$$t_{ASI} = t_{SI} - \frac{r_{ASI} - r_{SI}}{v_{CA}}$$

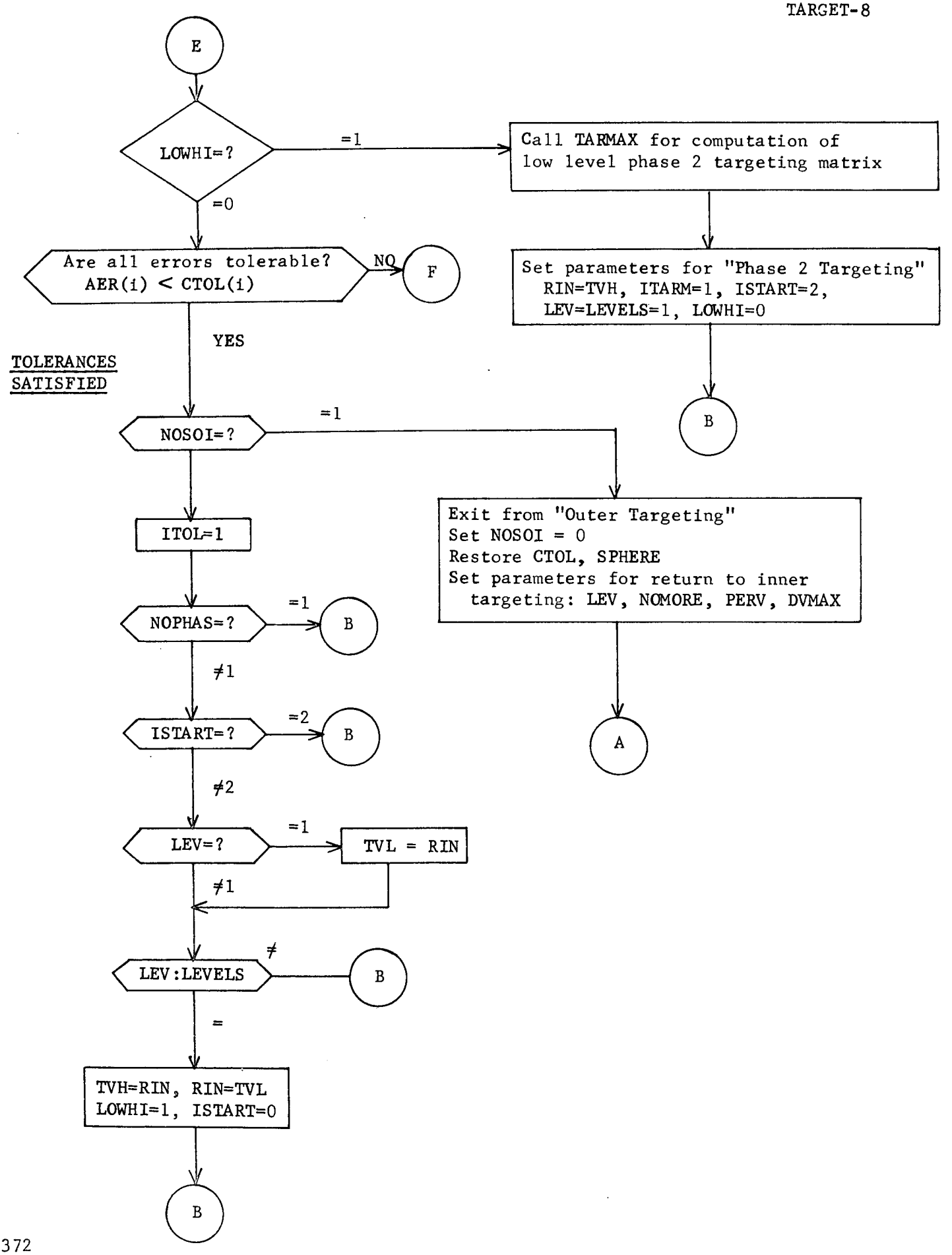
where the first formula should be used if the target time is t_{CA} or t_{CS} and the second formula is used for t_{SI}.

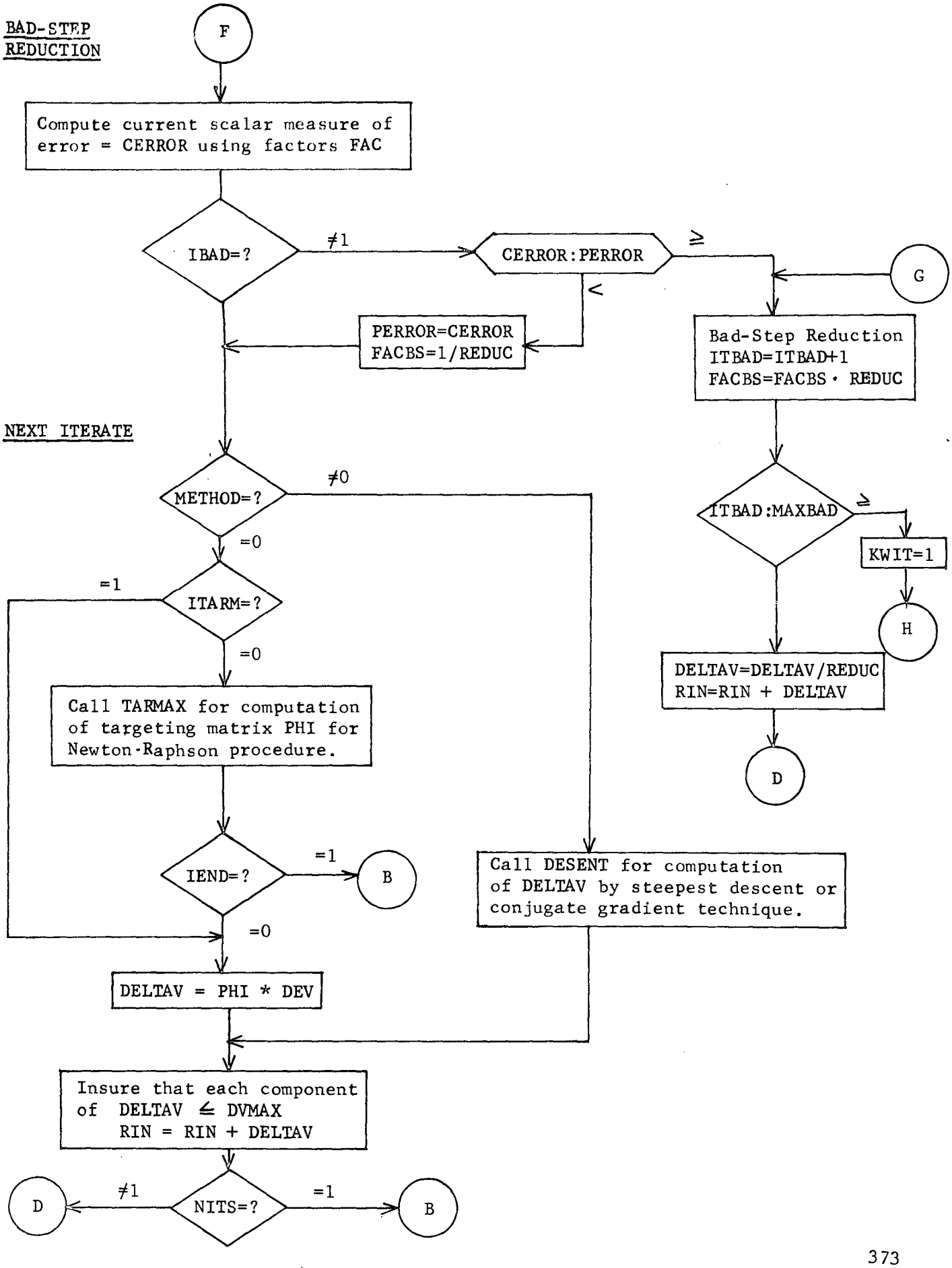
Thus when a trajectory is found which misses the normal SOI, the closest approach state r_{CA}, v_{CA} is recorded. The normal SOI radius is stored and the artificial SOI radius given above is used in its place. Target parameters of $B \cdot T_A$, $B \cdot R_A$, and t_{ASI} are then set up as the targets. When targeting of this artificial problem is complete, the resulting trajectory will intersect the normal SOI and the original problem may be solved.

TARGET Flow Chart

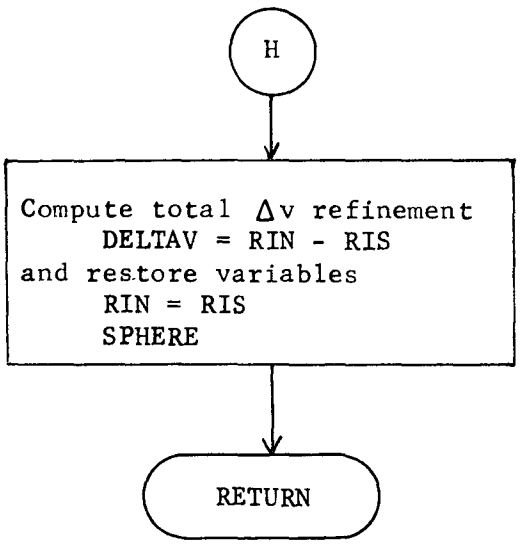



C
Set IBAD flag.
Set other constants for current level:
ITOL=ITER=ITBAD=0, PERROR, NITS, MAXBAD
ITERATE CALCULATIONS
D
ITER = ITER + 1
ITER:NITS
>
KWIT = 1
H
≤
Set integration parameters: INPR, DELTP,
IPRINT,ISPH,ICL,ISP2,ICL2,ACK,INCMT
Call VMP to propagate current iterate RS
ISTOP=?
=2,3
ISPH=?
=1
=1
NOMORE=?
=1
ITBAD=ITBAD+1
G
NO
=0
ITER=1
or
IBAD=1
YES
Call TAROPT(2) to compute desired and
actual target (DTAR,ATAR) and auxiliary
(DAUX,AAUX) parameters and differences
AER(i) = |DTAR(i) - ATAR(i)|
DEV(i) = DAUX(i) - AAUX(i)
E
Enter "Outer Targeting"
Set NOSOI = 1
Store original parameters SOI, TOL
Set up artificial parameters KEYTAR,
KAXTAR, DTAR, DAUX, CTOL, ISTOP,
ITARM, IBAD, NITS, DVMAX, PERV
B


E
LOWHI=?
=1
Call TARMAX for computation of low level phase 2 targeting matrix
=0
Are all errors tolerable? AER(i) < CTOL(i)
NO
F
Set parameters for "Phase 2 Targeting" RIN=TVH, ITARM=1, ISTART=2, LEV=LEVELS=1, LOWHI=0
YES
TOLERANCES SATISFIED
B
NOSOI=?
=1
Exit from "Outer Targeting"
Set NOSOI = 0
Restore CTOL, SPHERE
Set parameters for return to inner targeting: LEV, NOMORE, PERV, DVMAX
ITOL=1
NOPHAS=?
=1
B
≠1
ISTART=?
=2
B
≠2
LEV=?
=1
TVL = RIN
≠1
LEV:LEVELS
≠
B
=
TVH=RIN, RIN=TVL
LOWHI=1, ISTART=0
A
B


BAD-STEP REDUCTION
F
Compute current scalar measure of error = CERROR using factors FAC
IBAD=?
≠1
CERROR:PERROR
≥
<
G
PERROR=CERROR
FACBS=1/REDUC
Bad-Step Reduction
ITBAD=ITBAD+1
FACBS=FACBS · REDUC
NEXT ITERATE
METHOD=?
≠0
=0
ITBAD:MAXBAD
≥
KWIT=1
H
=1
ITARM=?
=0
DELTAV=DELTAV/REDUC
RIN=RIN + DELTAV
D
Call TARMAX for computation of targeting matrix PHI for Newton-Raphson procedure.
IEND=?
=1
B
Call DESENT for computation of DELTAV by steepest descent or conjugate gradient technique.
=0
DELTAV = PHI * DEV
Insure that each component of DELTAV ≤ DVMAX
RIN = RIN + DELTAV
D
≠1
NITS=?
=1
B

PREPARATIONS FOR RETURN



TARMAX Analysis

TARMAX computes the targeting matrix used by TARGET for Newton-Raphson refinements. The targeting matrix is computed by numerical differencing.

Let the current iterate initial state be denoted $\vec{r}, \vec{v}$. Let the auxiliary parameters corresponding to this state be $\vec{\alpha}$. Let the perturbation size for the sensitivities be Δv.

The k-th column of the sensitivity matrix is computed as follows. Perturb the k-th component of velocity by Δv:

$$\vec{v}_p = \vec{v} + \Delta v \left[\delta_{1K}, \delta_{2K}, \delta_{3K}\right]^T \quad (1)$$

Propagate the perturbed initial state $(\vec{r}, \vec{v}_p)$ to the final stopping conditions. Let the auxiliary parameters of that trajectory be denoted α_p. The k-th column of the sensitivity matrix x is then given by

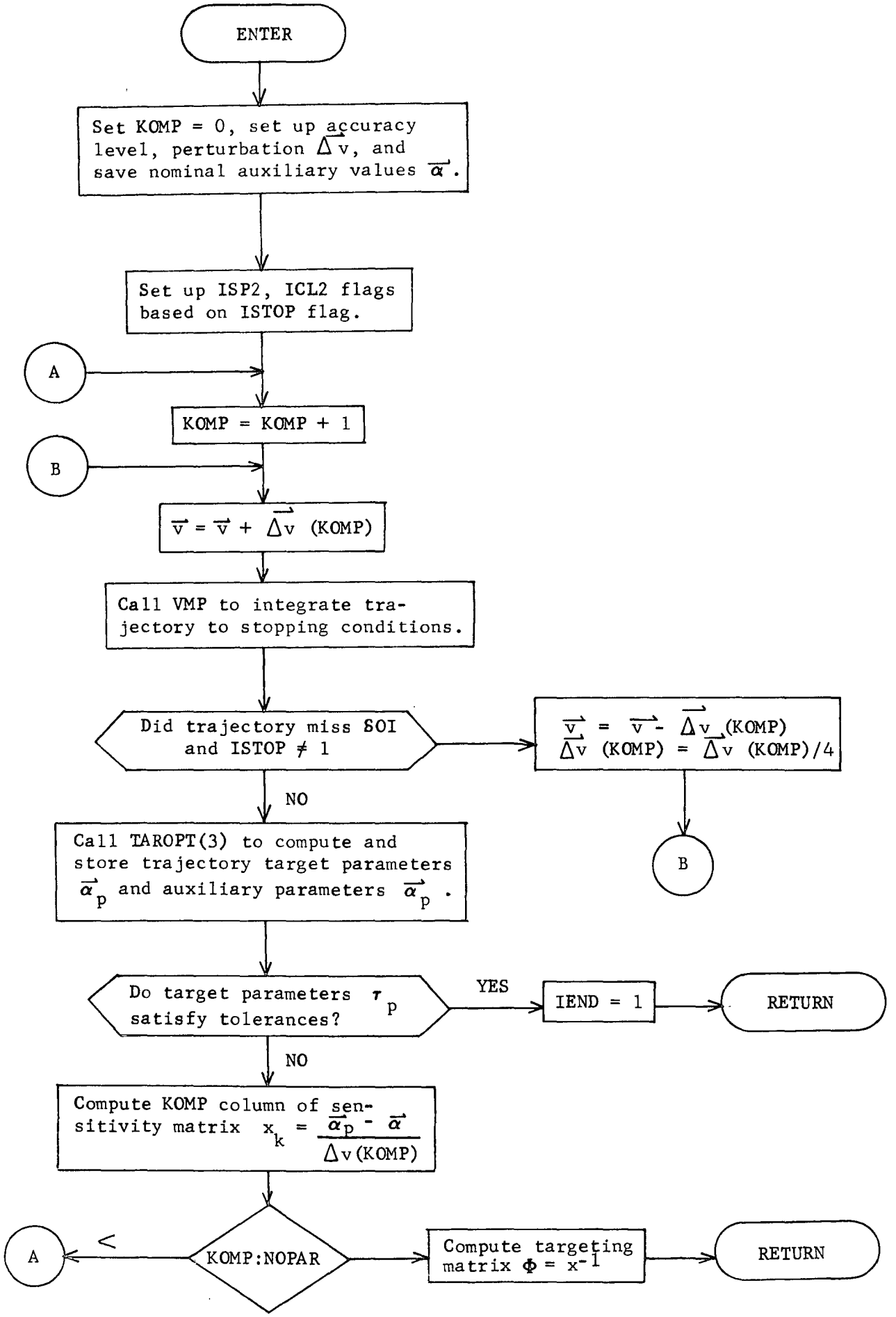
$$x_k = \frac{\vec{\alpha}_p - \vec{\alpha}}{\Delta v} \quad (2)$$

Having computed all the columns of x, the targeting matrix is then given by the inverse of x:

$$\Phi = x^{-1} \quad (3)$$

The targeting matrix then has the property that to obtain a change $\Delta\alpha$ in the nominal auxiliary parameters, the velocity should be changed by the amount

$$\vec{\Delta v} = \Phi \cdot \vec{\Delta\alpha} \quad (4)$$


ENTER
Set KOMP = 0, set up accuracy level, perturbation $\Delta\vec{v}$, and save nominal auxiliary values $\vec{\alpha}$.
Set up ISP2, ICL2 flags based on ISTOP flag.
A
KOMP = KOMP + 1
B
$\vec{v} = \vec{v} + \Delta\vec{v}$ (KOMP)
Call VMP to integrate trajectory to stopping conditions.
Did trajectory miss SOI and ISTOP ≠ 1
$\vec{v} = \vec{v} - \Delta\vec{v}$(KOMP)
$\Delta\vec{v}$ (KOMP) = $\Delta\vec{v}$ (KOMP)/4
B
NO
Call TAROPT(3) to compute and store trajectory target parameters $\vec{\alpha}_p$ and auxiliary parameters $\vec{\alpha}_p$.
Do target parameters τ_p satisfy tolerances?
YES
IEND = 1
RETURN
NO
Compute KOMP column of sensitivity matrix $x_k = \frac{\vec{\alpha}_p - \vec{\alpha}}{\Delta v(KOMP)}$
A
<
KOMP:NOPAR
Compute targeting matrix $\Phi = x^{-1}$
RETURN

TARØPT Analysis

TARØPT is responsible for computing the desired and achieved target parameter values for all the targeting subroutines. To add any new target parameters, TARØPT is the only subroutine that must be modified.

The key variables used by TARØPT and their definitions are:

Variable	-	Definition;
KTAR(6,10)	-	Codes of target parameters of all targeting event
TAR(6,10)	-	Desired values of target parameters of all targeting events;
KEYTAR(3)	-	Codes of target parameters of current event;
DTAR(3)	-	Desired values of target parameters of current event;
ATAR(3)	-	Actual values of target parameters of current iterate;
KAXTAR(3)	-	Codes of auxiliary parameters of current iterate;
DAUX(3)	-	Desired values of auxiliary parameters on current iterate;
AAUX(3)	-	Actual values of auxiliary parameters of current iterate.

The available target parameters and their codes and definitions are tabulated.

Code	Parameter	Definition
1	t_{PS}	Time at probe sphere impact (n-body integration to sphere of influence (SOI), conic propagation to probe-sphere)
2	t_{SI}	Time at SOI of target body (n-body integration to SOI)
3	t_{CS}	Time at CA (n-body integration to SOI, conic propagation to CA)
4	t_{CA}	Time at CA (n-body integration to CA)
5	B•T	Impact parameter B•T
6	B•R	Impact parameter B•R
7	i	Inclination to target planet equator
8	r_{CA}	Radius of closest approach to target body
9	a_{SI}	Semimajor axis of conic with respect to target body
10	x_f	X-component of final state (inertial ecliptic system)
11	y_f	Y-component of final state
12	z_f	Z-component of final state
13	δ_I	Declination of probe target point in planetocentric probe-sphere coordinate system specified by IPCS
14	α_I	Right ascension of probe target point in planetocentric probe-sphere coordinate system specified by IPCS
15	t_{PR}	Time at probe-sphere impact (n-body integration to probe sphere)

The term target parameter refers to a variable with a final value that conforms to a desired value. The term auxiliary parameter refers to a variable used to compute the progressive corrections. The target parameters and auxiliary parameters are identical unless the target parameters include either of the pairs i and r_{CA} or α_I and δ_I. In these cases the more linear parameters B·T and B·R are used as auxiliary targets in place of the actual ones. The desired values of the asymptote pierce-point coordinates, B_D·T and B_D·R, as well as the actual values, B_A·T and B_A·R, are computed by IMPCT for each successive iterate of the trajectory, based on the desired values of i and r_{CA} or α_I and δ_I.

TARØPT is called under three different options, which are distinguished by an argument ITARØ. The three different options will be discussed in order.

TARØPT(1) is called by TARGET at the beginning of each phase to set up the proper variables for the targeting. The arrays KEYTAR, KAXTAR, DTAR, and DAUX are set to the current event values of KTAR and TAR. If t_{CA} or t_{PR} is a target parameter, the number of phases NØPHAS is set to 2. Then for the first phase of a two-phase problem, the time target variable t_{CA} or t_{PR} is replaced by t_{CS} or t_{PS}, respectively, in the KEYTAR and KAXTAR arrays. Thus in the initial high-speed phase of a high-precision double-phase targeting problem, integration is stopped at the SOI and the trajectory is extrapolated to the target conditions as in a single-phase case. If i and r_{CA} or α_I and δ_I are included in the actual target parameters, the corresponding indices of the KAXTAR array are set for the auxiliary targets B·T and B·R. TARØPT then sets up the integration parameters.

There are three propagation stop modes. The first terminates the trajectory after a maximum prepagation time interval, Δt, set equal to the nominal difference between the target time t_T and the current guidance event time t_G:

$$\Delta t = t_T - t_G \quad . \tag{1}$$

For this stopping condition, ISTØP is set to 1. The second termination mode stops propagation at the SOI (or at impact if the SOI radius is temporarily set to that of the probe sphere). This ISTØP = 2 mode is used when the target time is t_{SI}, t_{CS}, t_{PS}, or t_{PR}.

The final ISTØP = 3 mode terminates the propagation at closest approach. It is used only when the target time is t_{CA}. In either of the last two modes, the maximum propagation time interval is set to 1.1 times the value assigned for the first mode except when one of the target parameters is a_{SI}, the semimajor axis of the planetocentric conic at the target time. In this case Δt is set to the same value as in the first stopping mode. The augmented propagation interval guarantees that the trajectory will be propagated long enough to reach the desired targeting termination.

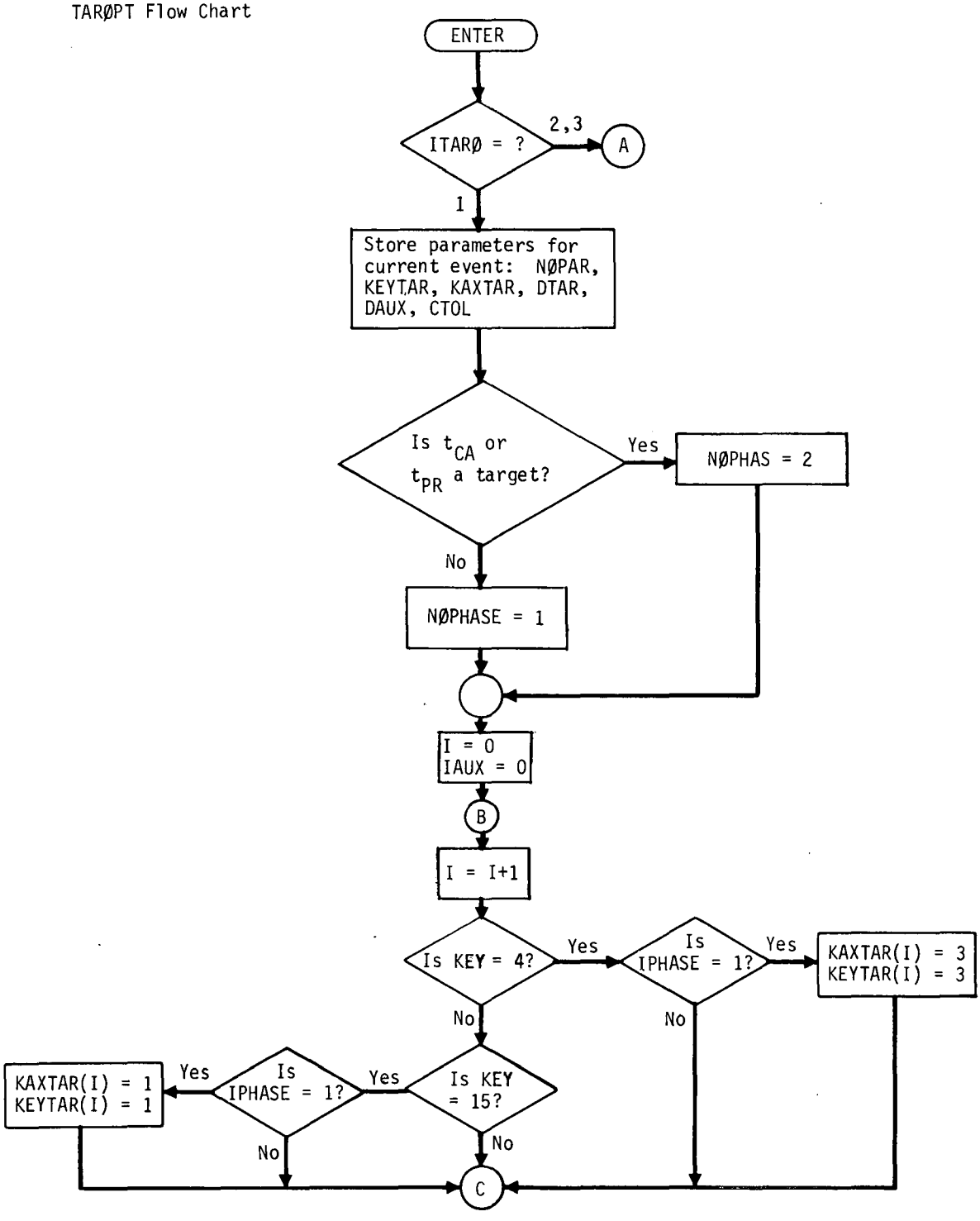
Finally the weighting factors FAC(3) used in computing the scalar loss function are set. Since the loss function is calculated solely from the auxiliary targets, which all have units of either length or time, only the relative weight of time to length need be input. Thus the length factors are set to unity and the time factor to the input value in WGHTM.

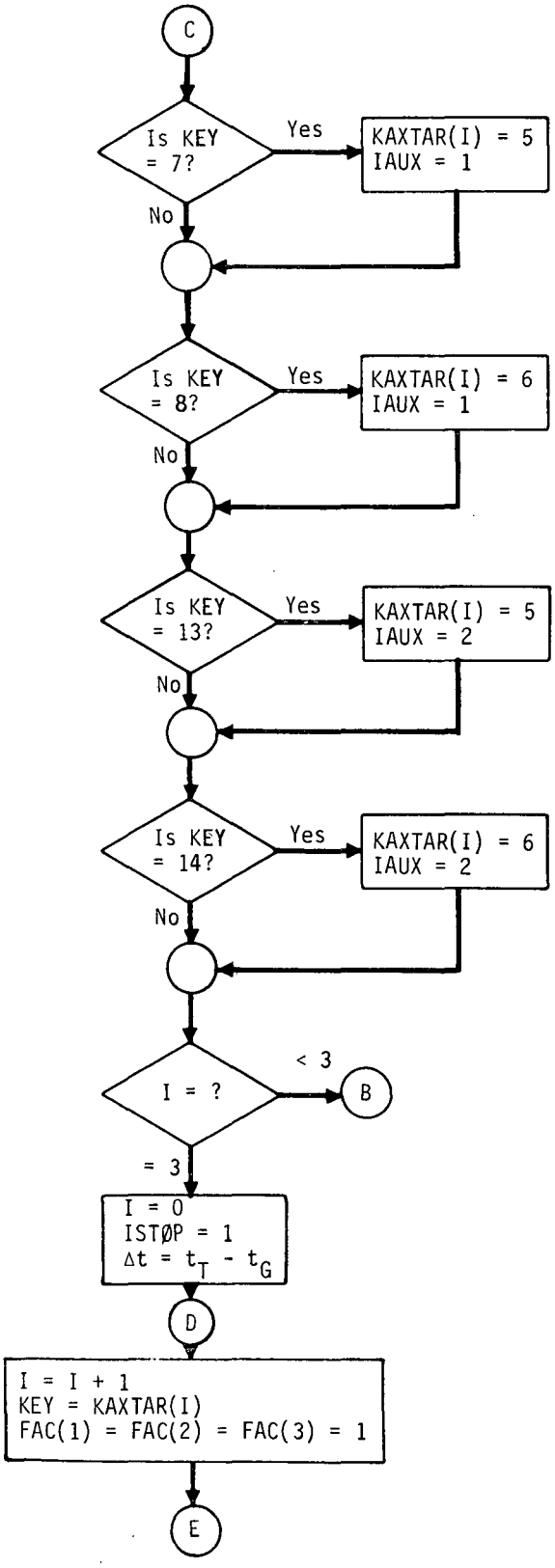
TARØPT(2) is called by TARGET after integrating each iterate to the final stopping conditions. Here TARØPT performs mainly a bookkeeping role. It must fill the proper cells of the ATAR, AAUX, and DAUX arrays with values generally computed by the virtual-mass routines. If auxiliary targeting is required, both the actual and desired values of the B-plane coordinates are computed by calling IMPCT.

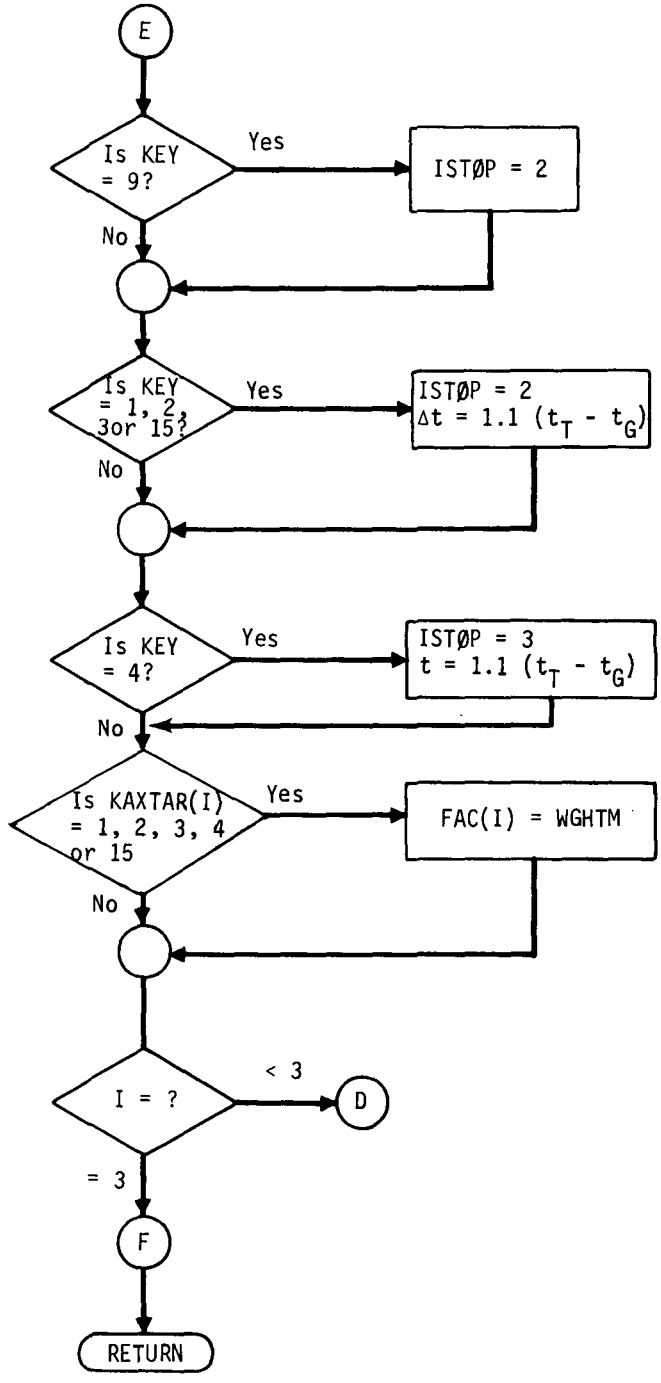
Since TARØPT(3) is called by TARMAX and DESENT after integrating each perturbed trajectory to compute the perturbed values of the auxiliary parameters, the desired values of DAUX need not be computed at this time. If auxiliary targeting is in process, the actual values of the B-plane coordinates, $B_A \cdot T$ and $B_A \cdot R$, are computed by a call to IMPCT. Once again this task is simply a bookkeeping job to store the trajectory data correctly in the ATAR and AAUX cells. TARMAX and DESENT may then operate easily on these arrays to compute the targeting matrix or gradient directions.

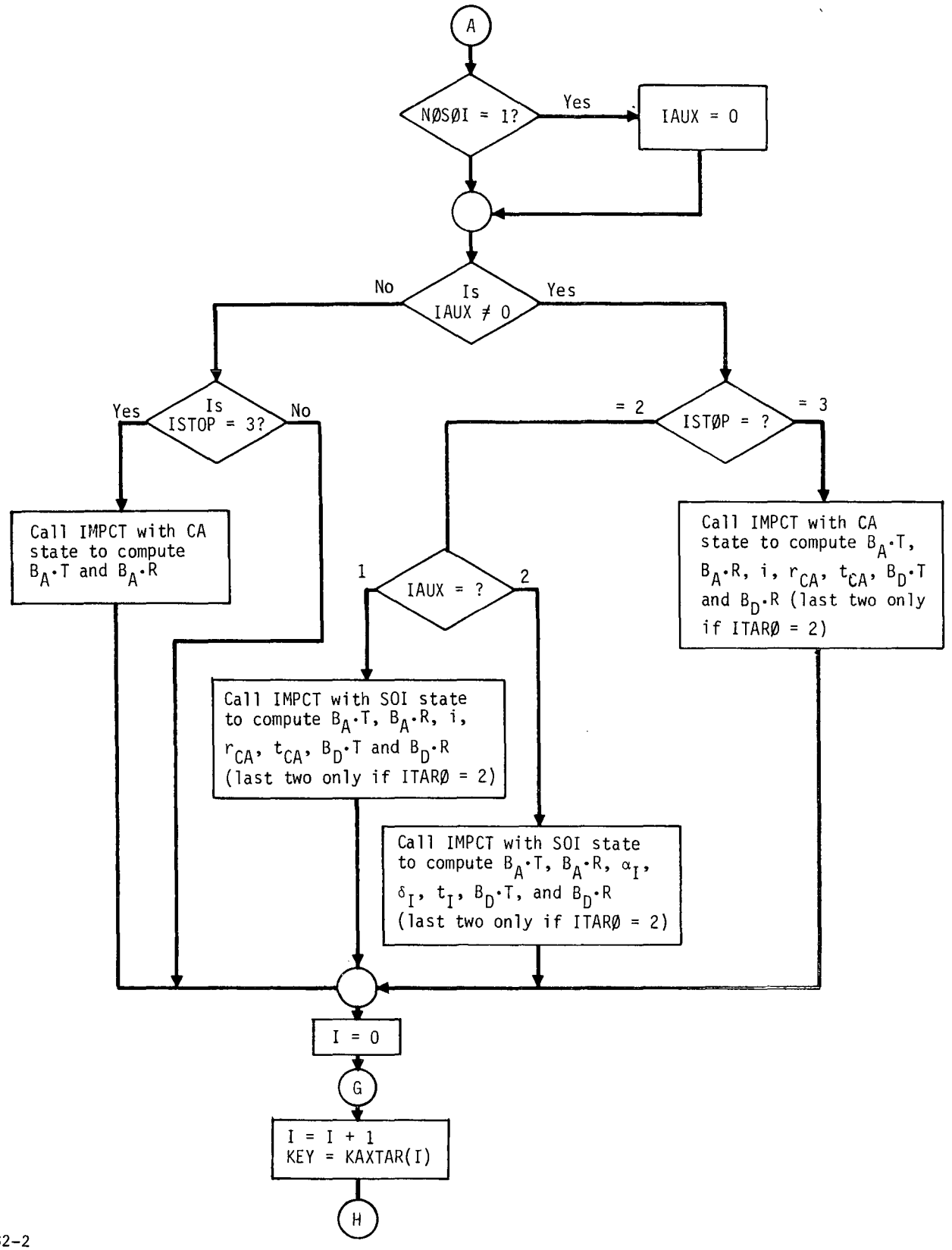
In both calls TARØPT(2) and TARØPT(3), the trajectory data are printed out before exiting from TARØPT.

TARØPT Flow Chart

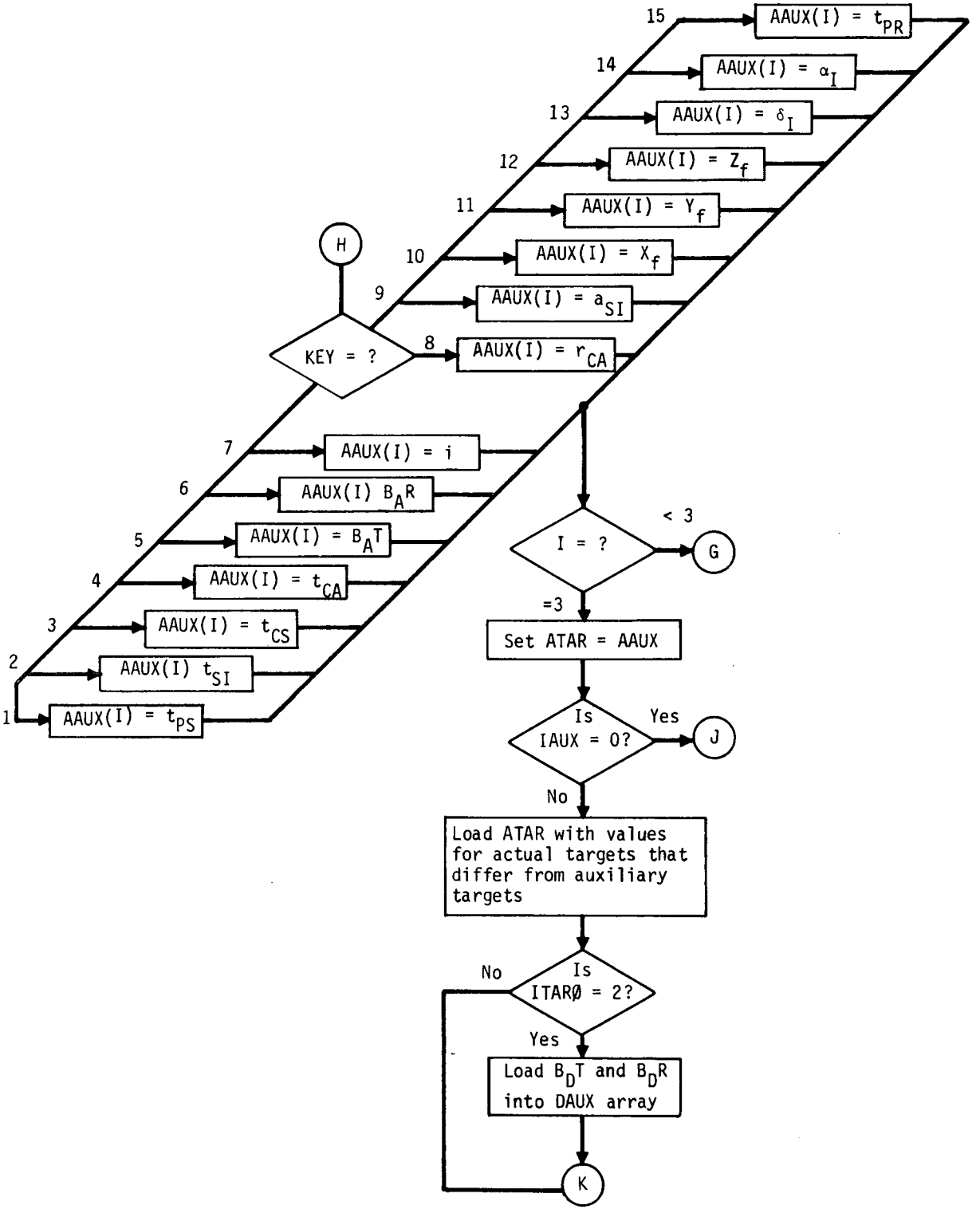


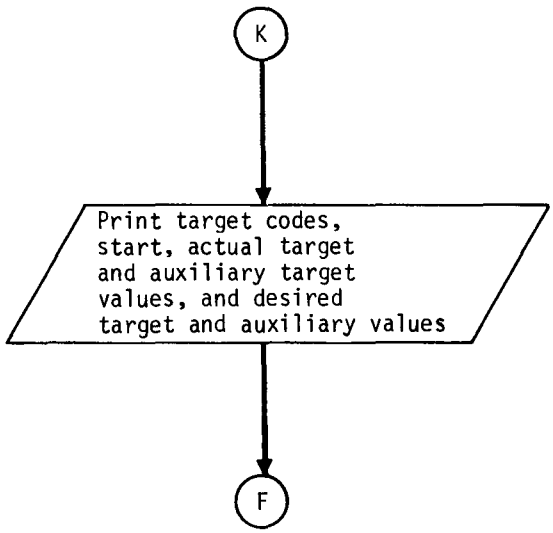

C
Is KEY = 7?
Yes
KAXTAR(I) = 5
IAUX = 1
No
Is KEY = 8?
Yes
KAXTAR(I) = 6
IAUX = 1
No
Is KEY = 13?
Yes
KAXTAR(I) = 5
IAUX = 2
No
Is KEY = 14?
Yes
KAXTAR(I) = 6
IAUX = 2
No
I = ?
< 3
B
= 3
I = 0
ISTØP = 1
Δt = t_T - t_G
D
I = I + 1
KEY = KAXTAR(I)
FAC(1) = FAC(2) = FAC(3) = 1
E


E
Is KEY = 9?
Yes
ISTØP = 2
No
Is KEY = 1, 2, 3or 15?
Yes
ISTØP = 2
Δt = 1.1 (t_T - t_G)
No
Is KEY = 4?
Yes
ISTØP = 3
t = 1.1 (t_T - t_G)
No
Is KAXTAR(I) = 1, 2, 3, 4 or 15
Yes
FAC(I) = WGHTM
No
I = ?
< 3
D
= 3
F
RETURN


A
NØSØI = 1?
Yes
IAUX = 0
No
Is
IAUX ≠ 0
Yes
Yes
Is
ISTOP = 3?
No
= 2
ISTØP = ?
= 3
Call IMPCT with CA state to compute $B_A \cdot T$ and $B_A \cdot R$
1
IAUX = ?
2
Call IMPCT with CA state to compute $B_A \cdot T$, $B_A \cdot R$, i, r_{CA}, t_{CA}, $B_D \cdot T$ and $B_D \cdot R$ (last two only if ITARØ = 2)
Call IMPCT with SOI state to compute $B_A \cdot T$, $B_A \cdot R$, i, r_{CA}, t_{CA}, $B_D \cdot T$ and $B_D \cdot R$ (last two only if ITARØ = 2)
Call IMPCT with SOI state to compute $B_A \cdot T$, $B_A \cdot R$, α_I, δ_I, t_I, $B_D \cdot T$, and $B_D \cdot R$ (last two only if ITARØ = 2)
I = 0
G
I = I + 1
KEY = KAXTAR(I)
H

382-2


15
AAUX(I) = t_{PR}
14
AAUX(I) = α_I
13
AAUX(I) = δ_I
12
AAUX(I) = Z_f
11
AAUX(I) = Y_f
H
10
AAUX(I) = X_f
9
AAUX(I) = a_{SI}
KEY = ?
8
AAUX(I) = r_{CA}
7
AAUX(I) = i
6
AAUX(I) B_AR
5
AAUX(I) = B_AT
4
AAUX(I) = t_{CA}
3
AAUX(I) = t_{CS}
2
AAUX(I) t_{SI}
1
AAUX(I) = t_{PS}
I = ?
< 3
G
=3
Set ATAR = AAUX
Is IAUX = 0?
Yes
J
No
Load ATAR with values for actual targets that differ from auxiliary targets
No
Is ITARØ = 2?
Yes
Load B_DT and B_DR into DAUX array
K


K
Print target codes, start, actual target and auxiliary target values, and desired target and auxiliary values
F

TARPRL Analysis

The position components of a planet are related to its orbital elements a, e, i, Ω, ω, and M through the following set of equations:

$$x = r\left[\cos\Omega\cos(\omega+\nu) - \sin\Omega\sin(\omega+\nu)\cos i\right] \quad (1)$$

$$y = r\left[\sin\Omega\cos(\omega+\nu) + \cos\Omega\sin(\omega+\nu)\cos i\right] \quad (2)$$

$$z = r\sin(\omega+\nu)\sin i \quad (3)$$

$$r = \frac{a(1-e^2)}{1+e\cos\nu} \quad (4)$$

$$\tan\frac{\nu}{2} = \sqrt{\frac{1+e}{1-e}}\ \tan\frac{E}{2} \quad (5)$$

$$M = E - e\sin E \quad (6)$$

We can write equations (1), (2), (3), and (4) symbolically as

$$x_i = f_i(a, e, i, \Omega, \omega, \nu)$$

and equations (5) and (6) as

$$\nu = \nu\ (e, M) \quad .$$

Then the partials of x_i with respect to a, e, i, Ω, ω, and M can be evaluated as follows:

$$\frac{\partial x_i}{\partial a} = \frac{\partial f_i}{\partial a} \quad (7)$$

$$\frac{\partial x_i}{\partial e} = \left(\frac{\partial f_i}{\partial e}\right)_{\nu} + \frac{\partial f_i}{\partial \nu}\cdot\frac{\partial \nu}{\partial e} \quad (8)$$

$$\frac{\partial x_i}{\partial i} = \frac{\partial f_i}{\partial i} \quad (9)$$

$$\frac{\partial x_i}{\partial \Omega} = \frac{\partial f_i}{\partial \Omega} \quad (10)$$

$$\frac{\partial x_i}{\partial \omega} = \frac{\partial f_i}{\partial \omega} \tag{11}$$

$$\frac{\partial x_i}{\partial M} = \frac{\partial f_i}{\partial \nu} \cdot \frac{\partial \nu}{\partial M} \tag{12}$$

Only $\frac{\partial \nu}{\partial e}$ and $\frac{\partial \nu}{\partial M}$ require further consideration before equations (7) through (11) can be used to obtain expressions for the 18 desired partial derivatives.

We obtain $\frac{\partial \nu}{\partial M}$ by first differentiating equation (5) with respect to E and equation (6) with respect to M to obtain

$$\frac{\partial \nu}{\partial E} = \frac{a}{r} \sqrt{1 - e^2}$$

and

$$\frac{\partial E}{\partial M} = \frac{a}{r} \quad .$$

Then

$$\frac{\partial \nu}{\partial M} = \frac{\partial \nu}{\partial E} \cdot \frac{\partial E}{\partial M} = \left(\frac{a}{r}\right)^2 \sqrt{1 - e^2} \quad . \tag{13}$$

We obtain $\frac{\partial \nu}{\partial e}$ by first differentiating equation (6) with respect to e to obtain

$$\frac{\partial M}{\partial e} = - \frac{\sqrt{1 - e^2} \sin \nu}{1 + e \cos \nu} \quad .$$

This result is then combined with equation (13) to yield

$$\frac{\partial \nu}{\partial e} = \frac{\partial \nu}{\partial M} \frac{\partial M}{\partial e} = - \left(\frac{a}{r}\right)^2 \frac{(1 - e^2) \sin \nu}{1 + e \cos \nu} \tag{14}$$

The evaluation of the desired partials can now proceed. The results are summarized below.

a. Partials with respect to a.

$$\frac{\partial x}{\partial a} = \frac{x}{a}$$

$$\frac{\partial y}{\partial a} = \frac{y}{a}$$

$$\frac{\partial z}{\partial a} = \frac{z}{a}$$

b. Partials with respect to e.

$$\frac{\partial x}{\partial e} = \frac{xq}{r} + r\frac{\partial \nu}{\partial e}\left[-\cos\Omega\sin(\omega+\nu) - \sin\Omega\cos(\omega+\nu)\cos i\right]$$

$$\frac{\partial y}{\partial e} = \frac{yq}{r} + r\frac{\partial \nu}{\partial e}\left[-\sin\Omega\sin(\omega+\nu) + \cos\Omega\cos(\omega+\nu)\cos i\right]$$

$$\frac{\partial z}{\partial e} = \frac{zq}{r} + r\frac{\partial \nu}{\partial e}\cos(\omega+\nu)\sin i$$

where $$q = \frac{r}{ae(1-e^2)}\left[r - a - ae^2(1+\sin^2\nu)\right]$$

c. Partials with respect to i.

$$\frac{\partial x}{\partial i} = r\sin\Omega\sin(\omega+\nu)\sin i$$

$$\frac{\partial y}{\partial i} = -r\cos\Omega\sin(\omega+\nu)\sin i$$

$$\frac{\partial z}{\partial i} = r\sin(\omega+\nu)\cos i$$

d. Partials with respect to Ω.

$$\frac{\partial x}{\partial \Omega} = -y$$

$$\frac{\partial y}{\partial \Omega} = x$$

$$\frac{\partial z}{\partial \Omega} = 0$$

e. Partials with respect to ω.

$$\frac{\partial x}{\partial \omega} = r\left[-\cos\Omega\sin(\omega+\nu) - \sin\Omega\cos(\omega+\nu)\cos i\right]$$

$$\frac{\partial y}{\partial \omega} = r\left[-\sin\Omega\sin(\omega+\nu) + \cos\Omega\cos(\omega+\nu)\cos i\right]$$

$$\frac{\partial z}{\partial \omega} = r\cos(\omega+\nu)\sin i$$

f. Partials with respect to M.

$$\frac{\partial x}{\partial M} = \frac{xs}{r} + r\frac{\partial \nu}{\partial M}\left[-\cos\Omega\sin(\omega+\nu) - \sin\Omega\cos(\omega+\nu)\cos i\right]$$

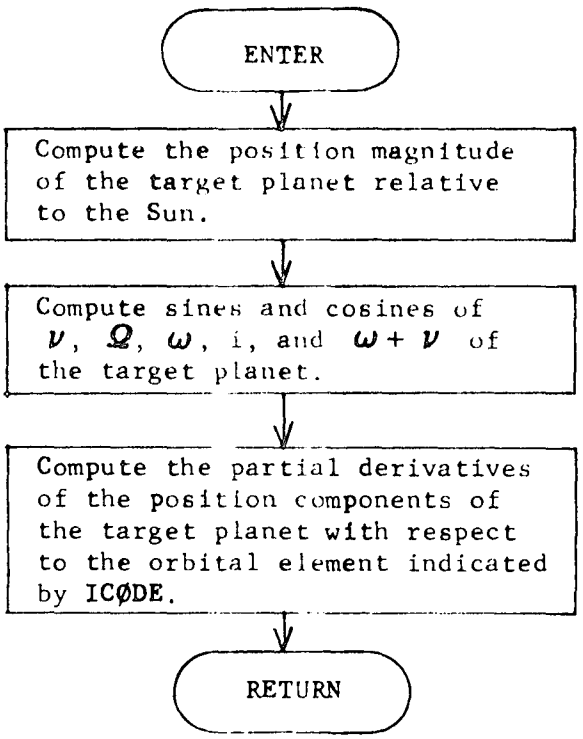
$$\frac{\partial y}{\partial M} = \frac{ys}{r} + r\frac{\partial \nu}{\partial M}\left[-\sin\Omega\sin(\omega+\nu) + \cos\Omega\cos(\omega+\nu)\cos i\right]$$

$$\frac{\partial z}{\partial M} = \frac{zs}{r} + r\frac{\partial \nu}{\partial M}\cos(\omega+\nu)\sin i$$

where $s = \dfrac{ae\sin\nu}{\sqrt{1-e^2}}$

Reference: Battin, R. H.: Astronautical Guidance, McGraw-Hill Book Company, Inc., New York, 1964.

TARPRL Flow Chart



THPØSM Analysis

THPØSM locates by cubic interpolation the minimum of a scalar function f of one variable on the closed interval from 0 to λ (positive), assuming f has a unique point on that interval where its derivative vanishes and that this extreme value is indeed a minimum. The algorithm fits a cubic polynomial through function values at 0, $\alpha\lambda$, and λ as well as through the function's slope at 0. Here $\alpha\lambda$ should be on the open interval from 0 to λ and preferably near the middle. The abscissa of the minimum is then approximately by the abscissa of the corresponding minimum of the cubic.

The coefficients of the approximating cubic can readily be derived once and for all and cast into a form facilitating speedy execution. This approach proves much more economical of machine time than solving for them each pass with a linear system routine as is frequently done in polynomial fitting.

Denote the approximating cubic polynomial by

$$c(X) = a_0X^3 + a_1X^2 + a_2X + a_3. \tag{1}$$

Then clearly

$$a_3 = f(0) \tag{2}$$

and

$$a_2 = f'(0) \tag{3}$$

One also has that

$$f(\lambda) = a_0\lambda^3 + a_1\lambda^2 + f'(0)\lambda + f(0) \tag{4}$$

and

$$f(\alpha\lambda) = a_0\alpha^3\lambda^3 + a_1\alpha^2\lambda^2 + f'(0)\alpha\lambda + f(0). \tag{5}$$

Solving these last two equations for a_0 and a_1 yields

$$a_0 = \frac{1}{\lambda^3\alpha^2}\left[\lambda f'(0)\alpha + f(0)(1+\alpha) + \frac{\alpha^2 f(\lambda) - f(\alpha\lambda)}{1-\alpha}\right] \tag{6}$$

and

$$a_1 = \frac{1}{\lambda^2\alpha^2}\left|\frac{f(\alpha\lambda) - \alpha^3 f(\lambda)}{1-\alpha} - \lambda\alpha(1+\alpha)f'(0) - f(0)(1+\alpha+\alpha^2)\right|. \quad (7)$$

For an extreme value, X_e, one knows that $c'\left(X_e\right) = 0$, that is,

$$3a_0X_e^{\,2} + 2a_1X_e + a_2 = 0. \quad (8)$$

Hence

$$X_{e_{1,2}} = \frac{-a_1 \pm \sqrt{a_1^{\,2} - 3a_0a_2}}{3a_0}.$$

The question remains as to which of these two extrema is a minimum. It is shown in elementary calculus that an extremum is a minimum if, and only if, the second derivative is positive there. Now

$$c''(X) = 6a_0X + 2a_1.$$

Hence

$$c''\left(X_{e_{1,2}}\right) = 6a_0\left[\frac{-a_1 \pm \sqrt{a_1^{\,2} - 3a_0a_2}}{3a_0}\right] + 2a_1$$

$$= \pm 2\sqrt{a_1^{\,2} - 3a_0a_2}.$$

Thus the cubic has its minimum at

$$X_{min} = \frac{-a_1 + \sqrt{a_1^{\,2} - 3a_0a_2}}{3a_0}.$$

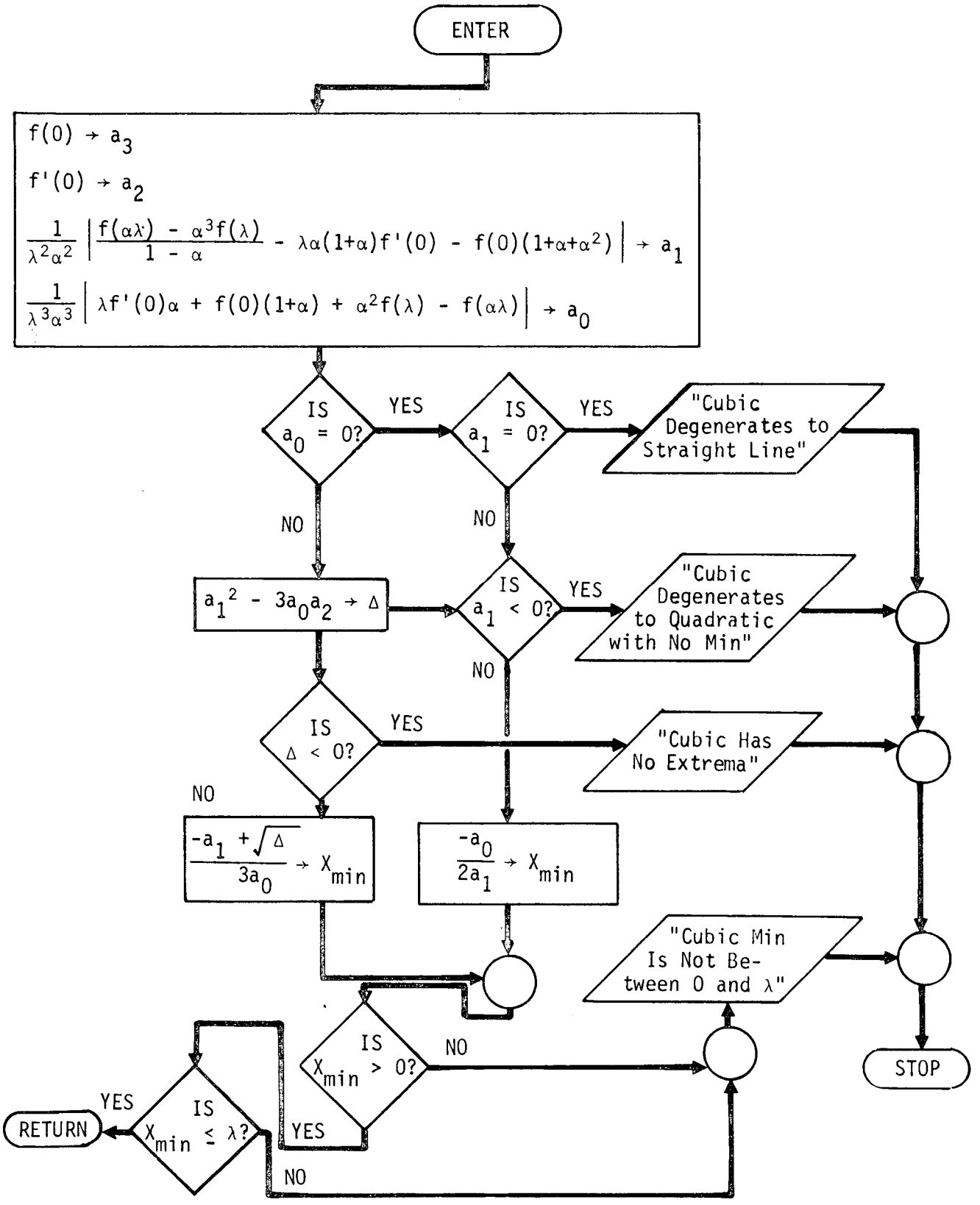
The preceding formula for the minimum will obviously be inadequate when $a_0 = 0$ as is the case when minimizing a quadratic. However it can be shown that

$$\lim_{a_0 \to 0} X_{min} = -\frac{a_2}{2a_1}$$

and hence when $a_0 = 0$ we take

$$X_{min} = -\frac{a_2}{2a_1}$$

THPØSM provides diagnostic printouts and stops execution of the calling program in the three conceivable cases of difficulty: (1) the cubic degenerates to a straight line, (2) the cubic degenerates to a quadratic with no minimum, (3) the cubic is nondegenerate but has no extrema, or (4) the minimum of the cubic does not fall between 0 and λ.


ENTER
$f(0) \rightarrow a_3$
$f'(0) \rightarrow a_2$
$\frac{1}{\lambda^2\alpha^2}\left|\frac{f(\alpha\lambda) - \alpha^3 f(\lambda)}{1-\alpha} - \lambda\alpha(1+\alpha)f'(0) - f(0)(1+\alpha+\alpha^2)\right| \rightarrow a_1$
$\frac{1}{\lambda^3\alpha^3}\left|\lambda f'(0)\alpha + f(0)(1+\alpha) + \alpha^2 f(\lambda) - f(\alpha\lambda)\right| \rightarrow a_0$
IS $a_0 = 0$?
YES
IS $a_1 = 0$?
YES
"Cubic Degenerates to Straight Line"
NO
NO
$a_1^2 - 3a_0a_2 \rightarrow \Delta$
IS $a_1 < 0$?
YES
"Cubic Degenerates to Quadratic with No Min"
NO
IS $\Delta < 0$?
YES
"Cubic Has No Extrema"
NO
$\frac{-a_1 + \sqrt{\Delta}}{3a_0} \rightarrow X_{min}$
$\frac{-a_0}{2a_1} \rightarrow X_{min}$
"Cubic Min Is Not Between 0 and λ"
IS $X_{min} > 0$?
NO
STOP
YES
IS $X_{min} \leq \lambda$?
YES
RETURN
NO

TPPRØP Analysis

The subroutine TPPRØP has the sole responsibility for propagating miniprobes. It is called on to do so in two basic applications. The first is in calculating the miniprobe targeting constraint $\underline{\psi}$ given the release control $\underline{u}$ as repeatedly required in the miss-minimization process (see analysis of TPRTRG). The second is in describing the minimum-miss miniprobe approach trajectories once the optimal release controls have been found. This includes generating impact data for both conic and virtual-mass propagation as well as time histories of the latter. The logic of TPPRØP is considerably complicated by the requirement that it be able to propagate the approach trajectories according to either a high-speed conic model or an accurate virtual-mass n-body integrator.

First consider the problem of calculating $\underline{\psi}$ given $\underline{u}$. For this application of TPPRØP, the miss-minization status flag, IFIN2, is set to 1. In either propagation mode the velocity, $\underline{v}_{iR}$, of the ith miniprobe just after release must be computed first of all. Let v_{BR} be the velocity of the bus at release and ϕ_i be the release roll angle of the ith probe. Let $\underline{U}$, $\underline{V}$ and $\underline{H}$ be the probe release reference vectors defined in the analysis section of TPRTRG. By ϕ_i we mean the angle the release velocity increment of the ith miniprobe makes with the U direction. It should not be confused with the angle the ith probe arm makes with the U direction, which is $\pi/2$ radians smaller. Next define v_T to be the common tangential velocity of the miniprobes at release. Then the velocity of the ith miniprobe at release is

$$\underline{v}_{iR} = v_T \left[\cos \phi_i \underline{U} + \sin \phi_i \underline{V}\right] + \underline{v}_{BR} \qquad i = 1,2,3. \qquad (1)$$

Let α_H and δ_H denote the right ascension and declination, respectively, of the spin axis at release. Then by expressing $\underline{U}$ and $\underline{V}$ in terms of these angles, equation (1) reduces to the following in the planetocentric ecliptic system:

$$\underline{v}_{iR} = v_T \left[\cos \phi_i \begin{pmatrix} \sin \alpha_H \\ -\cos \alpha_H \\ 0 \end{pmatrix} + \sin \phi_i \begin{pmatrix} \cos \alpha_H \sin \delta_H \\ \sin \alpha_H \sin \delta_H \\ -\cos \delta_H \end{pmatrix}\right] + \underline{v}_{BR} \qquad i = 1,2,3 \qquad (2)$$

Simplifying equation (2) yields the computational form of the ith miniprobe velocity increment:

$$\underline{v}_{iR} = v_T \begin{pmatrix} \sin \alpha_H \cos \phi_i + \cos \alpha_H \sin \delta_H \sin \phi_i \\ -\cos \alpha_H \cos \phi_i + \sin \alpha_H \sin \delta_H \sin \phi_i \\ - \cos \delta_H \sin \phi_i \end{pmatrix} + \underline{v}_{BR} \quad (3)$$

$$i = 1,2,3.$$

The sines and cosines of α_H and δ_H necessary in equation (3) are all calculated in a single call to the subroutine SAØCS given the spin-axis orientation flag, ISAØ.

At this point the algorithms for calculating the constraint $\underline{\psi}$ diverge for the two miniprobe propagation models. In the conic propagation mode, signaled by the flag IPRØP set to 1, $\underline{v}_{BR}$ in equation (3) represents the velocity of the equivalent conic release state of the bus (see TPRTRG analysis). Hence, to determine the actual B-plane pierce point coordinates $\left(B_i \cdot T_i\right)_A$ and $\left(B_i \cdot R_i\right)_A$ as well as the parameters $\underline{S}$, $\underline{T}$, $\underline{R}$ and a for the ith miniprobe, TPPRØP simply applies the subroutine STIMP to the state $\left(r_{BR}^T \,\vdots\, v_{iR}^T\right)^T$ where $\underline{r}_{BR}$ denotes the equivalent conic position of the bus (and hence also of the ith probe) at release. Determining the desired B-plane pierce point coordinates, $\left(B_i \cdot T_i\right)_D$ and $\left(B_i \cdot R_i\right)_D$, of the ith probe is complicated by the fact that the miniprobes must be correctly paired with the impact sites. The first miniprobe is targeted to the miniprobe site whose pierce point at the time of the calculation of the initial control estimate in TPRTRG was nearest the bus pierce point. Hence $\left(B_1 \cdot T_1\right)_D$ and $\left(B_1 \cdot R_1\right)_D$ are readily calculated by the appropriate call to DIMPCP, with the right ascension and declination of the target site used in the initial control estimate and the $\underline{S}$ and a of the current miniprobe 1 trajectory. Next TPRTRG computes two sets of desired B-plane coordinate pairs, $\left(B_2 \cdot T_2\right)_D$ and $\left(B_2 \cdot R_2\right)_D$, by applying DIMPCP successively to each of the remaining pairs of miniprobe target sites using in both cases the

$\underline{S}$ and a of the current miniprobe 2 trajectory. With these two sets of desired pierce point coordinate pairs relative to miniprobe 2 now available for comparison, TPPRØP selects the set whose Euclidean distance from the pierce point of miniprobe 2 is the smaller. Finally, $\left(B_3 \cdot T_3\right)_D$ and $\left(B_3 \cdot R_3\right)_D$ are calculated by calling DIMPCP with right ascension and declination of the remaining miniprobe target site and the $\underline{S}$ and a of the current miniprobe 3 trajectory. At this point an approximately inherent in the application of the subroutine DIMPCP to the miniprobes must be noted. The time-varying transformation from planetocentric-ecliptic coordinates to the probe-impact frame required by DIMPCP is held fixed at the time of the bus impact. This approximation is more than adequate for engineering purposes. All of the required B-plane data for the three miniprobes having been assembled, TPPRØP can now calculate the ith and (i+3)rd components of the constraint vector:

$$\psi_i = C_i \left[\left(B_i \cdot T_i\right)_D - \left(B_i \cdot T_i\right)_A\right]$$

$$\psi_{i+3} = C_i \left[\left(B_i \cdot R_i\right)_D - \left(B_i \cdot T_i\right)_A\right] \qquad i = 1,2.3. \qquad (4)$$

Here the C_i's are weighting factors indicating the relative importance of achieving nearby impacts at the respective miniprobe target sites. Finally, the release roll angle for the next miniprobe can be found from that of the current one by applying the addition formulas for the sine and cosine:

$$\sin \phi_{i+1} = \left(\sqrt{3} \sin \phi_i + \cos \phi_i\right) /2 \qquad (5)$$

$$\cos \phi_{i+1} = \left(\cos \phi_i - \sqrt{3} \sin \phi_i\right) /2. \qquad (6)$$

The iterative process is started by noting that ϕ_1 is simply ϕ, the first component of the control vector.

For virtual-mass propagation indicated by IPRØP=2, the general structure of the $\underline{\psi}$ computation remains the same but the interpretation of the contituent state vectors changes. The $\underline{v}_{BR}$ in equation (3) is taken as the velocity of the actual virtual-mass release state. Then VMP is called to integrate the virtual mass state $\left(\underline{r}_{BR}^T \mid \underline{v}_{iR}^T\right)^T$ of the ith miniprobe just after

release to a pseudosphere whose radius is one-tenth that of the actual Laplacian sphere of influence. From this distance inward, conic extrapolation of the current state suffices for engineering calculations. For the actual B-plane pierce-point coordinates, $(B_i \cdot T_i)_A$ and $(B_i \cdot R_i)_A$, as well as the quantities $\underline{S}$, $\underline{T}$, $\underline{R}$ and for the ith miniprobe, TPPRØP again simply makes a single call to STIMP but this time with the virtual-mass state at the pseudosphere rather than the equavalent conic state at release. The remainder of the virtual-vass constraint vector computation including the computation of the desired miniprobe B-plane pierce points via the subroutine DIMPCP proceeds exactly as for the conic model.

Next consider the provisions in TPRTRG for describing the minimum-miss miniprobe approach trajectories. Throughout this application, the miss-minimization status flag, IFIN2, must be set to 2. First in this area TPPRØP must supply the impact data for the miniprobes using the conic propagation model on the conic minimum-miss release controls. This is done by setting the progation mode flag, IPRØP to 1. Then with IFIN2=2, TPPRØP uses the state $\left(\underline{r}_{BR}^T \,\vdots\, \underline{v}_{iR}^T\right)^T$ constructed from the bus equivalent conic release state, the conic minimum-miss release controls, and equation (3) to generate an osculating conic by a call to CAREL. The sine and cosine of the true anomaly and the time at impact are determined by a call to SPHIMP. Then the Cartesian planetocentric ecliptic state is evaluated by a call to CONCAR. From these data the right ascension and declination of the impact site, as well as the time, speed, and flightpath angle at impact for the ith miniprobe can be calculated from the formulas used in computing the same quantities for the bus. These are described in the TPRTRG analysis. The angle of attack, α_i, for the ith miniprobe however requires separate treatment. It is assumed that the longitudinal body axies of each miniprobe remains parallel until impact to the spacecraft spin axis at realse. Thus if v_{iI} represents the velocity of the ith miniprobe at impact,

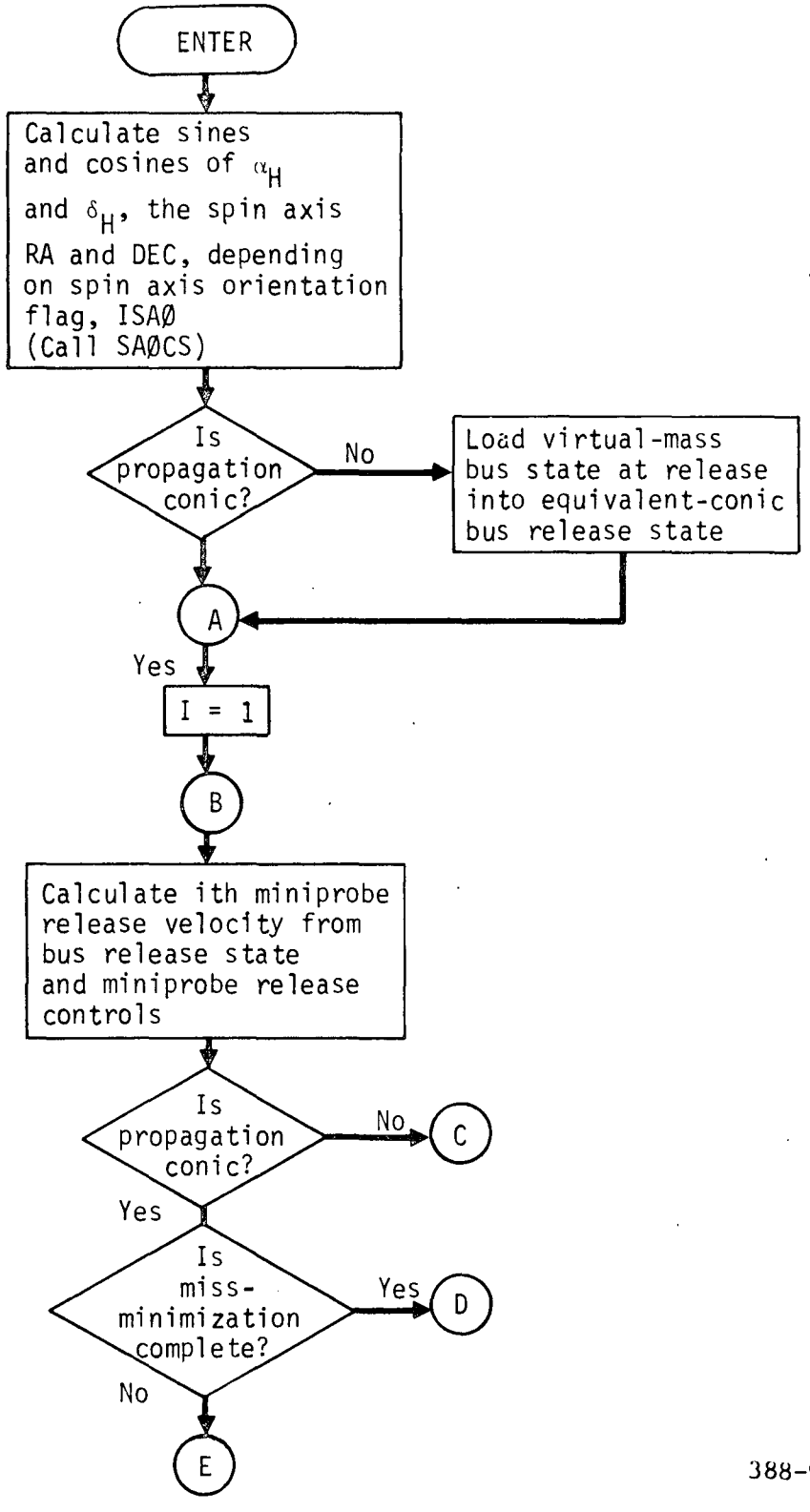
$$\alpha_i = \cos^{-1}\left(\underline{H} \cdot \underline{v}_{iI} \Big/ \|v_{iI}\|\right) \qquad i = 1,2.3. \tag{7}$$

Equation (5) can be expressed in terms of the angles α_H and δ_H as follows:

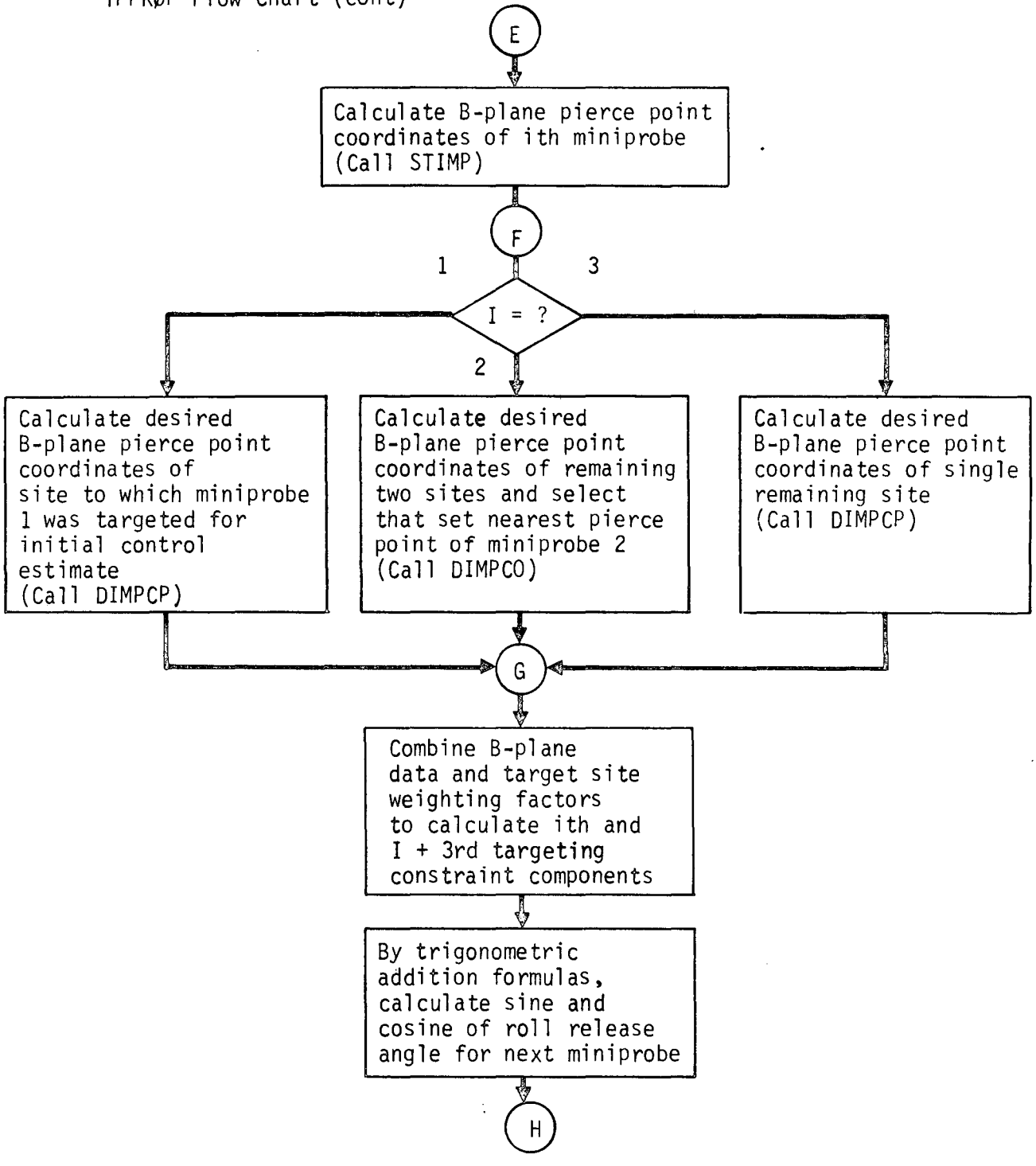
$$\alpha_i = \cos^{-1} \left\{ \left(\cos \delta_H \left[\left(v_{iI} \right)_1 \cos \alpha_H + \left(v_{iI} \right)_2 \sin \alpha_H \right] + \left(v_{iI} \right)_3 \sin \delta_H \right) \Big/ \left\| v_{iI} \right\| \right\} \qquad i = 1,2.3. \qquad (8)$$

The second trajectory-descriptive function of TPPRØP is calculating the virtual-mass miniprobe approach-trajectory time histories and impact data. This is always done for the conic minimum-miss release controls and also for the virtual-mass controls whenever these are calculated. The propagation flag, IPRØP is simply set to 2 and the heading "Miniprobe I Minimum-Miss Approach Trajectory" is printed. Then with IFIN2=2, the state $\left(\underline{r}_{BR}^T \mid \underline{v}_{iR}^T \right)^T$ produced from the bus actual virtual-mass release state, the appropriate minimum-miss release controls (conic or virtual-mass as the case may be), and equation (3) is integrated all the way to impact by VMP with its print flag on and its print increments set at 5 days and 100 integration steps. In this simple manner the miniprobe approach time histories are provided. Finally, the impact data for the virtual-mass ith miniprobe trajectory are calculated by the same steps as for the conic case except that the actual virtual-mass miniprobe impact state rather than the equivalent conic miniprobe release state is used in generating the osculating conic via the call to CAREL.

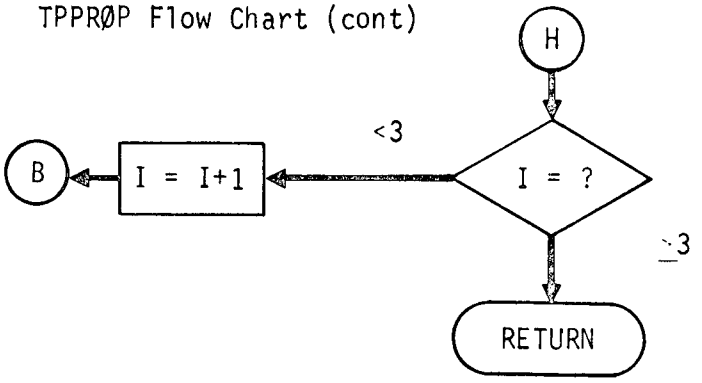
TPPRØP Flow Chart



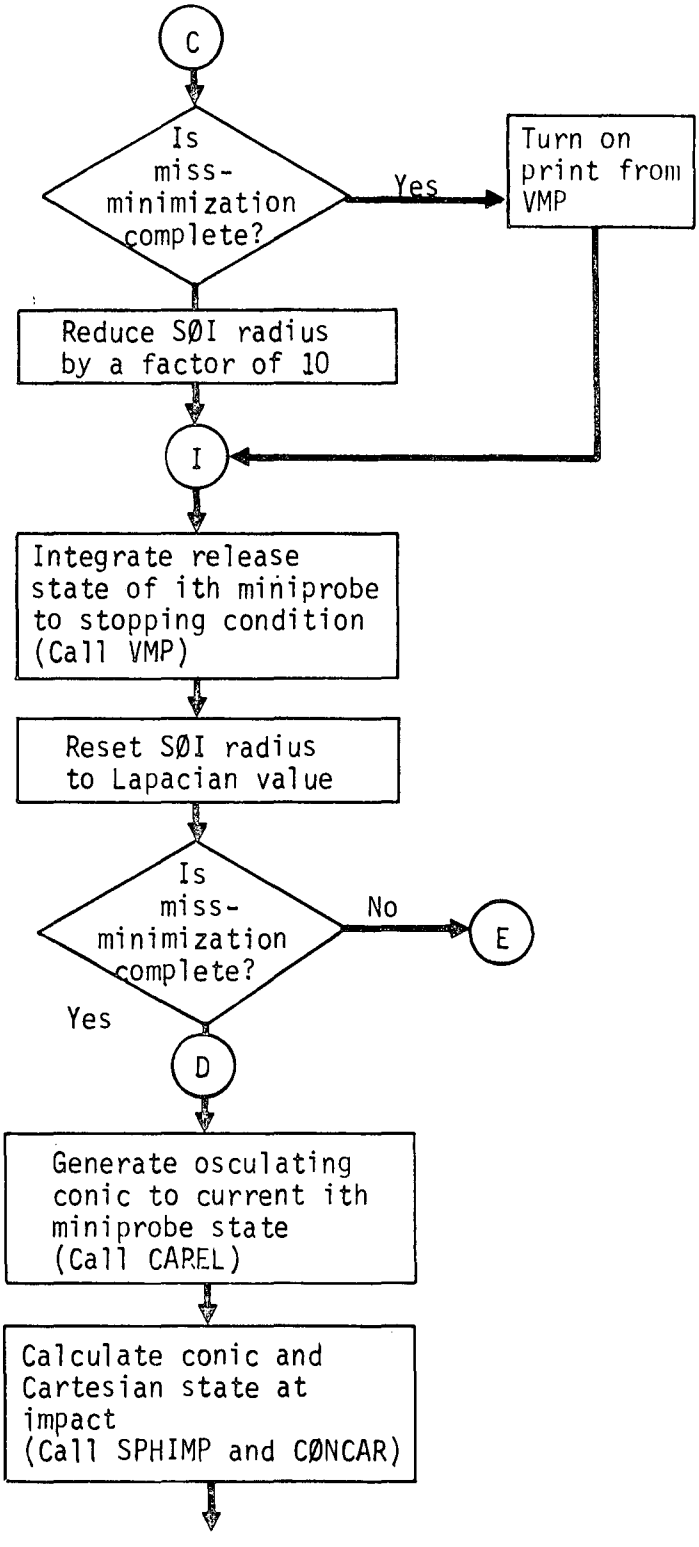
TPPRØP Flow Chart (cont)



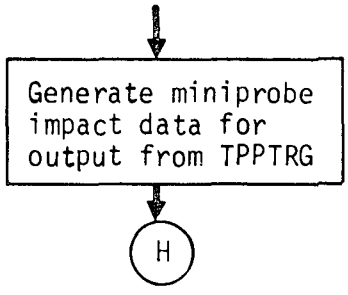
TPPRØP Flow Chart (cont)



TPPRØP Flow Chart (cont)



TPPRØP Flow Chart (concl)



TPRTRG Analysis

TPRTRG is the executive routine directing multiprobe targeting. To do so it must accomplish four basic tasks: (1) process miniprobe targeting input data, (2) generate initial estimates for the release controls, (3) apply the least-squares routine, GAUSLS, to minimize the miniprobe miss index, and (4) use the miniprobe propagating routine, TPPROP, to generate minimum-miss approach trajectories and impact data. Each of these objectives, as well as the routine's printout, is discussed in the following paragraphs.

The processing of the miniprobe targeting input involves three major aspects. First an equivalence must be set up between the multipurpose targeting variables of NØMINAL and the mnemonic symbolism of TPRTRG. Second, a conic miniprobe release state must be determined that is equivalent to the virtual-mass release state for use in the high-speed conic miniprobe propagation model. By equivalent we mean that although one state falls on a conic and the other on a virtual-mass trajectory, both states occur at the same absolute time and the two trajectories are tangent at impact. To generate the equivalent conic release state, the subroutine VMP is called to propagate the bus state from release to impact. Before doing this, however, the VMP trajectory condition flags are stored for subsequent restoration after exiting TPRTRG. The virtual-mass impact state is then fit with an osculating conic by a call to CAREL. The osculating conic is then propagated back to the actual release time via calls to the subroutines HPOST and CONCAR. Using this conic release state, a set of minimum-miss controls for the conic model can now readily be determined. The third phase of processing miniprobe input data, namely setting up the several targeting options, is next begun. First the coordinate transformation matrix, C, from the planetocentric inertial ecliptic coordinates to the Cartesian frame in which the miniprobe impact sites are specified is generated by a call to either the subroutine SUBSOL or PECEQ, depending on the state of the coordinate-system flag IPCSK. The spin axis orientation mode desired is stored in the flag ISAØ. It is used by TPRTRG in devising an initial control estimate and by TPPRØP in all miniprobe propagations. Finally, the desired miniprobe propagation mode is stored in the flag IPRØPI. This variable is used solely by TPRTRG in deciding on which propagation mode to request from TPPRØP via the common flag IPRØP.

TPRTRG next deals with the problem of generating an initial estimate of the control vector. First consider the problem of estimating the conic-model minimum-miss controls. Initial estimates for two of the controls (the spin axis orientation angles) depend of course on the spin axis orientation mode. In three of the four possible modes, the inertial-ecliptic declination and right ascension of the spin axis are fixed rather than free controls. Hence no initial estimates for them need be provided. In the remaining mode, however, both of these controls are free, and the initial estimates provided for them are simply those that bring the spin axis into coincidence with the spacecraft velocity vector at release. This orientation was chosen for the initial estimate since it produces the widest distribution of miniprobe entry sites for a given combination of the remaining two release controls. Regardless of the orientation mode, the spin axis pointing direction is specified throughout miniprobe targeting by the sines and cosines of its ecliptic declination and right ascension. These trigonometric functions are always calculated by the subroutine SAØCS from the appropriate components of the control vector and the spin axis orientation flag. Since the spin axis pointing direction is necessary in initially estimating the other two controls, TPRTRG must call SAØCS to obtain the above trigonometric characterization.

Initial estimates for the roll release angle, ϕ, of the first probe and the tangential velocity at release, v_T, can then be generated by merely targeting the first miniprobe to the miniprobe target site nearest the impact point for the bus on the trajectory existing at release. First, the B-plane pierce point of the bus is calculated by calling the subroutine STIMP with the bus impact state. Then the desired B-plane pierce point corresponding to each of the three miniprobe target sites is computed through a call to DIMPCP. It must be noted that this calculation is only approximate since it assumes all of the miniprobes have the same $\underline{S}$ and impact time as the bus. Nonetheless, the accuracy is more than sufficient for engineering purposes. Next the pierce points of the desired miniprobe impact sites are compared to find the one nearest that of the bus. The release velocity increment perpendicular to the bus spin axis that would target the first miniprobe is then approximated by a single Newton-Raphson step. Let $B_B \cdot T$, $B_B \cdot R$, $B_1 \cdot T$, and $B_1 \cdot R$ denote the B-plane pierce point components corresponding to the bus and the desired miniprobe impact sites, respectively. Define a constraint vector as

$$\psi = \begin{pmatrix} B_1 \cdot T - B_B \cdot T \\ B_1 \cdot R - B_B \cdot R \end{pmatrix} \tag{1}$$

Let $\underline{H}$ denote a unit vector in the direction of the spin axis of the spacecraft. Using $\underline{H}$, define $\underline{U}$ and $\underline{V}$ as

$$\underline{U} = \underline{H} \times \underline{Z}_{ec} \Big/ || \underline{H} \times \underline{Z}_{ec} || \tag{2}$$

$$\underline{V} = \underline{H} \times \underline{U} \tag{3}$$

where $\underline{Z}_{ec}$ is the inertial ecliptic pole vector. Then a convenient probe-release Cartesian frame is defined by the triple $\underline{U}$-$\underline{V}$-$\underline{H}$. Let Δv_U and Δu_V denote the components of the release velocity increment in the $\underline{U}$ and $\underline{V}$ directions, respectively.

Then the control vector is given by

$$\underline{X} = \begin{pmatrix} \Delta v_U \\ \Delta v_V \end{pmatrix}. \tag{4}$$

Let J denote the Jacobian matrix of $\underline{\psi}$ with respect to $\underline{X}$; i.e.,

$$Jij = \frac{\partial \psi_i}{\partial Xj} \qquad \begin{matrix} i = 1,2 \\ j = 1,2. \end{matrix} \tag{5}$$

One Newton-Raphson step then approximates the targeting control vector as

$$\underline{X} = J^{-1} \, \underline{\psi}. \tag{6}$$

TPRTRG computes the Jacobian matrix by divided differencing. Let v_{BR} represent the equivalent conic planetocentric velocity of the bus at release. Let $\delta_{U} \underline{v}_{BR}$ and $\delta_{V} \underline{u}_{BR}$ denote the perturbations in $\underline{v}_{BR}$ caused by a velocity increment of magnitude δv in the $\underline{U}$ and $\underline{V}$ directions, respectively. Then clearly

$$\delta_{U} \underline{v}_{BR} = \delta v \underline{U} \tag{7}$$

$$\delta_{V} \underline{v}_{BR} = \delta v \underline{V}. \tag{8}$$

Let α_H and δ_H be the right ascension and declination of the spin axis, respectively. By expressing the definitions of $\underline{U}$ and $\underline{V}$ in terms of these angles, equations (7) and (8) can be expressed in the planetocentic ecliptic frame as

$$\delta_{\underline{U}}\underline{v}_{BR} = \delta v \ (\sin \alpha, -\cos \alpha, 0)^T \tag{9}$$

$$\delta_{\underline{V}}\underline{v}_{BR} = \delta v \ (\cos \alpha \sin \delta, \sin \alpha \sin \delta, -\cos \delta)^T. \tag{10}$$

Let $\underline{r}_{BR}$ be the equivalent conic planetocentric position of the spacecraft at release. Then by applying the subroutine STIMP consecutively to the states $\left(\underline{r}_{BR}, \underline{v}_{BR} + \delta_U v_{BR}\right)^T$ and $\left(\underline{r}_{BR}, \underline{v}_{BR} + \delta_{\underline{V}}\underline{v}_{BR}\right)^T$, the perturbed state vectors $\delta_{\underline{U}}\underline{\psi}$ and $\delta_{\underline{V}}\underline{\psi}$ can be generated. TPRTRG then approximates J as $\left(\delta_{\underline{U}}\underline{\psi} \mid \delta_{\underline{V}}\underline{\psi}\right)/\delta v$. Having approximated Δv_U and Δv_V, the roll release angle of miniprobe 1 and the tangential velocity at release are readily calculated from the formulae

$$v_T = \sqrt{\Delta v_U^2 + \Delta v_V^2} \tag{11}$$

$$\phi = \tan^{-1}\left(\Delta v_V / \Delta v_U\right). \tag{12}$$

It must be noted that ϕ represents the angle the velocity increment of the first miniprobe makes with the U axis. It should not be confused with the angle between the first probe arm and the U direction, which is $\pi/2$ radians less than ϕ.

Consider next the initial control estimate for virtual-mass minimum-miss controls. TPRTRG simply uses the minimum-miss conic control for this estimate. Hence, irrespective of the desired miniprobe propagation mode indicated by IPRØPI, the conic controls are first found. Then if only the conic controls are desired, a return is made to the calling program; otherwise the least-squares algorithm is repeated with virtual-mass rather than conic miniprobe propagation.

The third basic task of TPRTRG, namely using the subroutine GAUSLS to calculate the minimum-miss index controls, requires some explanation. First the four miniprobe release controls must be

identified. They are simply ϕ, v_T, δ_H, and α_H (see Fig. 1). Thus if $\underline{u}$ denotes the control vector, then

$$\underline{u} = \begin{pmatrix} \phi \\ v_T \\ \delta_H \\ \alpha_H \end{pmatrix}. \tag{13}$$

By measuring the angles ϕ, α_H, and δ_H in radians and the velocity v_T in decameters/s all of the components of $\underline{u}$ fall in the range from 0.1 to 10 as required by the subroutine GAUSLS when Jacobian matrices are generated by a uniform control perturbation of 10^{-5}.

Let $\left(B_i \cdot T_i\right)_A$ and $\left(B_i \cdot R_i\right)_A$ denote the actual B-plane pierce point coordinates of the ith miniprobe when the release control $\underline{u}$ is applied. Let $\left(B_i \cdot T_i\right)_D$ and $\left(B_i \cdot R_i\right)_D$ represent the desired B-plane pierce point coordinates of the ith miniprobe based on the actual $\underline{S}$ and the energy of that miniprobe. The six components of the constraint vector are then given as

$$\psi_i = C_i \left[\left(B_i \cdot T_i\right)_A - \left(B_i \cdot T_i\right)_D\right] \qquad i = 1,2,3. \tag{14}$$

$$\psi_{i+3} = C_i \left[\left(B_i \cdot R_i\right)_A - \left(B_i \cdot R_i\right)_A\right] \tag{15}$$

Here the C_i's are weighting factors input by the user to indicate the relative importance of achieving nearby impacts at the various miniprobe target sites. The subroutine TPPRØP is always used to calculate $\underline{\psi}$ given $\underline{u}$ for whichever miniprobe propagation mode is specified by the flag IRPØP. Thus GAUSLS can be called to carry out the entire least-squares process of minimizing the miss-index $y = \underline{\psi}^T\underline{\psi}$ for either propagation mode once the initial control estimate uo and the corresponding constraint $\underline{\psi}$ ($\underline{uo}$) have been calculated. If the least-squares routine should fail to converge, the universal NØMNAL trouble flag, KWITT, is set to 1 to cause execution to terminate on return to the main program, and a return is made to the calling program, GIDANS.

The final responsibility of TPRTRG is to calculate the n-body miniprobe approach trajectory time histories and impact data for the bus and the miniprobes using the minimum-miss release controls corresponding to both the conic and virtual-mass propagation modes. The impact data for the bus are calculated in TPRTRG itself. Let $\underline{r}_{BI}$ and $\underline{\rho}_{BI}$ denote the impact position vectors in the planetocentric ecliptic and probe-impact coordinate frames. The $\underline{\rho}_{BI}$ is calculated from $\underline{r}_{BI}$, which is available from the virtual-mass propagation, as

$$\underline{\rho}_{BI} = C\underline{r}_{BI}. \tag{16}$$

The right ascension, α_B, and declination, δ_B, of the bus impact site relative to the probe-impact frame are then readily calculated as

$$\alpha_B = \tan^{-1} \left(\rho_{BI}\right)_2 \Big/ \left(\rho_{BI}\right)_1 \tag{17}$$

$$\delta_B = \sin^{-1} \left(\rho_{BI}\right)_3. \tag{18}$$

The flightpath angle, $\gamma_{BI} = \tan^{-1} \left(e\, r_I \sin \theta_I / p\right)$ (19)

where r_I is the radius of the impact sphere, e is the eccentricity, p is the semilatus rectum and θ_I is the true anomaly at impact. All of these conic dements refer to osculating hyperbola at impact. The magnitude of the bus impact velocity is calculated as

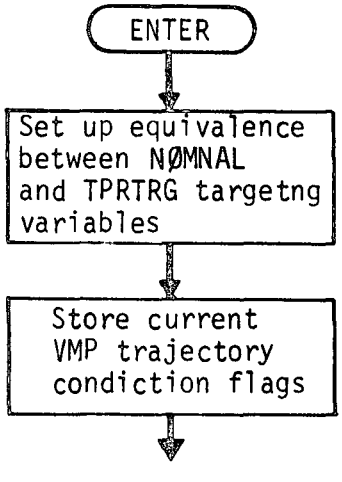
$$v_{BI} = \sqrt{u\left(2/r_I - 1/a\right)} \tag{20}$$

where a is the semimajor axis of the osculating conic, and u is the gravitational constant of the planet.

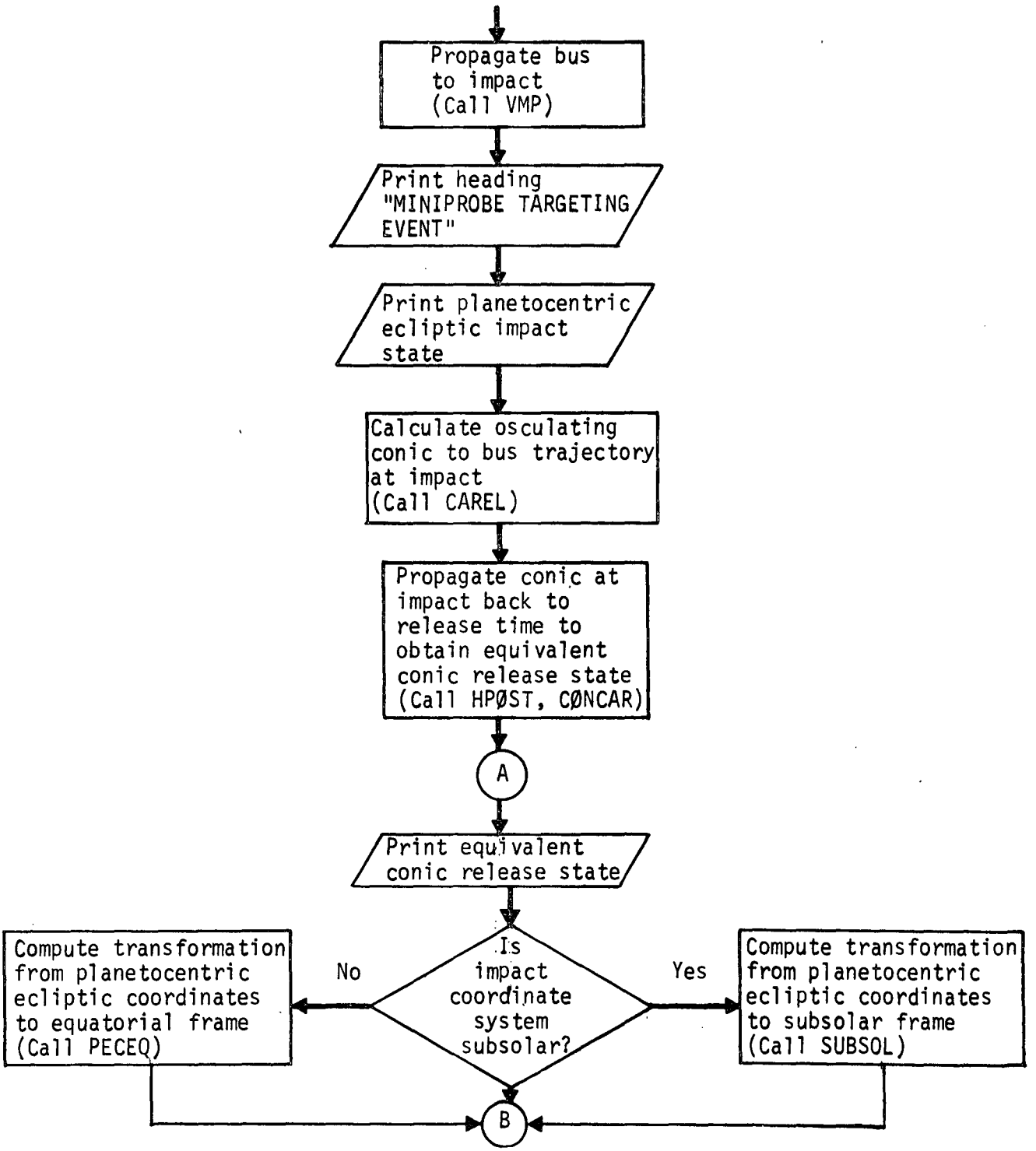
For the miniprobes, both the impact data and the virtual-mass approach trajectory time histories are generated by a call to TPPRØP with the least-squares status flag IFIN2 set to indicate that the miss-minimizing procedure is complete. When the least-squares algorithm is performed using conic miniprobe propagation, impact data are computed for both the conic and virtual-mass models.

The printout from TPRTRG is designed to meet two objectives: (1) to completely describe the minimum-miss miniprobe approach trajectories, and (2) to reveal any errors in the minimum-miss release controls caused by improper use of the program. To identify the type of nolinear guidance event, the heading "Miniprobe Targeting Event" is printed first of all. Next pertient data at release are printed. These consist of the planetocentric bus impact state and the equivalent conic release state as previously described. Then the printout from the miss-minimizing algorithm GAUSLS is provided in its entirety. After the minimum-miss release controls are found, they are printed out with a phrase indicating whether they correspond to the conic or virtual-mass model. If conic miniprobe propagation was used for the miss minimization, the conic-model probe impact data are printed. These include right ascension and declination of the impact point, together with time, velocity, and flightpath angle at impact for each of the miniprobes as well as the bus (numbered as probe number 0 in the printout). The bus data provided here are actually based on the initial virtual-mass propagation from the release state. The miniprobe data also contain the angles of attach, assuming the minprobe longitudinal body axes reamin parallel to the spacecraft spin axis at release. Next the time histories of the miniprobe minimum-miss approach trajectories are printed in succession from the subroutine VMP with print intervals of 5 days and 100 integration steps. Finally, the virtual-mass model probe impact information is printed. It is identical in content to the conic impact data except that the information is now based on virtual-mass propagation from the release state.

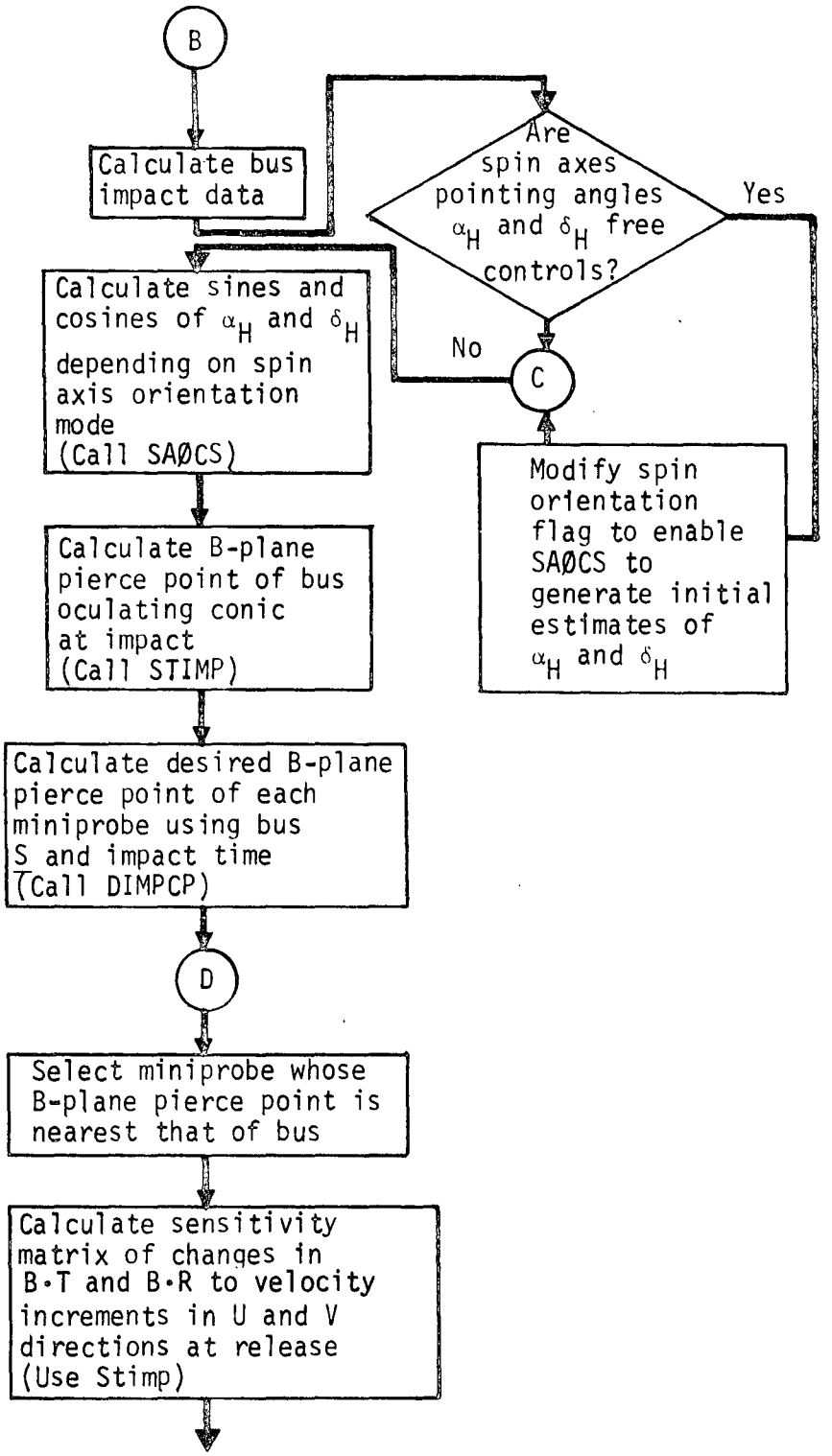
TPRTRG Flow Chart



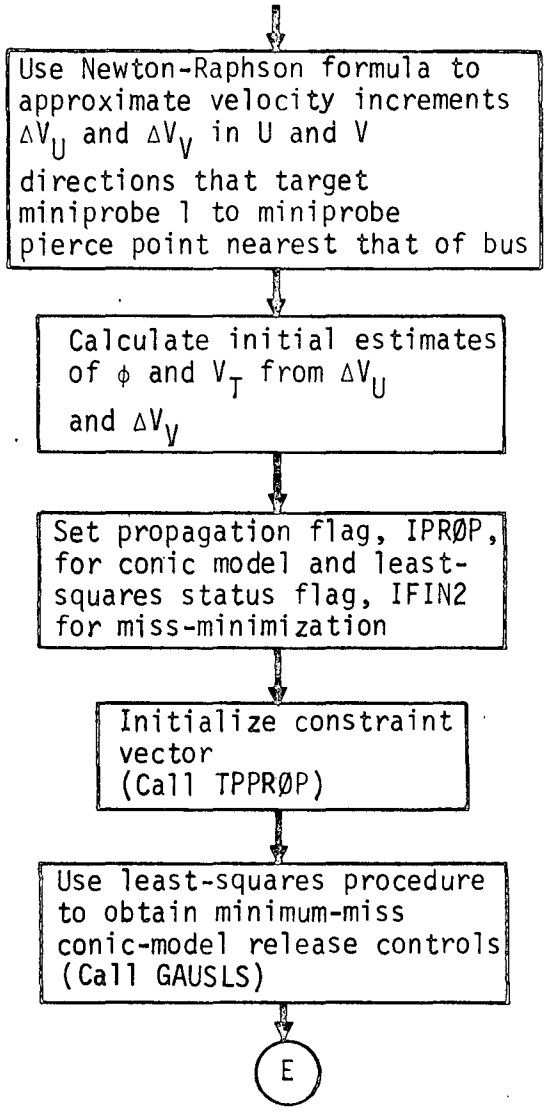
TPRTRG Flow Chart (cont)



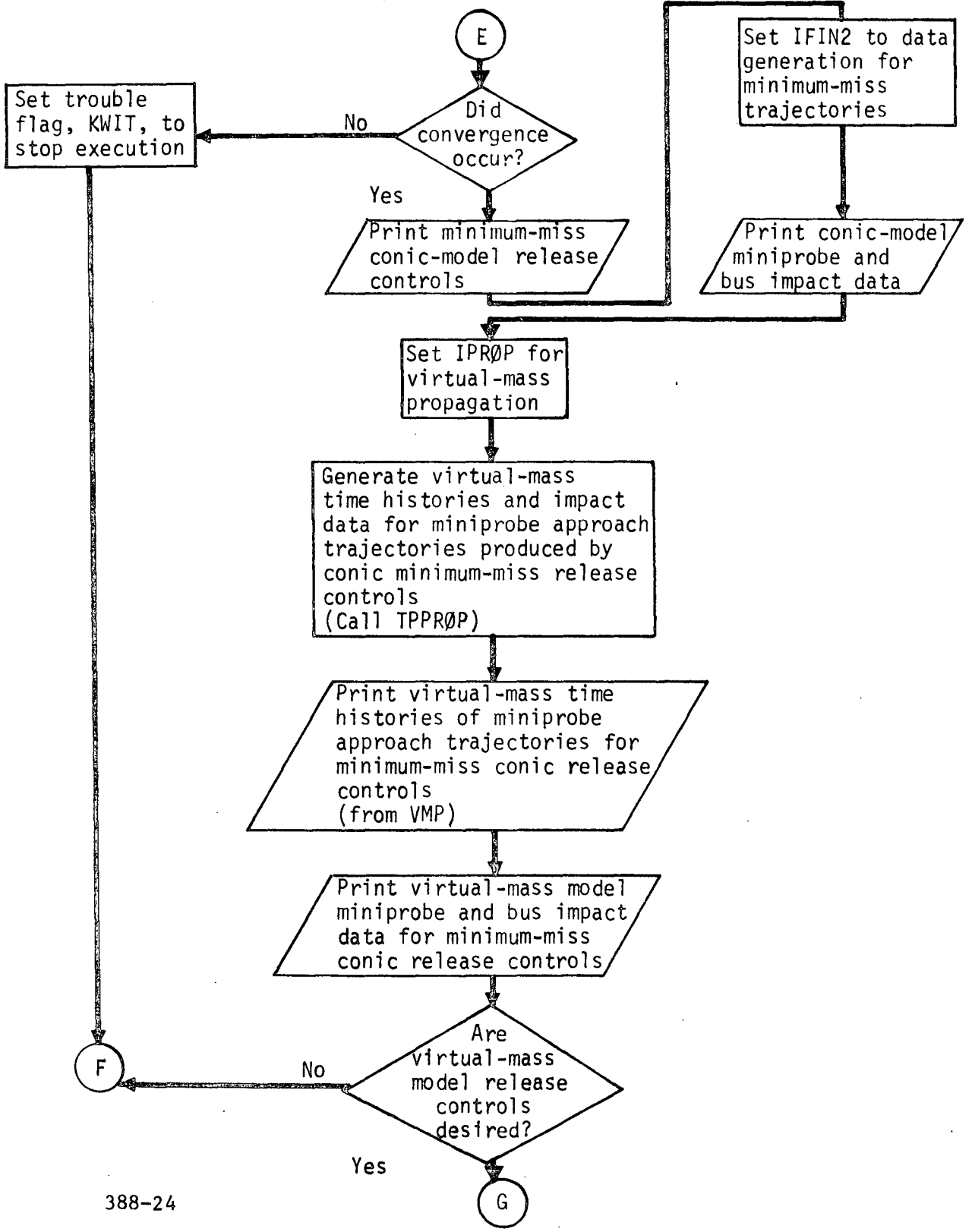
TPRTRG Flow Chart (cont)



TPRTRG Flow Chart (cont)

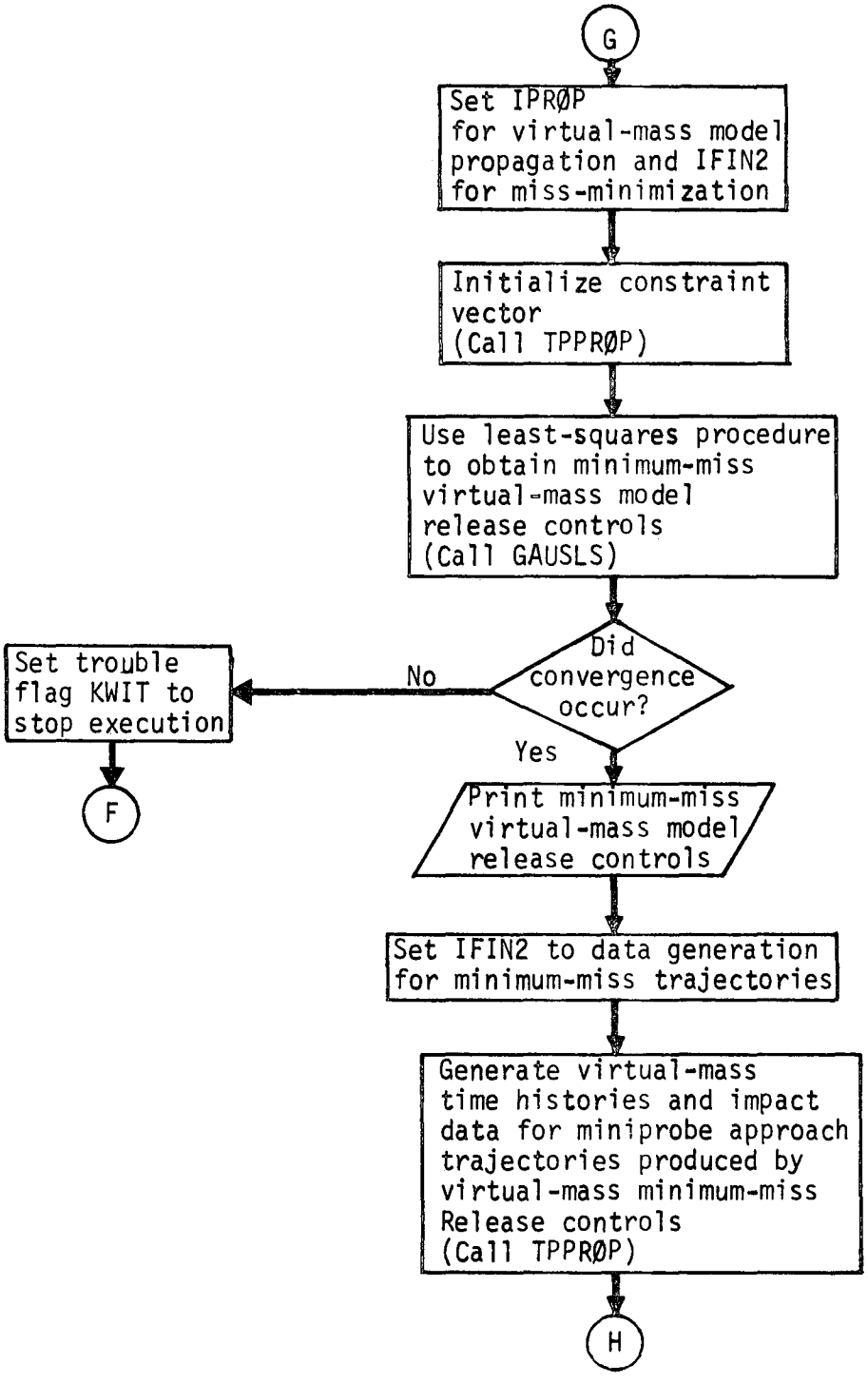


TPRTRG Flow Chart (cont)

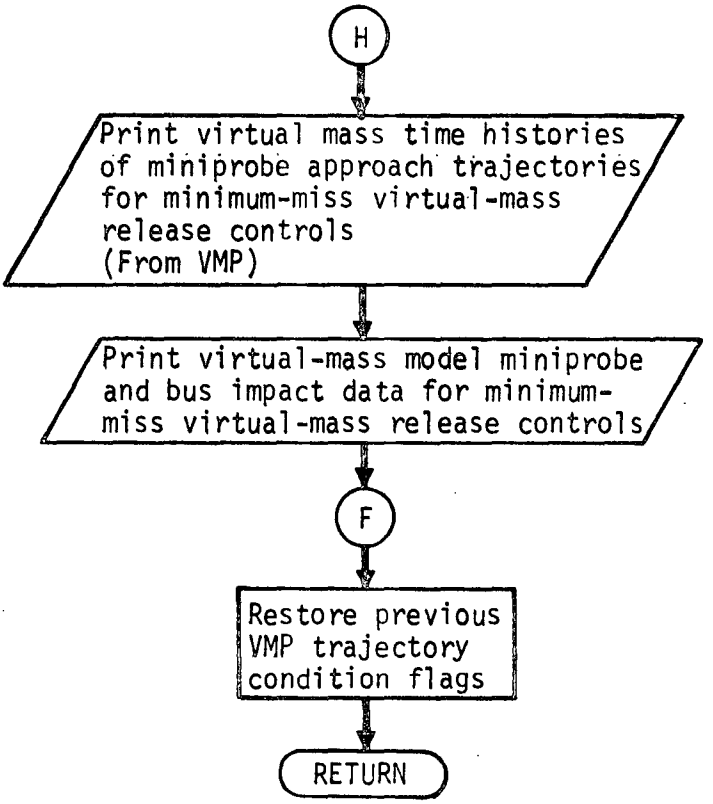


388-24

TPRTRG Flow Chart (cont)



TPRTRG Flow Chart (concl)



Key to Symbols

α	- right ascension of spin axis
δ	- declination of spin axis
ϕ	- roll release angle of first miniprobe
$\underline{V}_{T_i}$	- tangential release velocity of ith miniprobe
$\underline{H}$	- spacecraft spin-axis unit vector
$\underline{U}$	- $\underline{H}\times\underline{Z}_{ec}/\lvert\underline{H}\times\underline{Z}_{ec}\rvert$
$\underline{V}$	- $\underline{H}\times\underline{U}$
$\underline{X}_{ec}$, $\underline{Y}_{ec}$, $\underline{Z}_{ec}$	- inertial ecliptic coordinate-axis unit vectors

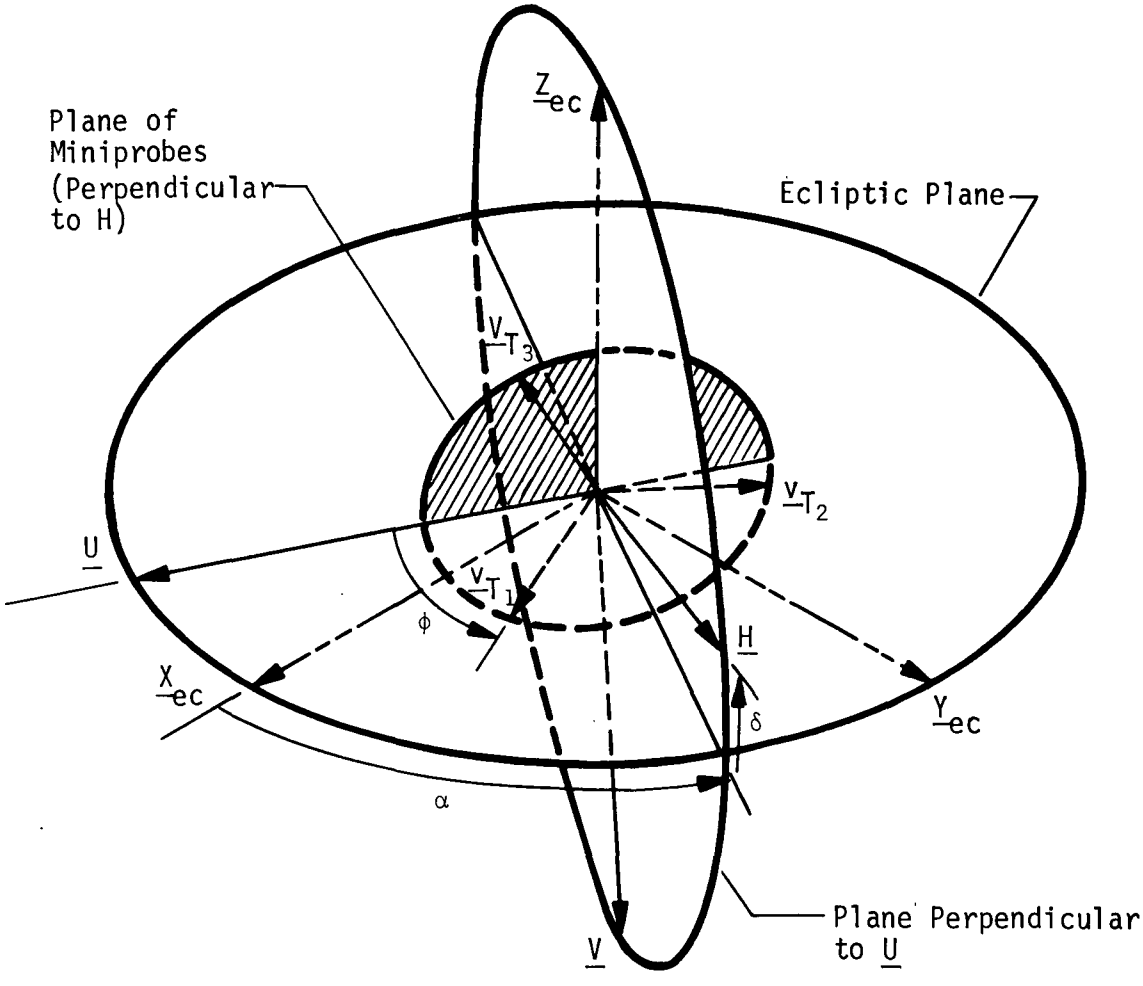


Figure 1 Miniprobe Release Geometry

TRAKM Analysis

The linearized observation equation can be written as

$$y = Hx + Mx_s + Gu + Lv + Nw$$

where y is the observable, x is the spacecraft state, and x_s, u, v, and w are solve-for, dynamic consider, measurement consider, and ignore parameter vectors, respectively. The function of subroutine TRAKM is to compute the observation matrix partitions H, M, G, L, and N, which indicate the sensitivity of the observable y to changes in x, x_s, u, v, and w, respectively, in the error analysis/generalized covariance analysis program. The matrix N is computed only for a generalized covariance analysis.

Except for computation of the ignore parameter observation matrix partition N, TRAKM is equivalent to subroutine TRAKS, which is used in the simulation program. See subroutine TRAKS for further analytical details and a flow chart.

TRAKS Analysis

Subroutine TRAKS performs two functions in the simulation mode. The first function, which corresponds to IØBS = 0, is to compute all observation matrix partitions for the measurement type indicated by ITRK. The second function, which corresponds to IØBS $\neq$ 0, is to compute the measurement itself. If IØBS = 1, TRAKS computes the measurement corresponding to the most recent nominal spacecraft state. If IØBS = 2, TRAKS computes the measurement corresponding to the actual spacecraft state, and, if the measurement is a range or range-rate measurement, to the actual tracking station locations. The number of rows, NR, in the measurement and the observation matrix partitions is also computed.

A general measurement has form

$$\vec{Y} = \vec{Y}\left(\vec{X}, \vec{p}, t\right)$$

where $\vec{X}$ is the spacecraft position/velocity state at time t and $\vec{p}$ is a vector of parameters. This equation can be linearized about nominal $\vec{X}$ and $\vec{p}$ to obtain

$$\delta\vec{Y} = \left(\frac{\partial\vec{Y}}{\partial\vec{X}}\right)^{*} \delta\vec{X} + \left(\frac{\partial\vec{Y}}{\partial\vec{p}}\right)^{*} \delta\vec{p}$$

where $(\ )^*$ indicates matrices are evaluated at the nominal condition. This perturbation equation can be rewritten as

$$\delta\vec{Y} = H\,\delta\vec{X} + M\,\delta\vec{x}_s + G\,\delta\vec{u} + L\,\delta\vec{v}$$

where $H = \left(\frac{\partial\vec{Y}}{\partial\vec{X}}\right)^{*}$, and $\left(\frac{\partial\vec{Y}}{\partial\vec{p}}\right)^{*}$ is distributed among the M, G, and L partitions to correspond to the partition of the parameter vector $\delta\vec{p}$ into solve-for parameters $\delta\vec{x}_s$, dynamic consider parameters $\delta\vec{u}$, and measurement consider parameters $\delta\vec{v}$.

In the remainder of this section the measurement equation and all partial derivatives required to construct the H, M, G, and L observation matrix partitions will be summarized for each measurement type.

A. Range measurement ρ.

A range measurement has form

$$\rho = \rho\left(\vec{X}, R, \theta, \phi, t\right)$$

where R, θ, and ϕ are the radius, latitude, and longitude of the relevant tracking station.

More explicitly,

$$\rho = \left[(X - X_E - X_S)^2 + (Y - Y_E - Y_S)^2 + (Z - Z_E - Z_S)^2 \right]^{\frac{1}{2}}$$

where X, Y, Z = inertial position components of spacecraft
X_E, Y_E, Z_E = inertial position components of Earth
X_S, Y_S, Z_S = station position components relative to Earth.

X_S, Y_S, and Z_S are related to R, θ, and ϕ as follows:

$$X_S = R \cos\theta \cos G$$

$$Y_S = R \cos\theta \cos\epsilon \sin G + R \sin\theta \sin\epsilon$$

$$Z_S = -R \cos\theta \sin\epsilon \sin G + R \sin\theta \cos\epsilon$$

where ϵ is the obliquity of the Earth, and

$$G = \phi + GHA$$

where GHA is the Greenwich hour angle at time t.

Partials of ρ with respect to spacecraft state are given by

$$\frac{\partial \rho}{\partial X} = \frac{1}{\rho}(X - X_E - X_S) \qquad \frac{\partial \rho}{\partial \dot{X}} = 0$$

$$\frac{\partial \rho}{\partial Y} = \frac{1}{\rho}(Y - Y_E - Y_S) \qquad \frac{\partial \rho}{\partial \dot{Y}} = 0$$

$$\frac{\partial \rho}{\partial Z} = \frac{1}{\rho}(Z - Z_E - Z_S) \qquad \frac{\partial \rho}{\partial \dot{Z}} = 0$$

Partials of ρ with respect to R, θ, and ϕ are given by

$$\frac{\partial \rho}{\partial R} = \frac{\partial \rho}{\partial X_S} \cdot \frac{\partial X_S}{\partial R} + \frac{\partial \rho}{\partial Y_S} \cdot \frac{\partial Y_S}{\partial R} + \frac{\partial \rho}{\partial Z_S} \cdot \frac{\partial Z_S}{\partial R}$$

$$\frac{\partial \rho}{\partial \theta} = \frac{\partial \rho}{\partial X_S} \cdot \frac{\partial X_S}{\partial \theta} + \frac{\partial \rho}{\partial Y_S} \cdot \frac{\partial Y_S}{\partial \theta} + \frac{\partial \rho}{\partial Z_S} \cdot \frac{\partial Z_S}{\partial \theta}$$

$$\frac{\partial \rho}{\partial \phi} = \frac{\partial \rho}{\partial X_S} \cdot \frac{\partial X_S}{\partial \phi} + \frac{\partial \rho}{\partial Y_S} \cdot \frac{\partial Y_S}{\partial \phi} + \frac{\partial \rho}{\partial Z_S} \cdot \frac{\partial Z_S}{\partial \phi}$$

where

$$\frac{\partial \rho}{\partial X_S} = -\frac{\partial \rho}{\partial X}, \quad \frac{\partial \rho}{\partial Y_S} = -\frac{\partial \rho}{\partial Y}, \quad \frac{\partial \rho}{\partial Z_S} = -\frac{\partial \rho}{\partial Z}$$

and the negatives of the partials of X_S, Y_S, and Z_S with respect to R, θ, and ϕ are summarized in the subroutine STAPRL analysis.

B. Range-rate measurement $\dot{\rho}$.

A range-rate measurement has form

$$\dot{\rho} = \dot{\rho}(\vec{X}, R, \theta, \phi, t)$$

where all arguments have been defined previously. More explicitly,

$$\dot{\rho} = \frac{\rho_x \dot{\rho}_x + \rho_y \dot{\rho}_y + \rho_z \dot{\rho}_z}{\rho}$$

where
$$\rho_x = X - X_E - X_S \qquad \dot{\rho}_x = \dot{X} - \dot{X}_E - \dot{X}_S$$
$$\rho_y = Y - Y_E - Y_S \qquad \dot{\rho}_y = \dot{Y} - \dot{Y}_E - \dot{Y}_S$$
$$\rho_z = Z - Z_E - Z_S \qquad \dot{\rho}_z = \dot{Z} - \dot{Z}_E - \dot{Z}_S$$

$\dot{X}_S$, $\dot{Y}_S$, and $\dot{Z}_S$ are related to R, θ, and ϕ as follows:

$$\dot{X}_S = -\omega R \cos\theta \sin G$$
$$\dot{Y}_S = \omega R \cos\theta \cos\epsilon \cos G$$
$$\dot{Z}_S = -\omega R \cos\theta \sin\epsilon \cos G$$

where ω is the rotational rate of the Earth.

Partials of $\dot{\rho}$ with respect to spacecraft state are given by

$$\frac{\partial \dot{\rho}}{\partial X} = \frac{\dot{\rho}_x}{\rho} - \frac{\rho_x \dot{\rho}}{\rho^2} \qquad \frac{\partial \dot{\rho}}{\partial \dot{X}} = \frac{\rho_x}{\rho}$$

$$\frac{\partial \dot{\rho}}{\partial Y} = \frac{\dot{\rho}_y}{\rho} - \frac{\rho_y \dot{\rho}}{\rho^2} \qquad \frac{\partial \dot{\rho}}{\partial \dot{Y}} = \frac{\rho_y}{\rho}$$

$$\frac{\partial \dot{\rho}}{\partial Z} = \frac{\dot{\rho}_z}{\rho} - \frac{\rho_z \dot{\rho}}{\rho^2} \qquad \frac{\partial \dot{\rho}}{\partial \dot{Z}} = \frac{\rho_z}{\rho}$$

The partial of $\dot{\rho}$ with respect to R is given by

$$\frac{\partial \dot{\rho}}{\partial R} = \frac{\partial \dot{\rho}}{\partial X_S} \cdot \frac{\partial X_S}{\partial R} + \frac{\partial \dot{\rho}}{\partial Y_S} \cdot \frac{\partial Y_S}{\partial R} + \frac{\partial \dot{\rho}}{\partial Z_S} \cdot \frac{\partial Z_S}{\partial R} +$$

$$\frac{\partial \dot{\rho}}{\partial \dot{X}_S} \cdot \frac{\partial \dot{X}_S}{\partial R} + \frac{\partial \dot{\rho}}{\partial \dot{Y}_S} \cdot \frac{\partial \dot{Y}_S}{\partial R} + \frac{\partial \dot{\rho}}{\partial \dot{Z}_S} \cdot \frac{\partial \dot{Z}_S}{\partial R}$$

where

$$\frac{\partial \dot{\rho}}{\partial X_S} = - \frac{\partial \dot{\rho}}{\partial X} \quad , \text{ etc.}$$

and $$\frac{\partial \dot{\rho}}{\partial \dot{X}_S} = - \frac{\partial \dot{\rho}}{\partial \dot{X}} \quad , \text{ etc.}$$

The negatives of the partials of X_S, Y_S, Z_S, $\dot{X}_S$, $\dot{Y}_S$, and $\dot{Z}_S$ with respect to R, θ, and ϕ are summarized in the subroutine STAPRL analysis. Partials of $\dot{\rho}$ with respect to θ and ϕ are treated similarly.

C. Star-planet angle measurement α .

A star-planet angle measurement has form α

$$\alpha = \alpha (\vec{X}, a, e, i, \Omega , \omega , M)$$

where a, e, i, Ω , ω , and M are the standard set of target planet orbital elements.

If we differ $\vec{\rho} = (\rho_x, \rho_y, \rho_z)$ to be the position of the target planet relative to the spacecraft and (u, v, w) to be the direction cosines of the relevant star, then

$$\alpha = \cos^{-1} \left[\frac{1}{\rho} (u\rho_x + v\rho_y + w\rho_z) \right]$$

where

$$\rho_x = X_p - X, \quad \rho_y = Y_p - Y, \quad \rho_z = Z_p - Z,$$

and (X_p, Y_p, Z_p) represent the position coordinates of the target planet.

Partials of α with respect to spacecraft state are given by

$$\frac{\partial \alpha}{\partial X} = \frac{1}{\sin \alpha}\left(\frac{u}{\rho} - \frac{\rho_x \cos \alpha}{\rho^2}\right) \qquad \frac{\partial \alpha}{\partial \dot{X}} = 0$$

$$\frac{\partial \alpha}{\partial Y} = \frac{1}{\sin \alpha}\left(\frac{v}{\rho} - \frac{\rho_y \cos \alpha}{\rho^2}\right) \qquad \frac{\partial \alpha}{\partial \dot{Y}} = 0$$

$$\frac{\partial \alpha}{\partial Z} = \frac{1}{\sin \alpha}\left(\frac{w}{\rho} - \frac{\rho_z \cos \alpha}{\rho^2}\right) \qquad \frac{\partial \alpha}{\partial \dot{Z}} = 0$$

where

$$\sin \alpha = + \left[1 - \cos^2 \alpha\right]^{\frac{1}{2}} .$$

The partial of α with respect to target planet semi-major axis is given by

$$\frac{\partial \alpha}{\partial a} = \frac{\partial \alpha}{\partial X_p} \cdot \frac{\partial X_p}{\partial a} + \frac{\partial \alpha}{\partial Y_p} \cdot \frac{\partial Y_p}{\partial a} + \frac{\partial \alpha}{\partial Z_p} \cdot \frac{\partial Z_p}{\partial a}$$

where $\frac{\partial \alpha}{\partial X_p} = -\frac{\partial \alpha}{\partial X}$, $\frac{\partial \alpha}{\partial Y_p} = -\frac{\partial \alpha}{\partial Y}$, $\frac{\partial \alpha}{\partial Z_p} = -\frac{\partial \alpha}{\partial Z}$,

and partials of X_p, Y_p, and Z_p with respect to semi-major axis are summarized in the subroutine TARPRL analysis. Partials of α with respect to $\dot{X}_p$, $\dot{Y}_p$, and $\dot{Z}_p$ do not appear in the above expression since they are all zero. Partials of α with respect to the remaining target planet orbital elements are treated similarly.

D. Apparent planet diameter measurement β.

An apparent planet diameter measurement has form

$$\beta = \beta(\vec{X}, a, e, i, \Omega, \omega, M)$$

where all arguments have been defined previously.

Defining $\vec{\rho} = (\rho_x, \rho_y, \rho_z)$ to be the position of the target planet relative to the spacecraft and R_p to be the radius of the target planet, the apparent planet diameter can then be written as

$$\beta = 2 \sin^{-1}\left(\frac{R_p}{\rho}\right)$$

Partials of β with respect to spacecraft state are given by

$$\frac{\partial \beta}{\partial X} = \frac{2 R_p \rho_x}{\rho^2 \left[\rho^2 - R_p^2\right]^{\frac{1}{2}}} \qquad \frac{\partial \beta}{\partial \dot{X}} = 0$$

$$\frac{\partial \beta}{\partial Y} = \frac{2 R_p \rho_y}{\rho^2 \left[\rho^2 - R_p^2\right]^{\frac{1}{2}}} \qquad \frac{\partial \beta}{\partial \dot{Y}} = 0$$

$$\frac{\partial \beta}{\partial Z} = \frac{2 R_p \rho_z}{\rho^2 \left[\rho^2 - R_p^2\right]^{\frac{1}{2}}} \qquad \frac{\partial \beta}{\partial \dot{Z}} = 0$$

The partial of β with respect to target planet semi-major axis is given by

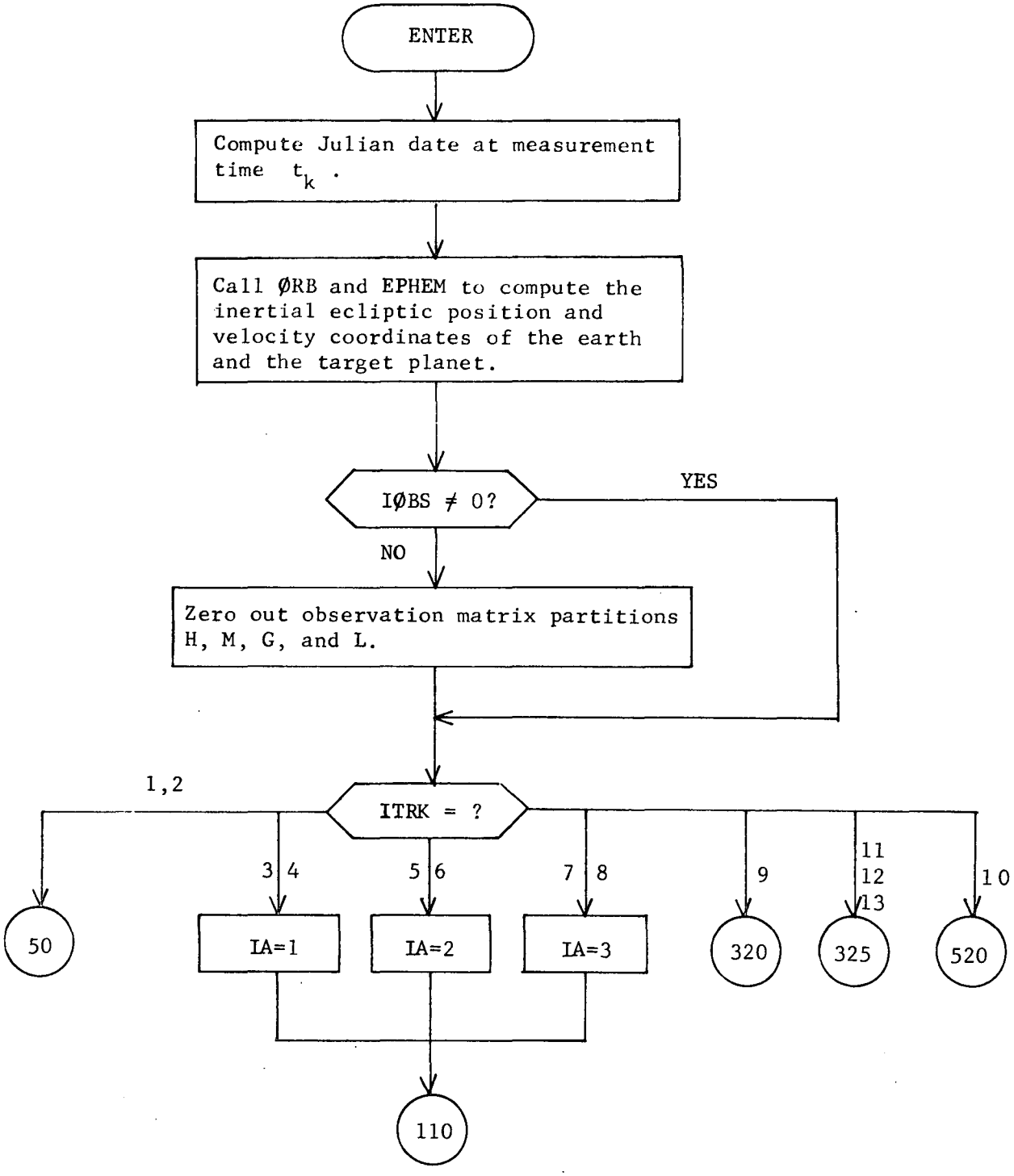
$$\frac{\partial \beta}{\partial a} = \frac{\partial \beta}{\partial X_p} \cdot \frac{\partial X_p}{\partial a} + \frac{\partial \beta}{\partial Y_p} \cdot \frac{\partial Y_p}{\partial a} + \frac{\partial \beta}{\partial Z_p} \cdot \frac{\partial Z_p}{\partial a}$$

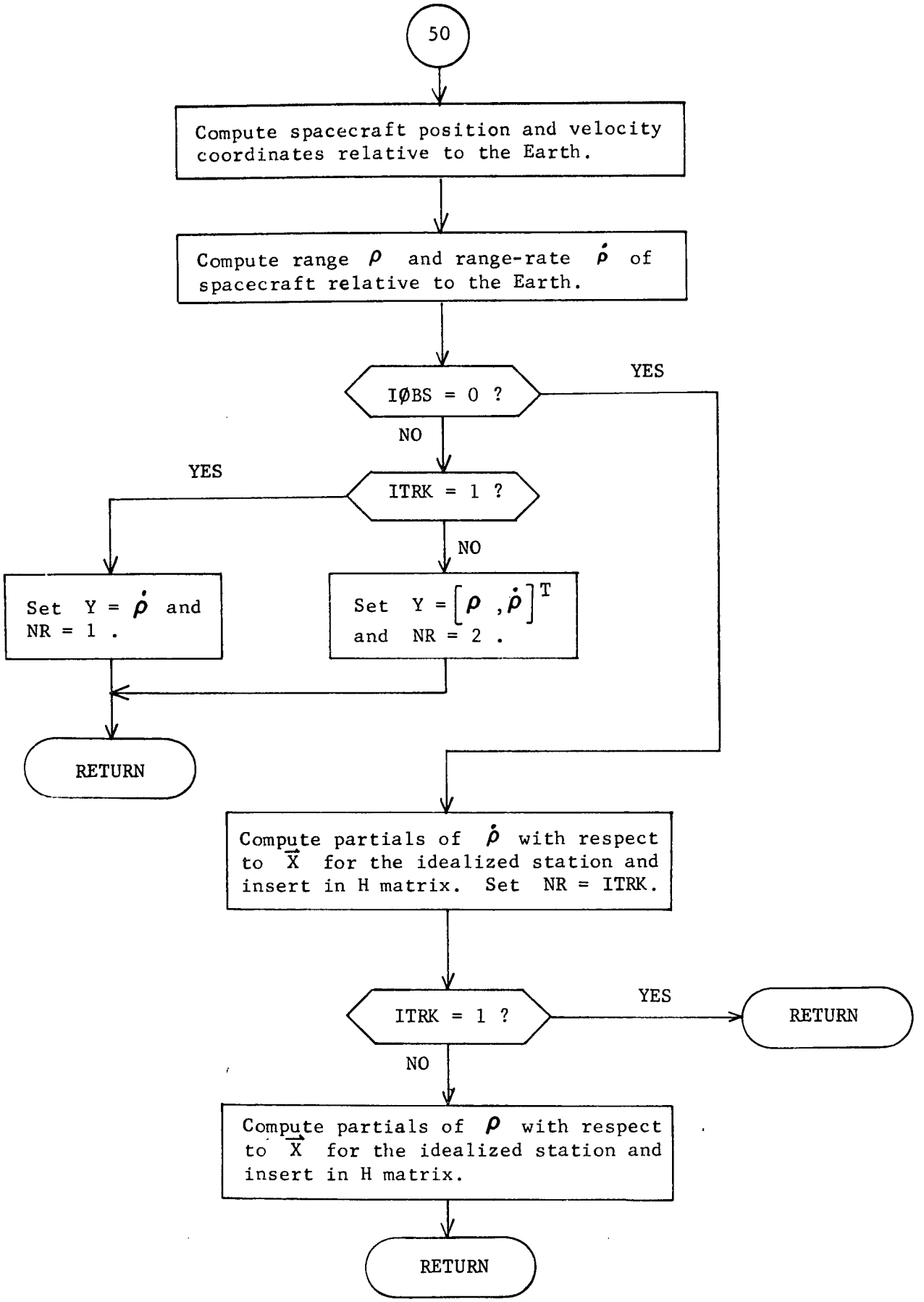
where $\frac{\partial \beta}{\partial X_p} = -\frac{\partial \beta}{\partial X}$, $\frac{\partial \beta}{\partial Y_p} = -\frac{\partial \beta}{\partial Y}$, $\frac{\partial \beta}{\partial Z_p} = -\frac{\partial \beta}{\partial Z}$,

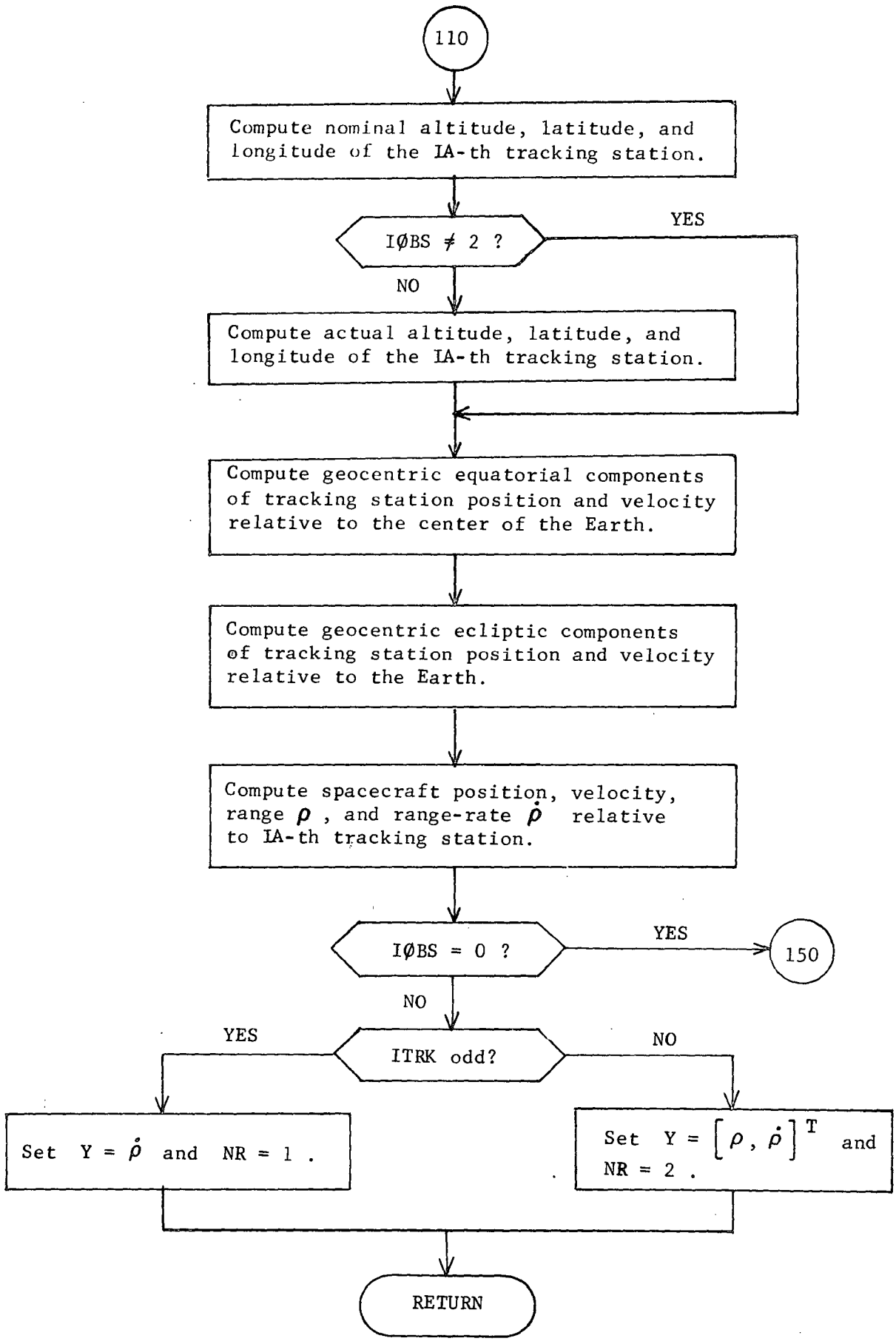
and partials of X_p, Y_p, and Z_p with respect to semi-major axis are summarized in the subroutine TARPRL analysis.

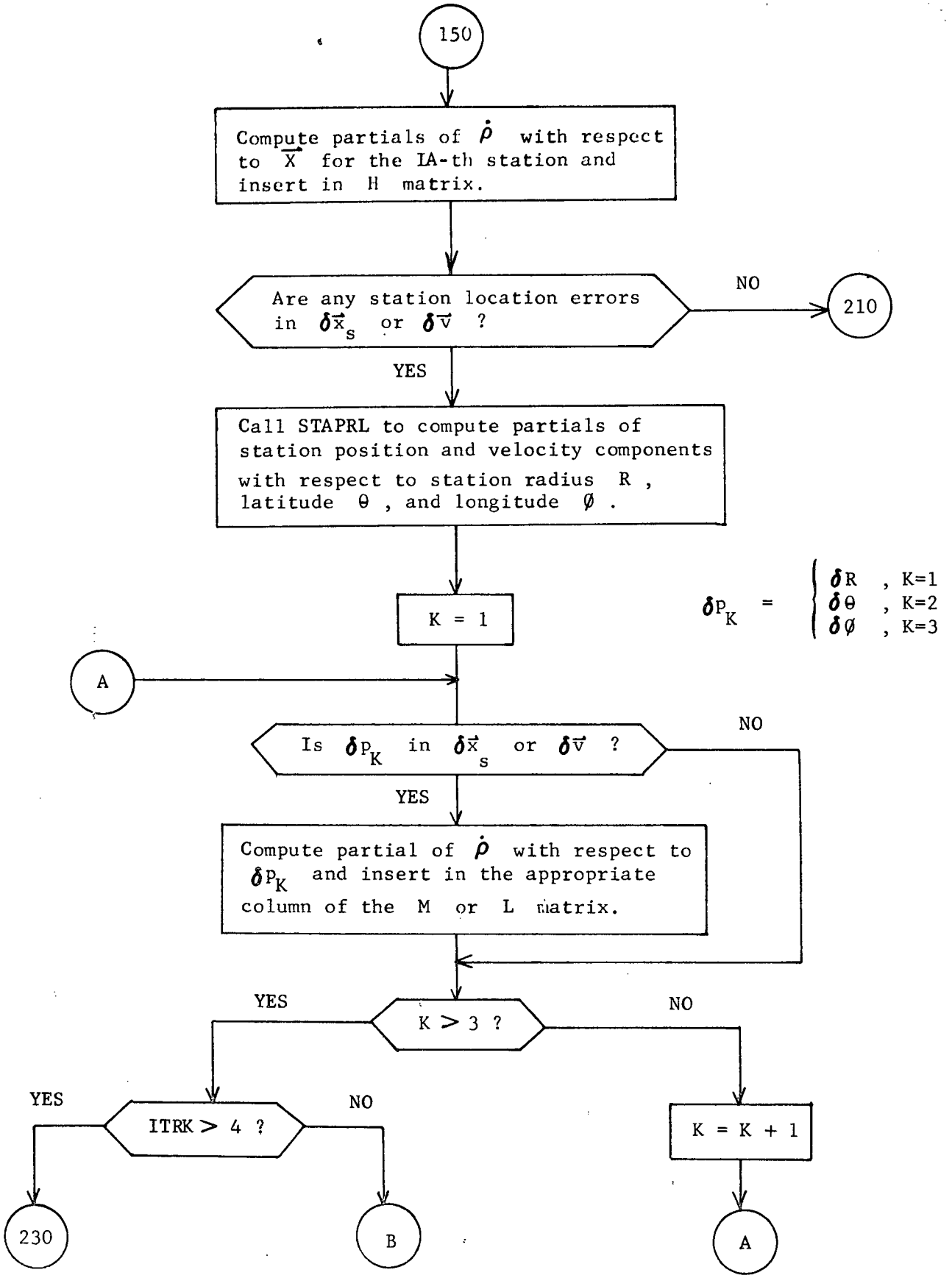
Partials of β with respect to $\dot{X}_p$, $\dot{Y}_p$, and $\dot{Z}_p$ do not appear in the above expression since they are all zero. Partials of β with respect to the remaining target planet orbital elements are treated similarly.

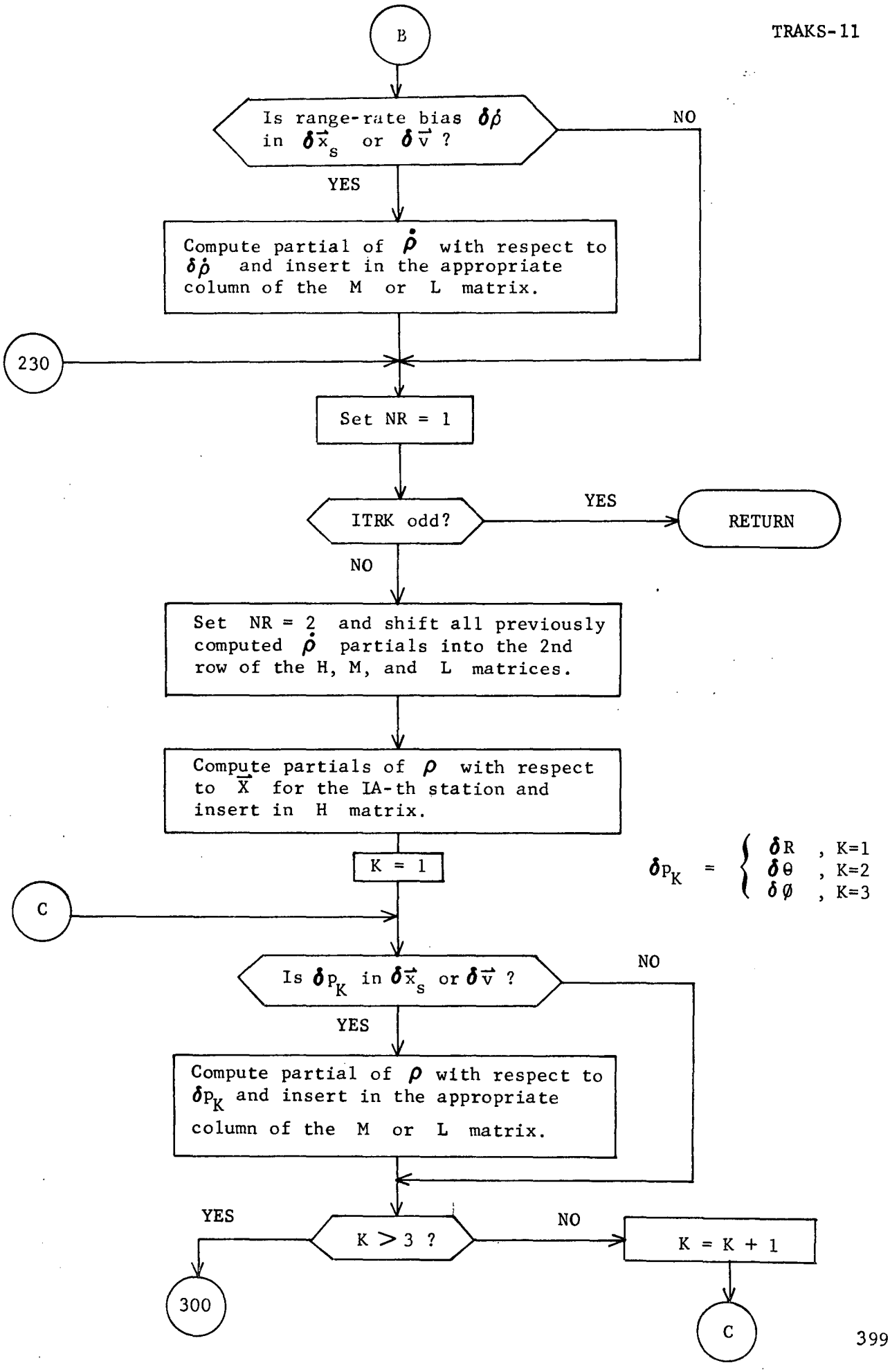
TRAKS Flow Chart

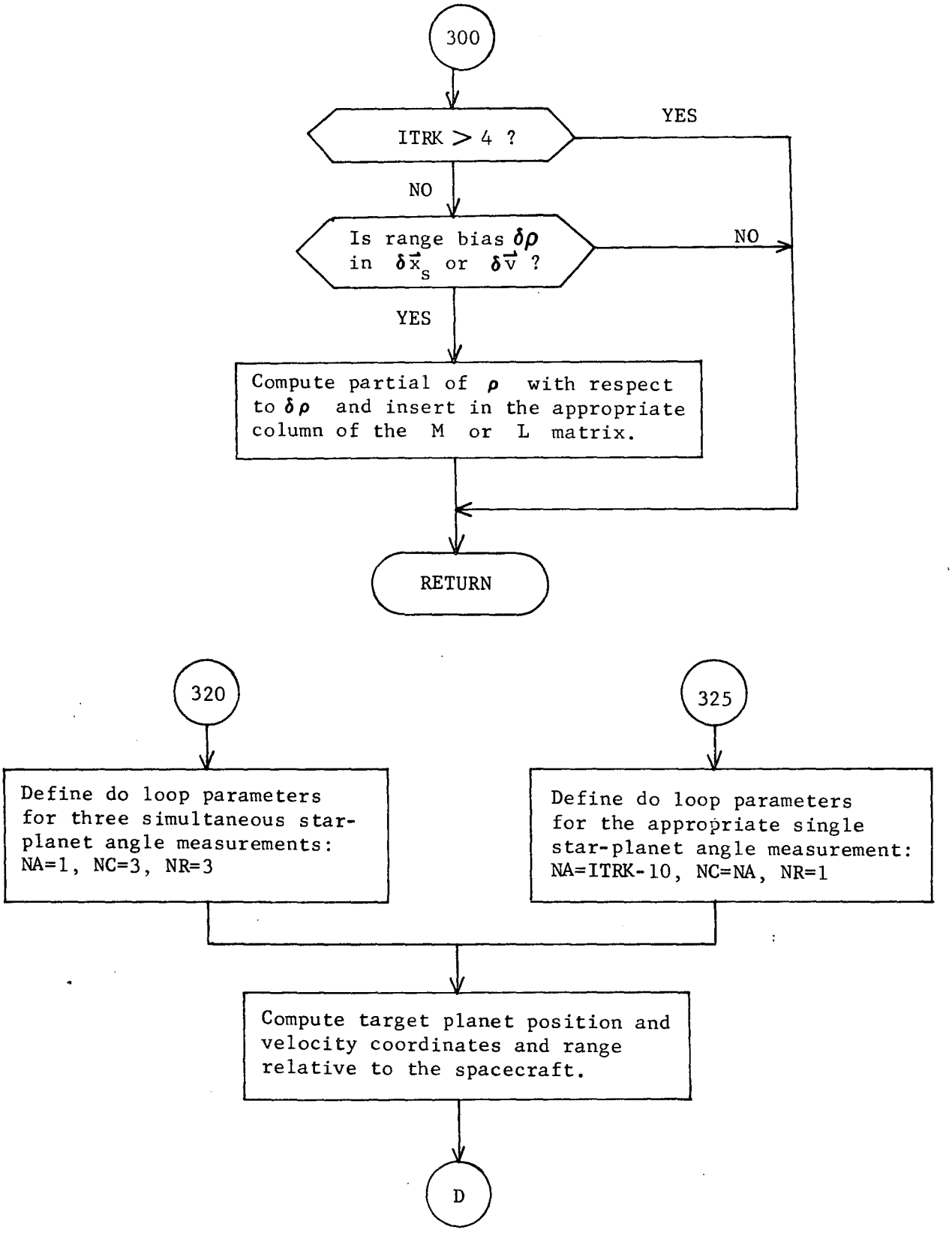


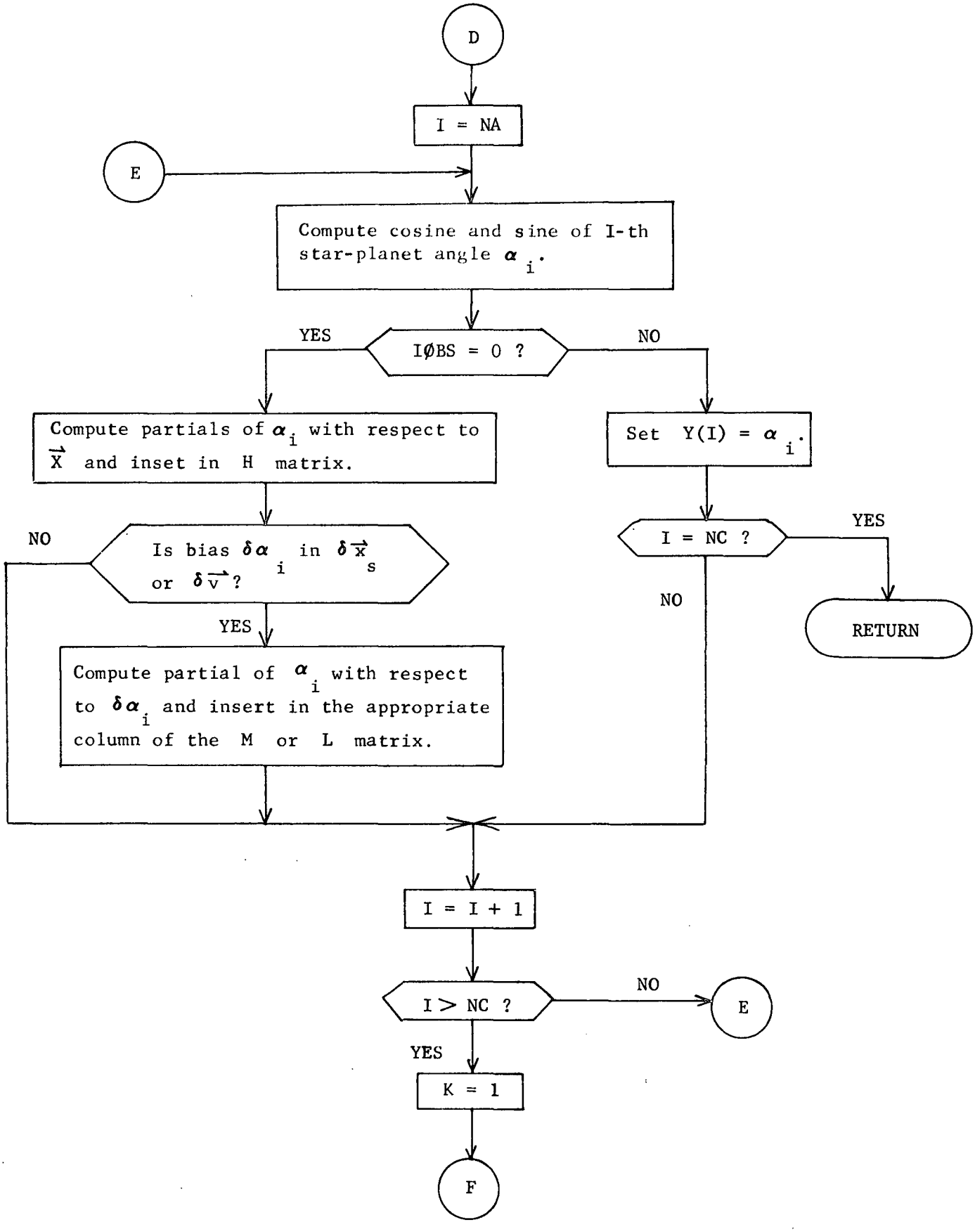

50
Compute spacecraft position and velocity coordinates relative to the Earth.
Compute range ρ and range-rate $\dot{\rho}$ of spacecraft relative to the Earth.
IØBS = 0 ?
YES
NO
ITRK = 1 ?
YES
NO
Set Y = $\dot{\rho}$ and NR = 1 .
Set Y = $\left[\rho , \dot{\rho}\right]^T$ and NR = 2 .
RETURN
Compute partials of $\dot{\rho}$ with respect to $\vec{X}$ for the idealized station and insert in H matrix. Set NR = ITRK.
ITRK = 1 ?
YES
RETURN
NO
Compute partials of ρ with respect to $\vec{X}$ for the idealized station and insert in H matrix.
RETURN


110
Compute nominal altitude, latitude, and longitude of the IA-th tracking station.
IØBS ≠ 2 ?
YES
NO
Compute actual altitude, latitude, and longitude of the IA-th tracking station.
Compute geocentric equatorial components of tracking station position and velocity relative to the center of the Earth.
Compute geocentric ecliptic components of tracking station position and velocity relative to the Earth.
Compute spacecraft position, velocity, range ρ , and range-rate ρ̇ relative to IA-th tracking station.
IØBS = 0 ?
YES
150
NO
YES
ITRK odd?
NO
Set Y = ρ̇ and NR = 1 .
Set Y = [ρ , ρ̇]^T and NR = 2 .
RETURN

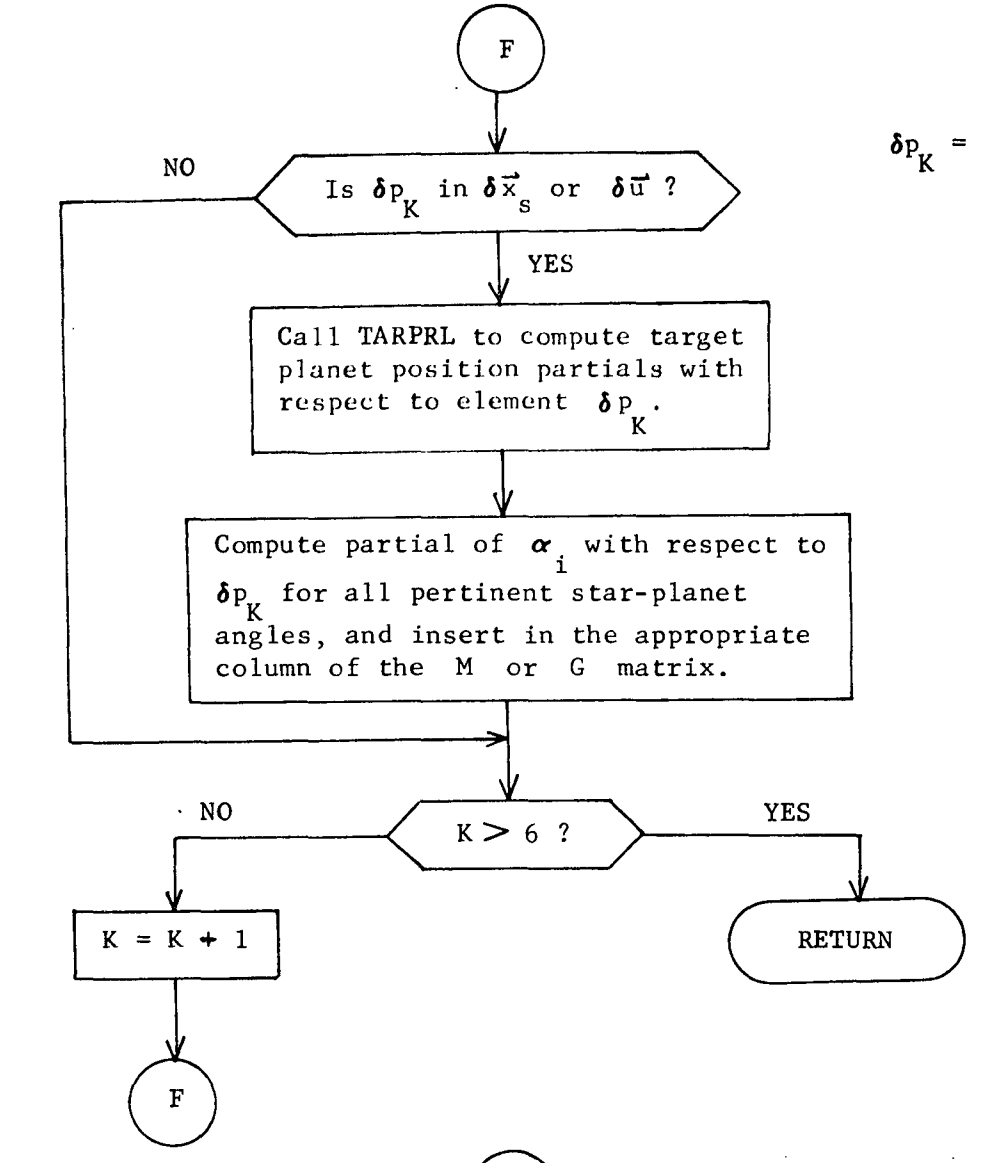

150
Compute partials of $\dot{\rho}$ with respect to $\vec{X}$ for the IA-th station and insert in H matrix.
Are any station location errors in $\delta\vec{x}_s$ or $\delta\vec{v}$?
NO
210
YES
Call STAPRL to compute partials of station position and velocity components with respect to station radius R , latitude θ , and longitude ϕ .
K = 1
$\delta p_K = \begin{cases} \delta R & , K=1 \\ \delta\theta & , K=2 \\ \delta\phi & , K=3 \end{cases}$
A
Is δp_K in $\delta\vec{x}_s$ or $\delta\vec{v}$?
NO
YES
Compute partial of $\dot{\rho}$ with respect to δp_K and insert in the appropriate column of the M or L matrix.
YES
K > 3 ?
NO
YES
ITRK > 4 ?
NO
K = K + 1
230
B
A

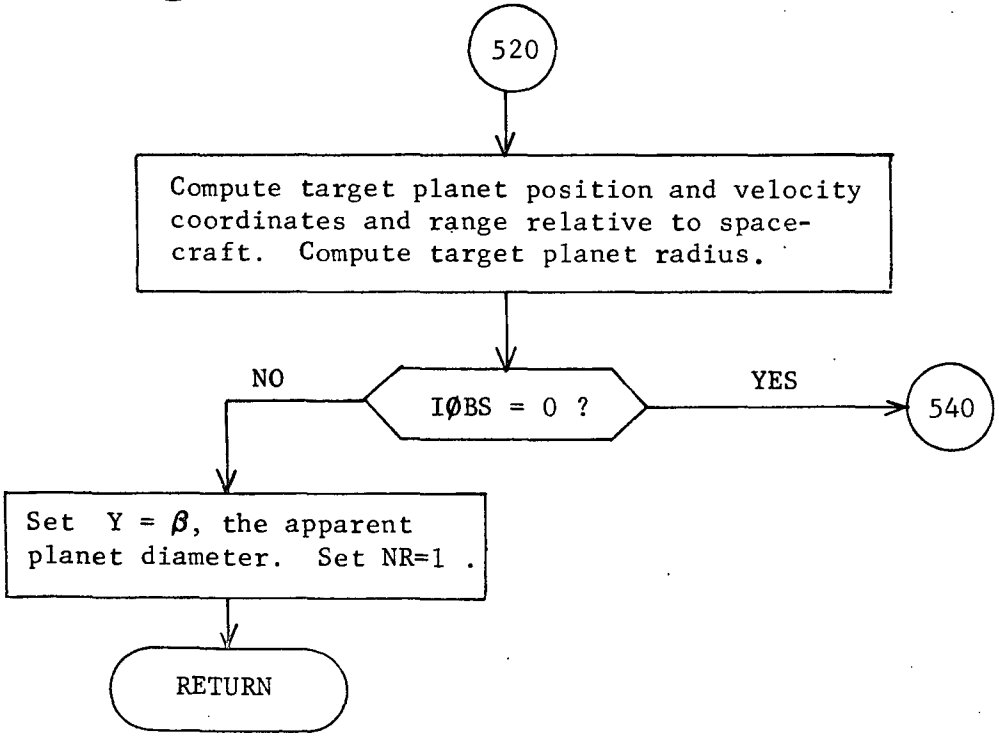

B
Is range-rate bias $\delta\dot{\rho}$ in $\delta\vec{x}_s$ or $\delta\vec{v}$?
NO
YES
Compute partial of $\dot{\rho}$ with respect to $\delta\dot{\rho}$ and insert in the appropriate column of the M or L matrix.
230
Set NR = 1
ITRK odd?
YES
RETURN
NO
Set NR = 2 and shift all previously computed $\dot{\rho}$ partials into the 2nd row of the H, M, and L matrices.
Compute partials of ρ with respect to $\vec{X}$ for the IA-th station and insert in H matrix.
K = 1
$\delta P_K = \begin{cases} \delta R, & K=1 \\ \delta\theta, & K=2 \\ \delta\phi, & K=3 \end{cases}$
C
Is δp_K in $\delta\vec{x}_s$ or $\delta\vec{v}$?
NO
YES
Compute partial of ρ with respect to δp_K and insert in the appropriate column of the M or L matrix.
YES
K > 3 ?
NO
K = K + 1
300
C

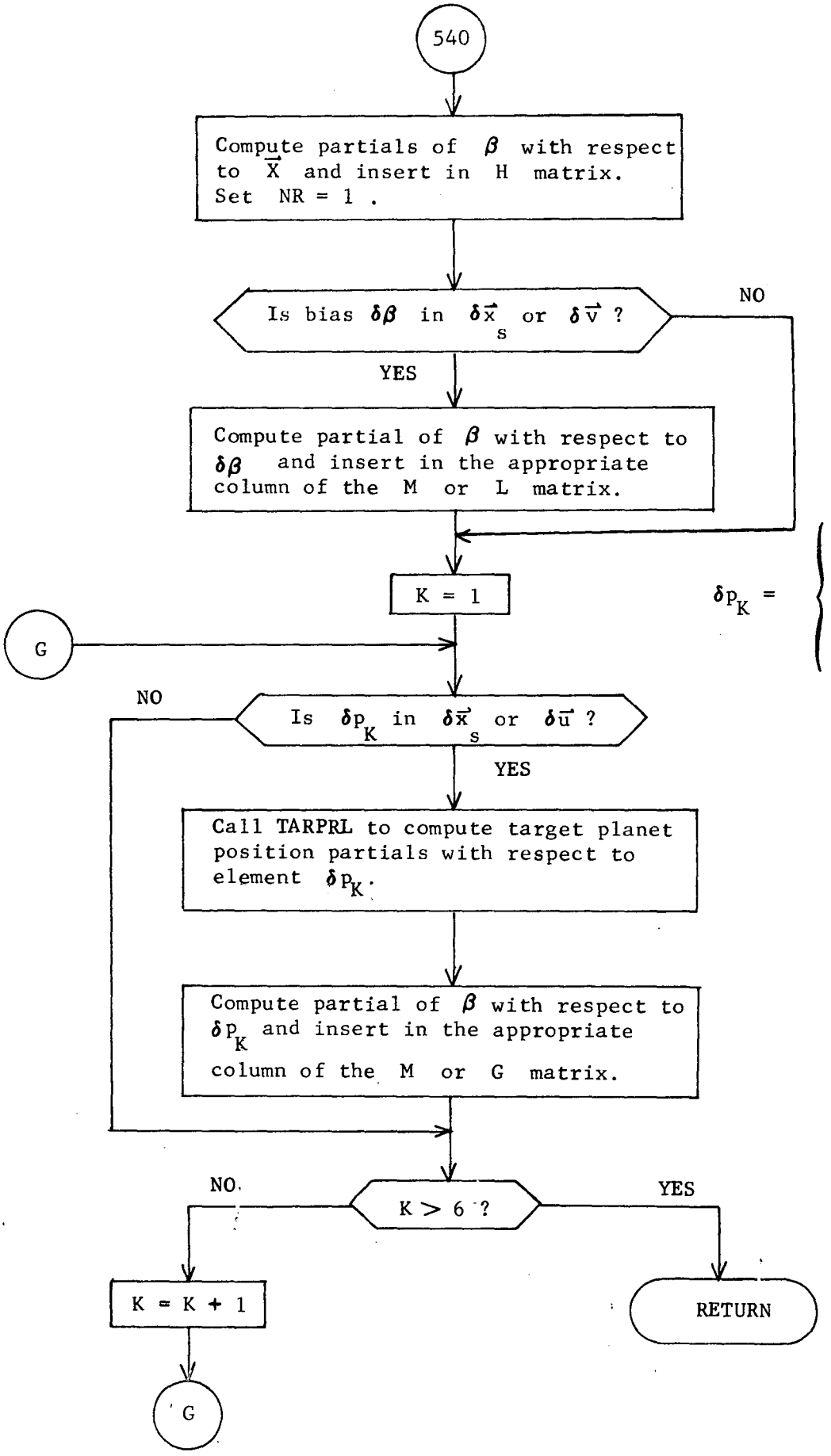

300
ITRK > 4 ?
YES
NO
Is range bias δρ in δx⃗s or δv⃗ ?
NO
YES
Compute partial of ρ with respect to δρ and insert in the appropriate column of the M or L matrix.
RETURN
320
325
Define do loop parameters for three simultaneous star-planet angle measurements: NA=1, NC=3, NR=3
Define do loop parameters for the appropriate single star-planet angle measurement: NA=ITRK-10, NC=NA, NR=1
Compute target planet position and velocity coordinates and range relative to the spacecraft.
D


D
I = NA
E
Compute cosine and sine of I-th star-planet angle α_i.
YES
IØBS = 0 ?
NO
Compute partials of α_i with respect to $\vec{X}$ and inset in H matrix.
Set Y(I) = α_i.
NO
Is bias $\delta\alpha_i$ in $\delta\vec{x}_s$ or $\delta\vec{v}$?
I = NC ?
YES
NO
RETURN
YES
Compute partial of α_i with respect to $\delta\alpha_i$ and insert in the appropriate column of the M or L matrix.
I = I + 1
I > NC ?
NO
E
YES
K = 1
F

$$\delta p_K = \begin{cases} \delta a, & K=1 \\ \delta e, & K=2 \\ \delta i, & K=3 \\ \delta \Omega, & K=4 \\ \delta \omega, & K=5 \\ \delta M, & K=6 \end{cases}$$






540
Compute partials of β with respect to $\vec{X}$ and insert in H matrix. Set NR = 1 .
Is bias $\delta\beta$ in $\delta\vec{x}_s$ or $\delta\vec{v}$?
NO
YES
Compute partial of β with respect to $\delta\beta$ and insert in the appropriate column of the M or L matrix.
K = 1
δp_K = δa , K=1; δe , K=2; δi , K=3; $\delta\Omega$, K=4; $\delta\omega$, K=5; δM , K=6
G
NO
Is δp_K in $\delta\vec{x}_s$ or $\delta\vec{u}$?
YES
Call TARPRL to compute target planet position partials with respect to element δp_K.
Compute partial of β with respect to δp_K and insert in the appropriate column of the M or G matrix.
NO
K > 6 ?
YES
K = K + 1
RETURN
G

TRANS Analysis

Subroutine TRANS transforms the position and velocity components of the spacecraft from one coordinate system to another. The three options available with this subroutine are summarized below.

1) Convert from geocentric equatorial coordinates to geocentric ecliptic coordinates using the following equations:

$$X = X$$
$$Y = Y \cos\epsilon + Z \sin\epsilon$$
$$Z = -Y \sin\epsilon + Z \cos\epsilon$$
$$\dot{X} = \dot{X}$$
$$\dot{Y} = \dot{Y} \cos\epsilon + \dot{Z} \sin\epsilon$$
$$\dot{Z} = -\dot{Y} \sin\epsilon + \dot{Z} \cos\epsilon$$

2) Convert from geocentric equatorial coordinates to heliocentric ecliptic coordinates. The same procedure as above is used to convert from geocentric equatorial to geocentric ecliptic. Then translate according to the following equations:

$$X = X + X_E \qquad \dot{X} = \dot{X} + \dot{X}_E$$
$$Y = Y + Y_E \qquad \dot{Y} = \dot{Y} + \dot{Y}_E$$
$$Z = Z + Z_E \qquad \dot{Z} = \dot{Z} + \dot{Z}_E$$

3) Convert from geocentric ecliptic coordinates to heliocentric ecliptic coordinates using the following equations:

$$X = X + X_E \qquad \dot{X} = \dot{X} + \dot{X}_E$$
$$Y = Y + Y_E \qquad \dot{Y} = \dot{Y} + \dot{Y}_E$$
$$Z = Z + Z_E \qquad \dot{Z} = \dot{Z} + \dot{Z}_E$$

TRAPAR Analysis

The coordinate systems and variables required for the derivation of the first four navigation parameters are shown in Figure 1. The inertial coordinate system XYZ may be heliocentric or barycentric ecliptic. The position and velocity of the earth in inertial space is given by $\vec{r}_E$ and $\vec{v}_E$; that of the spacecraft, by $\vec{r}$ and $\vec{v}$; and that of the target planet (or moon), by $\vec{r}_{TP}$ and $\vec{v}_{TP}$. The xyz coordinate system is geocentric equatorial.

1. Flight path angle, γ.

   Let θ denote the angle between $\vec{r}$ and $\vec{v}$, so that

$$\cos\theta = \frac{\vec{r}\cdot\vec{v}}{r\,v} \quad \text{and} \quad \sin\theta = +\left[1 - \cos^2\theta\right]^{\frac{1}{2}} .$$

   Then

$$\gamma = \frac{\pi}{2} - \theta .$$

2. Angle between relative velocity and plane of the sky, i'.

   The plane of the sky is defined as the plane perpendicular to the vector $\vec{r} - \vec{r}_E$. Let θ' denote the angle between $\vec{r} - \vec{r}_E$ and $\vec{v} - \vec{v}_E$, so that

$$\cos\theta' = \frac{(\vec{r} - \vec{r}_E)\cdot(\vec{v} - \vec{v}_E)}{|\vec{r} - \vec{r}_E|\cdot|\vec{v} - \vec{v}_E|} \quad \text{and} \quad \sin\theta' = +\left[1 - \cos^2\theta'\right]^{\frac{1}{2}}$$

   Then

$$i' = \frac{\pi}{2} - \theta' .$$

   Note that i' is not defined if the relative velocity $\vec{v} - \vec{v}_E$ is zero.

3. Geocentric declination, δ.

   Let (x, y, z) denote the geocentric equatorial components of $\vec{r} - \vec{r}_E$. Then

$$\delta = \tan^{-1}\left(\frac{z}{\sqrt{x^2 + y^2}}\right) .$$

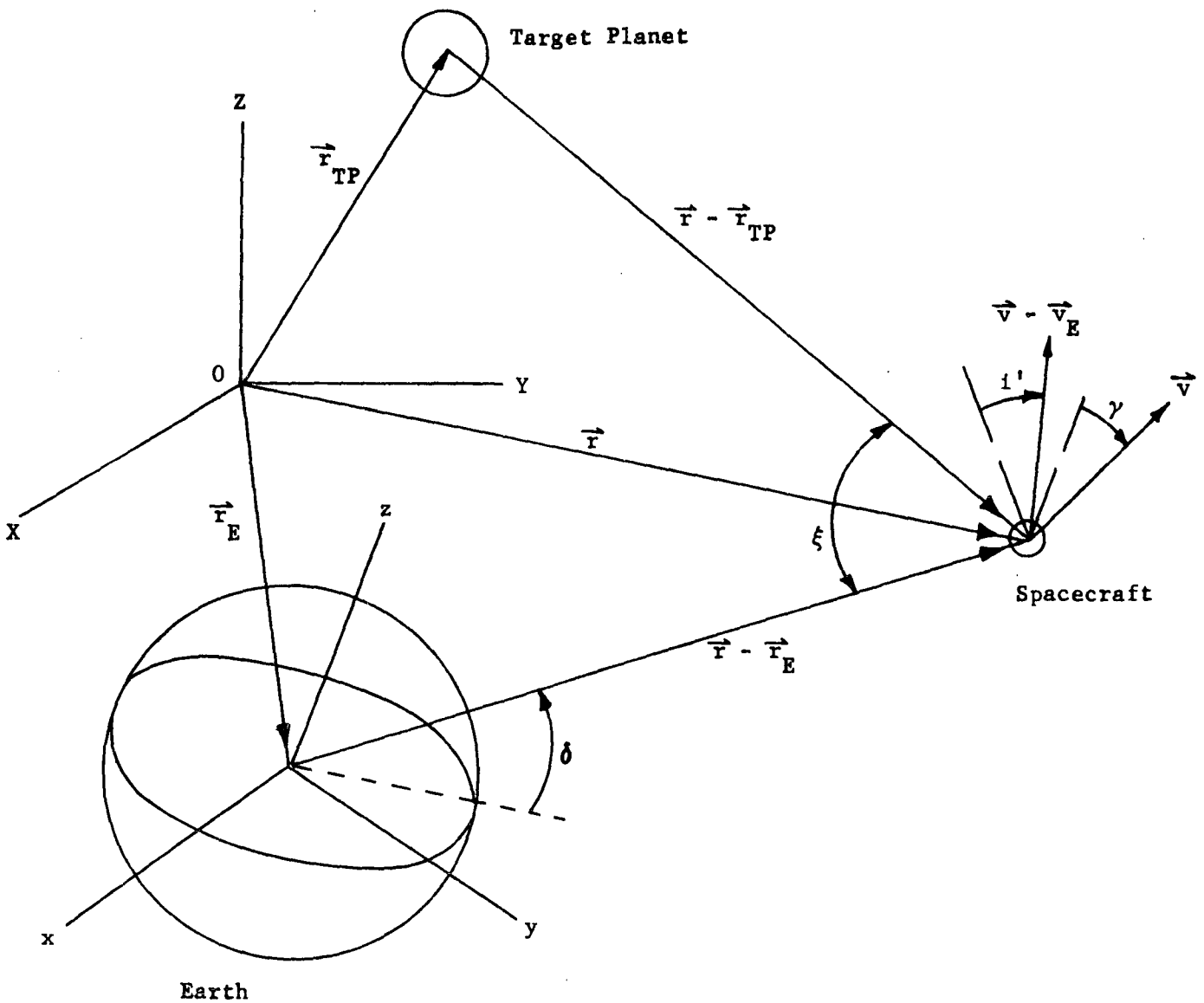

Target Planet
Z
$\vec{r}_{TP}$
$\vec{r} - \vec{r}_{TP}$
$\vec{v} - \vec{v}_E$
i'
γ
$\vec{v}$
0
Y
$\vec{r}$
X
$\vec{r}_E$
z
ξ
Spacecraft
$\vec{r} - \vec{r}_E$
δ
x
y
Earth

Figure 1

4. Earth/spacecraft/target planet angle, ξ.

The angle ξ is the angle between the vectors $\vec{r} - \vec{r}_E$ and $\vec{r} - \vec{r}_{TP}$, so that

$$\cos \xi = \frac{(\vec{r} - \vec{r}_E) \cdot (\vec{r} - \vec{r}_{TP})}{|\vec{r} - \vec{r}_E| \cdot |\vec{r} - \vec{r}_{TP}|}$$

and $$\sin \xi = + \left[1 - \cos^2 \xi \right]^{\frac{1}{2}} .$$

The next two navigation parameters relate to the spacecraft antenna axis. The pertinent geometry is shown in Figure 2. The antenna axis $\vec{\alpha}$ is defined as the intersection between the antenna plane (the plane perpendicular to the spacecraft spin axis $\vec{s}$) and the plane formed by the $\vec{r} - \vec{r}_E$ and $\vec{s}$ vectors. The vector $\vec{\rho}$ originates from the limb of the sun and lies in the $\vec{r}$, $\vec{\alpha}$ plane.

5. Antenna axis/Earth angle, β.

Let ψ denote the angle between the unit spin axis vector $\vec{s}$ and $\vec{r} - \vec{r}_E$, so that

$$\cos \psi = \frac{\vec{s} \cdot (\vec{r} - \vec{r}_E)}{|\vec{r} - \vec{r}_E|} \quad \text{and} \quad \sin \psi = + \left[1 - \cos^2 \psi \right]^{\frac{1}{2}} .$$

Then

$$\beta = \frac{\pi}{2} - \psi .$$

Note that the antenna axis is not uniquely defined when the angle $\psi = 0$.

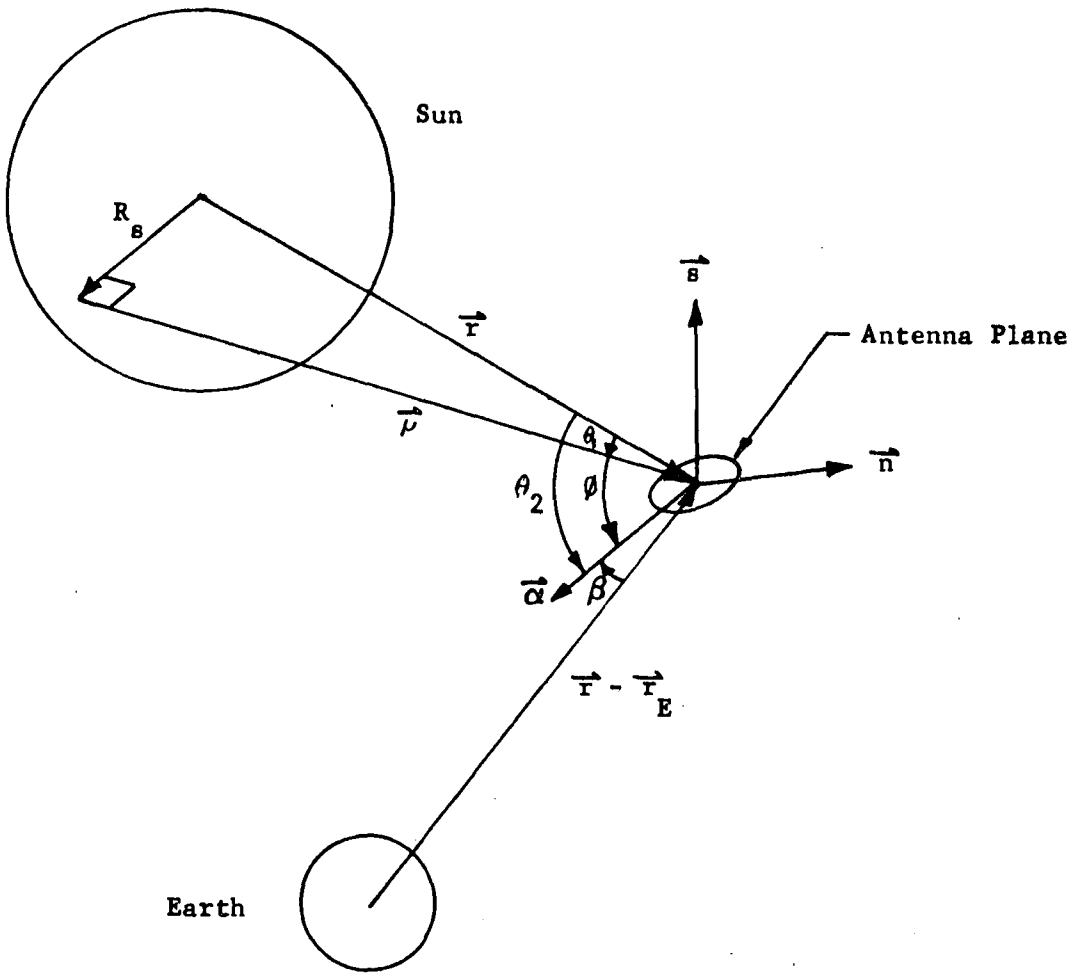


Figure 2. Antenna Axis Geometry

6. Antenna axis/limb of Sun angle, ϕ.

The unit vector $\vec{n}$ normal to the $\vec{s}$, $\vec{r} - \vec{r}_E$ plane is given by

$$\vec{n} = \frac{(\vec{r} - \vec{r}_E) \times \vec{s}}{|(\vec{r} - \vec{r}_E) \times \vec{s}|} .$$

Then the unit antenna axis vector $\vec{\alpha}$ is given by

$$\vec{\alpha} = \vec{n} \times \vec{s} .$$

The angle θ_2 denotes the angle between the vectors $\vec{r}$ and $\vec{\alpha}$, so that

$$\cos\theta_2 = -\frac{\vec{r}\cdot\vec{\alpha}}{r} \quad \text{and} \quad \sin\theta_2 = +\left[1 - \cos^2\theta_2\right]^{\frac{1}{2}}.$$

The angle θ_1 denotes the angle between the vectors $\vec{\rho}$ and $\vec{r}$, so that

$$\theta_1 = \sin^{-1}\left(\frac{R_s}{r}\right), \quad 0 \leq \theta_1 \leq \frac{\pi}{2}$$

where R_s is the radius of the Sun.

Then

$$\phi = \theta_2 - \theta_1 .$$

The final set of navigation parameters relate to spacecraft occultation ratios for the Sun and all other celestial bodies assumed in the dynamic model. The pertinent geometry is shown in Figure 3. The position of the i-th celestial body relative to the Sun is denoted by $\vec{r}_i$. Occultation parameters d_s and d_i are defined as the minimal distances from the centers of the Sun and i-th body, respectively, to the Earth/spacecraft vector $\vec{r} - \vec{r}_E$.

7. Spacecraft occultation ratio for the Sun.

The occultation ratio for the Sun is defined as d_s/R_s, where R_s is the Sun radius. As long as the occultation ratio is greater than one, the spacecraft is neither being occulted by the Sun nor passing in front of the Sun. The occultation ratio is computed only when the angle between the $\vec{r} - \vec{r}_E$ and $\vec{r}_E$ vectors is less than or equal to 90 degrees, or, equivalently, when

$$\vec{r}_E \cdot (\vec{r} - \vec{r}_E) \leq 0.$$

If this condition is satisfied, the occultation ratio is computed using the equations

$$d_s = \left[r_E^2 - b^2\right]^{\frac{1}{2}}$$

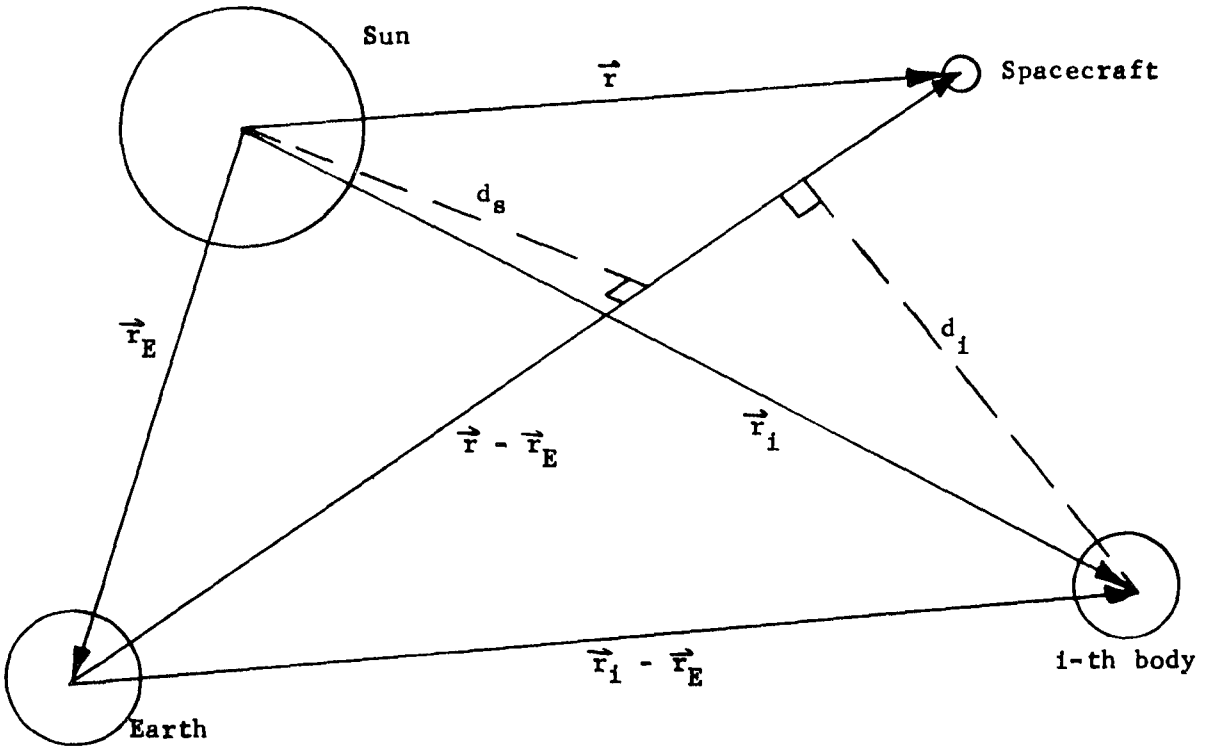


Figure 3. Occultation Geometry

and

$$b = \frac{-\vec{r}_E \cdot (\vec{r} - \vec{r}_E)}{|\vec{r} - \vec{r}_E|} \quad .$$

Occultation occurs if $\frac{d_s}{R_s} \leq 1$ and $|\vec{r} - \vec{r}_E| \geq r_E$; if $\frac{d_s}{R_s} \leq 1$ and $|\vec{r} - \vec{r}_E| < r_E$, then the spacecraft is passing in front of the Sun.

8. Spacecraft occultation ratios for other celestial bodies.

The occultation ratio for the i-th celestial body is defined as d_i/R_i, where R_i is the radius of the i-th body. The occultation ratio is

computed only when

$$(\vec{r} - \vec{r}_E) \cdot (\vec{r}_i - \vec{r}_E) \geq 0 \quad .$$

If this conditions is satisfied, the occultation ratio is computed using the equations

$$d_i = \left[a_i^2 - b_i^2 \right]^{\frac{1}{2}}$$

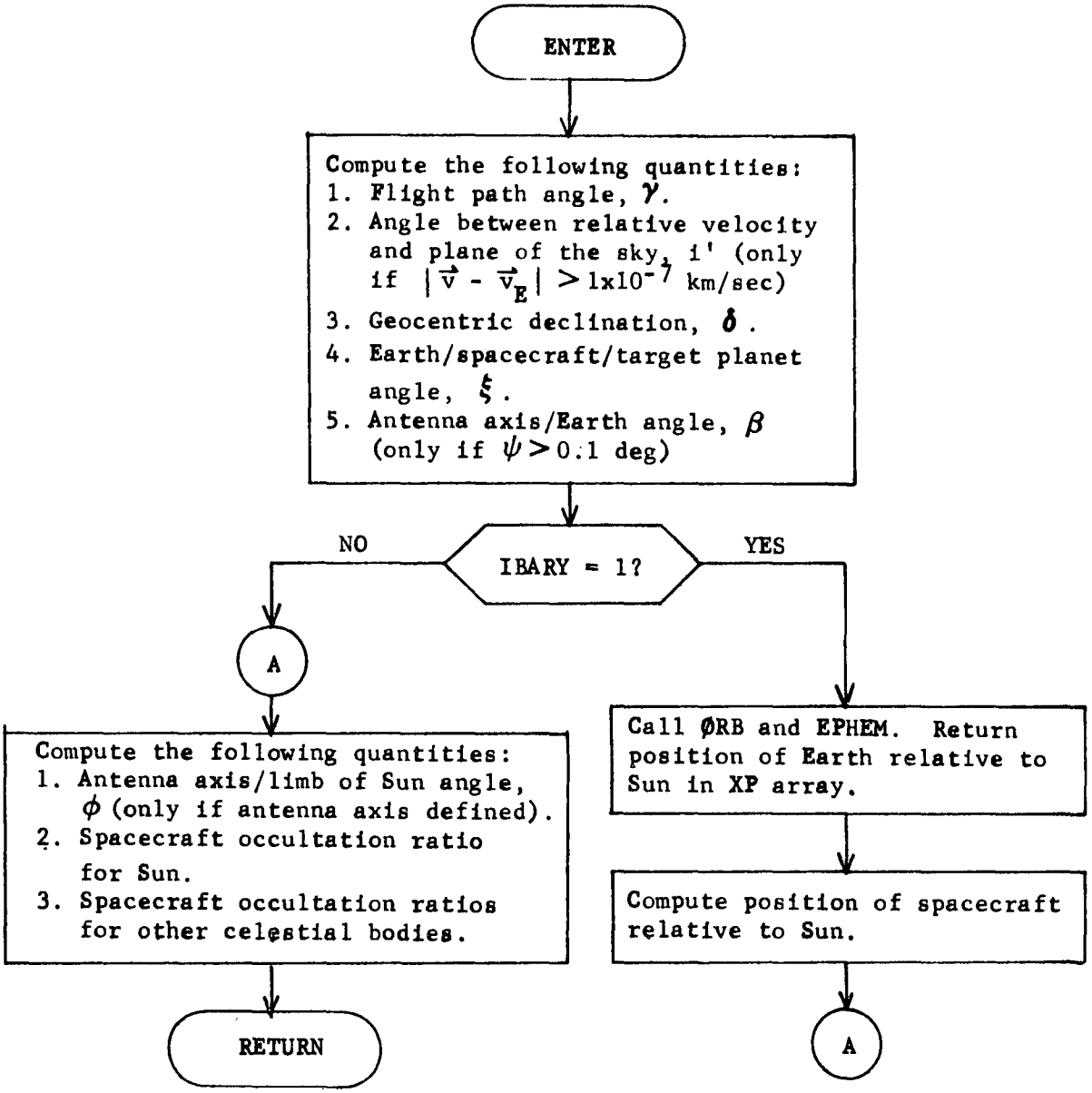
$$a_i = \left| \vec{r}_i - \vec{r}_E \right|$$

and

$$b_i = \frac{\left| \vec{r} - \vec{r}_E \right| \cdot \left(\vec{r}_i - \vec{r}_E \right)}{\left| \vec{r} - \vec{r}_E \right|} \quad .$$

Occultation occurs if $\frac{d_i}{R_i} \leq 1$ and $\left| \vec{r} - \vec{r}_E \right| \geq \left| \vec{r}_i - \vec{r}_E \right|$; if $\frac{d_i}{R_i} \leq 1$ and $\left| \vec{r} - \vec{r}_E \right| < \left| \vec{r}_i - \vec{r}_E \right|$, then the spacecraft is passing is front of the i-th celestial body.

TRAPAR Flow Chart



TRJTRY Analysis

TRJTRY determines the time of the next guidance event and integrates the nominal trajectory from the previous event time to the next time.

Special provisions must be made in determining the next guidance event because of the flexibility permitted in specifying the times of those guidance events. For every guidance event i, parameters KTIM(i) and TIMG(i) will have been set before entering TRJTRY. KTIM(i) prescribes the epoch to which the guidance event i is referenced with KTIM(i) = 1,2,3 corresponding to epochs of initial time, sphere of influence (SOI) intersection, and closest approach (CA) passage respectively. TIMG(i) then specifies the time interval (days) from the epoch to the guidance event. The guidance events do not need to be arranged chronologically. After execution of each guidance event i the flag KTIM(i) is set equal to 0.

The first computational procedure in TRJTRY is the sequencing loop. Here a search determines the minimum value of TIMG(i) over all values of i such that KTIM(i) = 1. The time interval Δt between that time and the current time is then computed. If Δt is less than an allowable tolerance ϵ ($=10^{-5}$ days) the program returns to NOMNAL for the processing of the current event.

If $\Delta t \geq \epsilon$ TRJTRY must perform an integration to the next guidance event. TRJTRY first sets up flags controlling integration stopping conditions depending upon the current value of KSICA. The flag KSICA determines the current phase of the trajectory. KSICA is initially set equal to 1 (PRELIM). When the target planet SOI is encountered KSICA is set to 2. Finally when CA to the target planet occurs it is set to 3.

The stopping condition flags are ISP2 and ICL2. The flag ISP2 determines whether the integration should be stopped at SOI if encountered (ISP2 = 1) or not (ISP2 = 0). The flag ICL2 determines whether the integration should be stopped at CA if encountered (ICL2 = 1) or not (ICL2 = 0).

Therefore if KSICA = 1, TRJTRY sets ISP2 = 1 so that the integration will stop at the guidance event time only if that time occurs before SOI. But if the SOI is encountered before the event time, all times referenced to the SOI must be updated before determining the next event. Similarly when KSICA = 2 TRJTRY sets ICL2 = 1 so that times referenced to CA may be updated when CA occurs. Of course when KSICA = 3, all times have been updated (referenced to initial time) and neither ISP2 nor ICL2 need be set to 1.

Having set the stopping condition flags, TRJTRY now calls VMP for the propagation of the trajectory to the required stopping condition. At the end of the integration it records the current trajectory time and state.

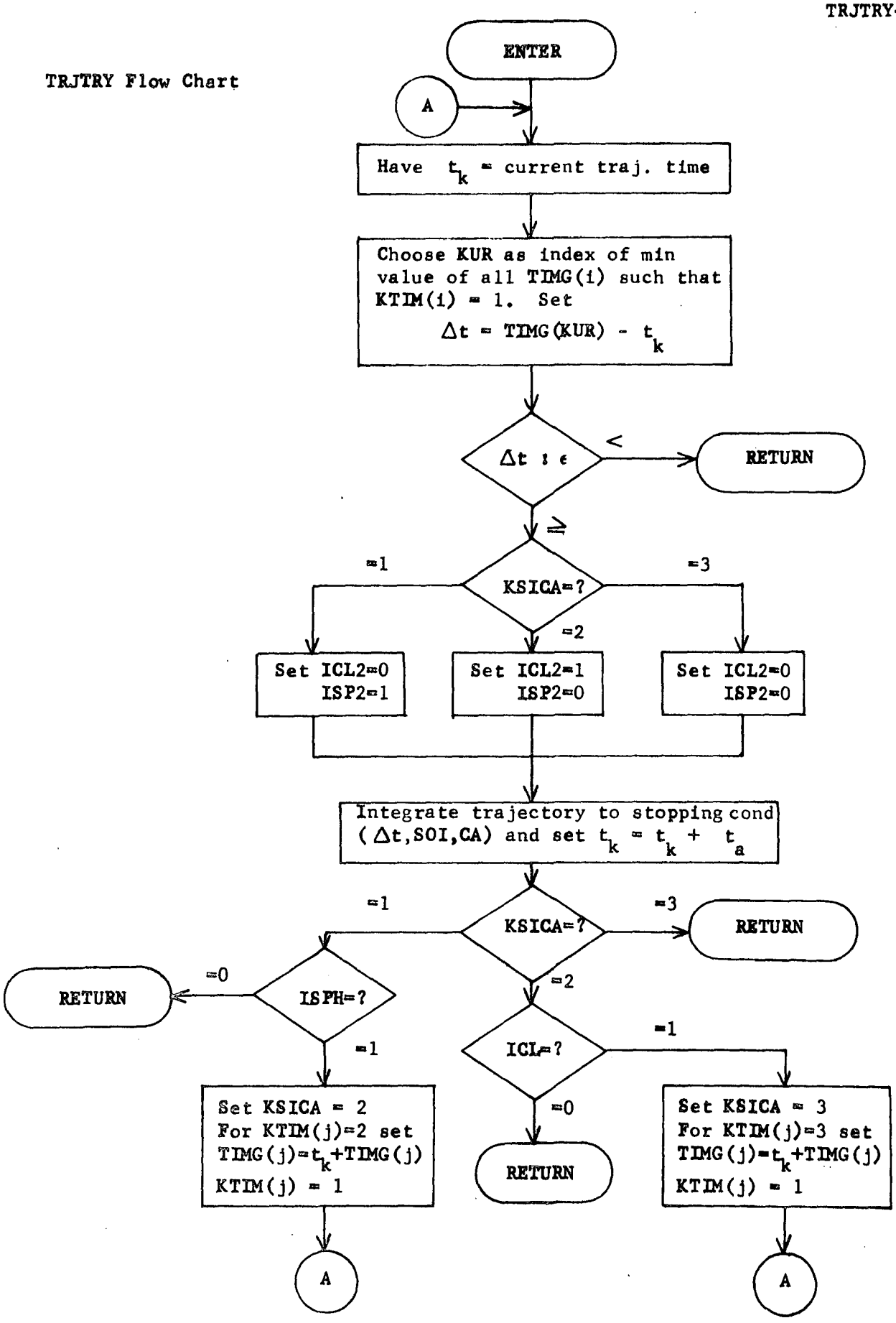
TRJTRY now sorts again on KSICA. If KSICA = 3, the trajectory has been integrated to the time of the current event and so control may be returned to NOMNAL.

If KSICA = 1 the SOI had not yet been reached at the previous event. TRJTRY then checks the flag ISPH. The flag ISPH reveals whether the current trajectory intersected the target planet SOI (ISPH = 1) or did not (ISPH = 0). Therefore if ISPH = 0, the current guidance event occurred before the trajectory intersected the SOI and thus the current state corresponds to the time of the guidance event. Therefore the return is made to NOMNAL.

If however KSICA = 1 and ISPH = 1 the trajectory integration was stopped at the SOI. TRJTRY now sets KSICA = 2 and updates all times referenced to the SOI so that they are now referenced to initial time (KTIM(i) = 1). It reenters the sequencing loop to determine the time of the next guidance event where the candidate events now include those originally referenced to SOI.

Similar steps are made when KSICA = 2. The flag ICL designates whether the current trajectory had a CA (ICL = 1) or not (ICL = 0). If KSICA = 2 and ICL = 0, the trajectory encountered the guidance event before reaching a CA so the return is made to NOMNAL. If KSICA = 2 and ICL = 1, the final time and state of the trajectory refer to closest approach. In this case TRJTRY sets KSICA = 3 and updates to initial time all times originally referenced to CA. It then returns to the sequencing loop.

TRJTRY Flow Chart



VARADA Analysis

Subroutine VARADA employs numerical differencing to compute the variation matrix η for the three-variable B-plane guidance policy in the guidance event of the error analysis mode. See subroutine VARSIM Analysis for further analytical details, since the only difference between VARADA and VARSIM is that VARADA computations are based on the most recent targeted nominal, while VARSIM computations are based on the most recent nominal. The VARADA flow chart is identical to that of VARSIM except for the fact that in VARADA the nominal position/velocity state at t_{SI} is saved prior to calling VARADA, while in VARSIM it is saved locally.

VARSIM Analysis

Subroutine VARSIM employs numerical differencing to compute the variation matrix η for the three-variable B-plane guidance policy in the guidance event of the simulation mode. This variation matrix relates deviations in the position/velocity state at t_k to deviations in B·T, B·R, and t_{SI}:

$$\begin{bmatrix} \delta B\cdot T \\ \delta B\cdot R \\ \delta t_{SI} \end{bmatrix} = \eta \begin{bmatrix} \delta\vec{R}_k \\ \delta\vec{V}_k \end{bmatrix} = \eta\delta\vec{X}_k$$

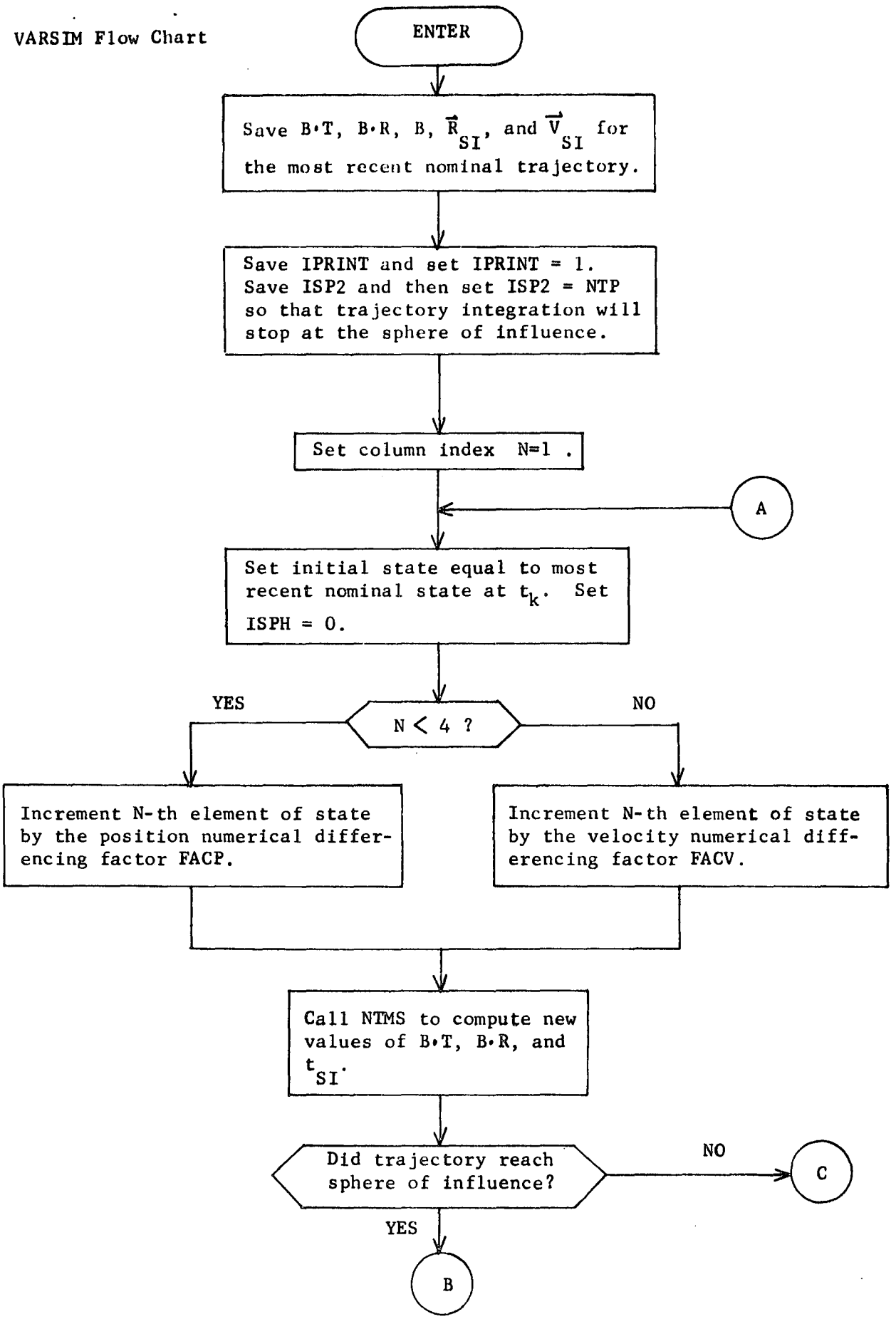
Since no good analytical formulas which relate δt_{SI} to $\delta\vec{R}_k$ and $\delta\vec{V}_k$ exist, numerical differencing must be employed to compute η .

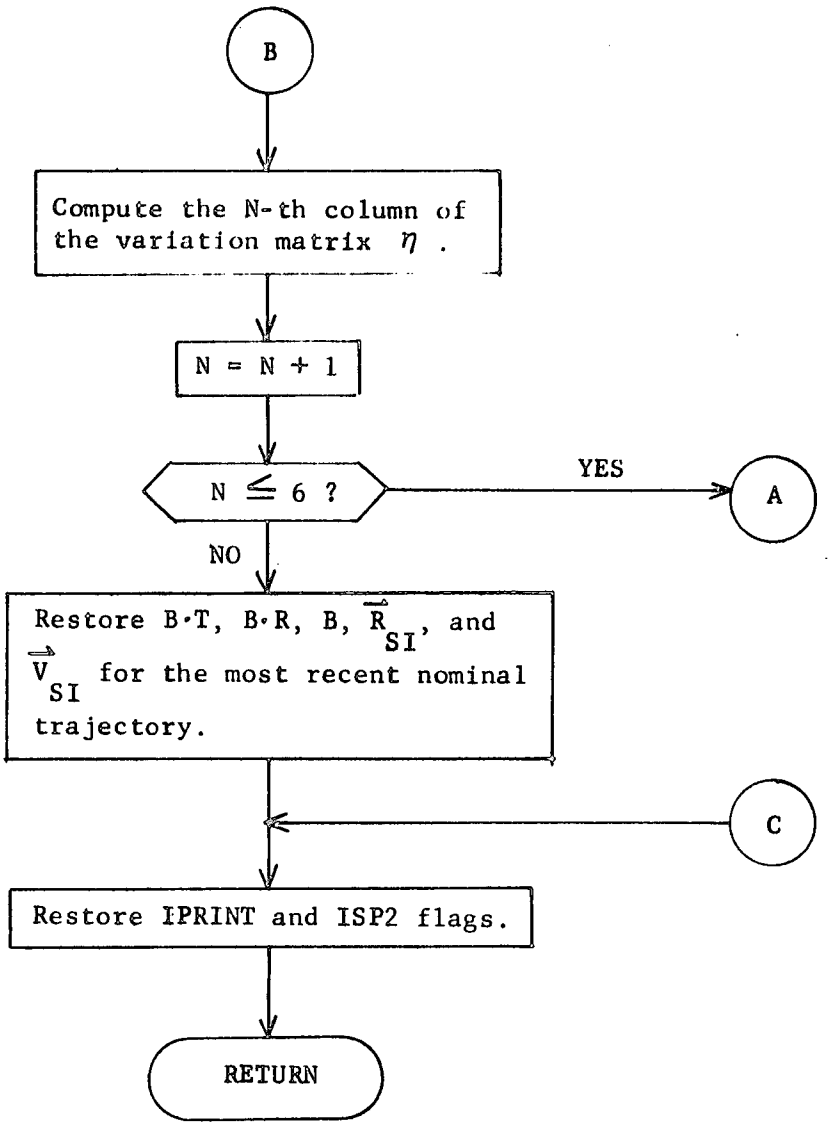
Let $\vec{\eta}_j$ be the j-th column of the matrix η , and assume (most recent) nominal $B\cdot T^*$, $B\cdot R^*$, t_{SI}^*, and $\vec{X}_k^*$ are available. To obtain $\vec{\eta}_j$ we increment the j-th element of $\vec{X}_k^*$ by the numerical differencing factor ΔX_j and numerically integrate the spacecraft equations of motion from t_k to the sphere of influence of the target planet to obtain the new values of B·T, B·R, and t_{SI} . Then

$$\vec{\eta}_j = \left[\frac{B\cdot T - B\cdot T^*}{\Delta X_j} , \frac{B\cdot R - B\cdot R^*}{\Delta X_j} , \frac{t_{SI} - t_{SI}^*}{\Delta X_j} \right]^T$$

$$j = 1,2,\ldots, 6$$

VARSIM Flow Chart




B
Compute the N-th column of the variation matrix η .
N = N + 1
N ≤ 6 ?
YES
A
NO
Restore B·T, B·R, B, $\vec{R}_{SI}$, and $\vec{V}_{SI}$ for the most recent nominal trajectory.
C
Restore IPRINT and ISP2 flags.
RETURN

VECTØR Analysis

The Kepler vector $\vec{k}$ representing twice the areal rate of the spacecraft with respect to the virtual mass to be used during the current interval is computed from

$$\vec{k} = \vec{r}_{VS_B} \times \dot{\vec{r}}_{VS_B} \tag{1}$$

where the position and velocity vectors are referenced to the virtual mass at the beginning of the interval. The eccentricity vector for the interval is given by

$$\vec{e} = -\frac{\vec{r}_{VS_B}}{r_{VS_B}} - \frac{\vec{k} \times \vec{r}_{VS_B}}{\bar{\mu}_V} \tag{2}$$

where $\bar{\mu}_V$ is the average value of the virtual mass during the interval.

The current time interval is computed from

$$\Delta\tau = \Delta t + K \Delta t^2 \tag{3}$$

where the factor K was precomputed during the previous iterations. The direction of the final position $\bar{\sigma}$ is determined from

$$\vec{\sigma} = \vec{r}_{VS_B} + \Delta\tau \, \dot{\vec{r}}_{VS_B} \tag{4}$$

The magnitude factor B is chosen to force the final position to satisfy the orbit equation ($\vec{e} \cdot \vec{r} = -r + k^2/\mu$)

$$B = \frac{k^2/\bar{\mu}_V}{\vec{e} \cdot \sigma + |\sigma|} \tag{5}$$

The position and velocity vectors of the spacecraft relative to the virtual mass at the end of the interval are then

$$\vec{r}_{VS_E} = B\vec{\sigma}$$

$$\dot{\vec{r}}_{VS_E} = \frac{\bar{\mu}_V}{k^2} \left[\vec{k} \times \left(\vec{e} + \frac{\vec{r}_{VS_E}}{r_{VS_E}} \right) \right] \tag{6}$$

The final position and velocity of the spacecraft in the reference inertial coordinates are computed from

$$\begin{aligned} \vec{r}_{S_E} &= \vec{r}_{VS_E} + \vec{r}_{V_E} \\ \dot{\vec{r}}_{S_E} &= \dot{\vec{r}}_{VS_E} + \dot{\vec{r}}_{V_E} \end{aligned} \tag{7}$$

The exact conic section time of flight is now computed. The in-plane normal to the major axis is

$$\vec{n} = \begin{cases} \dfrac{\vec{k} \times \vec{e}}{k\,e} & e \neq 0 \\ \dfrac{\vec{k} \times \vec{r}_{VS_B}}{k\, r_{VS_B}} & e = 0 \end{cases} \tag{8}$$

The length of the semi-major axis is given by

$$\begin{aligned} a &= \frac{k^2}{\bar{\mu}_V \left|1-e^2\right|^{\frac{1}{2}}} && e \neq 1 \\ a_i &= \frac{2}{r_{VS_i} - k^2/\bar{\mu}_V} && e = 1,\ i = B,E \end{aligned} \tag{9}$$

The projection of the radius vector orthogonal to the major axis divided by a is given by

$$X_i = \frac{\vec{n} \cdot \vec{r}_{VS_i}}{a_i} \qquad i = B,E \tag{10}$$

The mean angular rate is

$$\bar{\omega} = \begin{cases} \dfrac{\bar{\mu}_V(1-e^2)}{ka} & e \neq 1 \\ \dfrac{k}{2} & e = 1 \end{cases} \tag{11}$$

where $\omega < 0$ for hyperbolic orbits. The eccentric anomaly is given by

$$E_{\substack{i \\ i=B,E}} = \sin^{-1} X_i \qquad e < 1$$

$$= \frac{k^2/\bar{\mu}_V \; X_i}{3} \qquad e = 1 \tag{12}$$

$$= \sinh^{-1} X_i \qquad e > 1$$

Then $M_i = E_i - e\, X_i \qquad i = B,E$ (13)

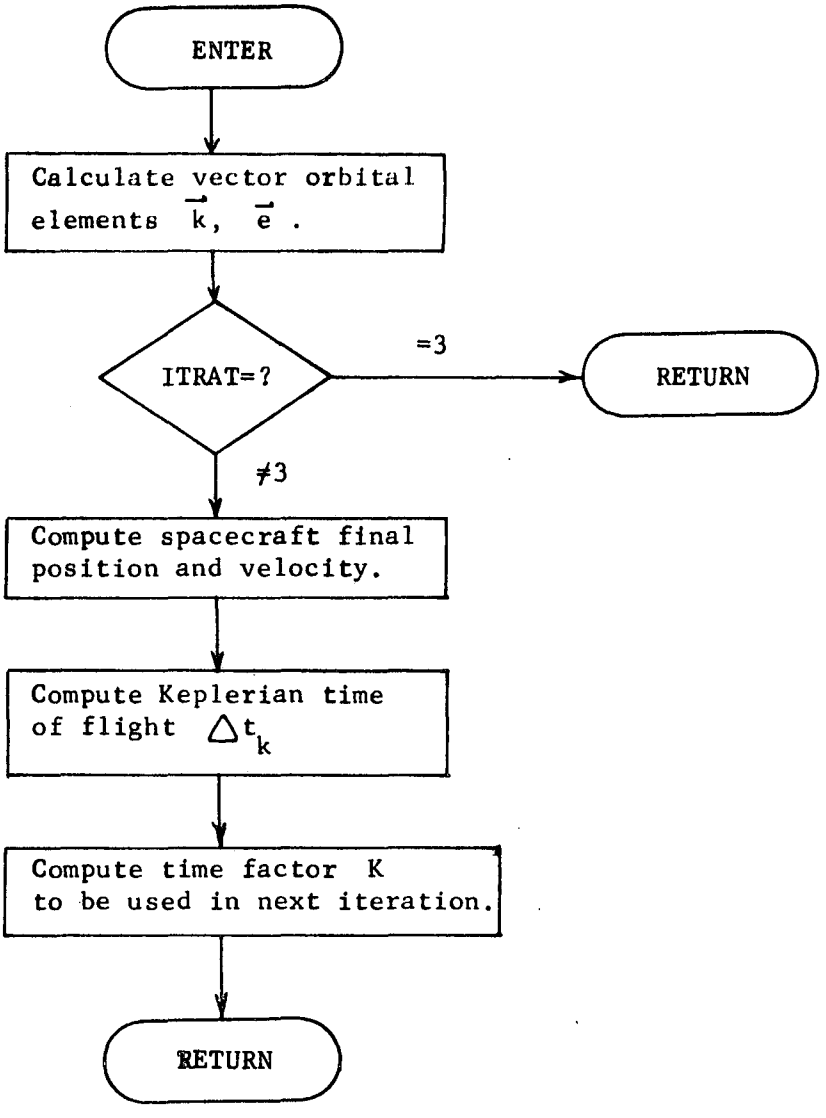
and the actual conic time of flight is

$$\Delta t = t_E - t_B = \frac{M_E - M_B}{\bar{\omega}} \tag{14}$$

The value of the time factor K to be used in the following interval is then computed

$$K = \frac{\Delta\tau - \Delta t}{(\Delta t)^2} \tag{15}$$

VECTOR Flow Chart



VMASS Analysis

The current virtual mass data is computed by VMASS. The magnitude and position of the virtual mass is given by

$$\mu_V = r_{VS}^3 M_S \tag{1}$$

$$\vec{r}_V = \frac{\vec{M}}{M_S} \tag{2}$$

where the intermediate variables are given by

$$\vec{M} = \sum_{i=1}^{n} \frac{\mu_i \vec{r}_i}{r_{iS}^3} \tag{3}$$

$$M_S = \sum_{i=1}^{n} \frac{\mu_i}{r_{iS}^3} \tag{4}$$

and of course $r_{iS} = |\vec{r}_i - \vec{r}_S|$ and $\vec{r}_{VS} = |\vec{r}_V - \vec{r}_S|$ where $\vec{r}_i$ represents the inertial position vector of the i-th body.

The time derivatives of these variables are given by

$$\dot{\mu}_V = \mu_V (\alpha_V + \frac{\dot{M}_S}{M}) \tag{5}$$

$$\dot{\vec{r}}_V = \frac{\dot{\vec{M}} - \vec{r}_V \dot{M}_S}{M_S} \tag{6}$$

$$\dot{\vec{M}} = \sum_{i=1}^{n} \frac{\mu_i}{r_{iS}^3} \left[\dot{\vec{r}}_i - \vec{r}_i \alpha_i\right] \tag{7}$$

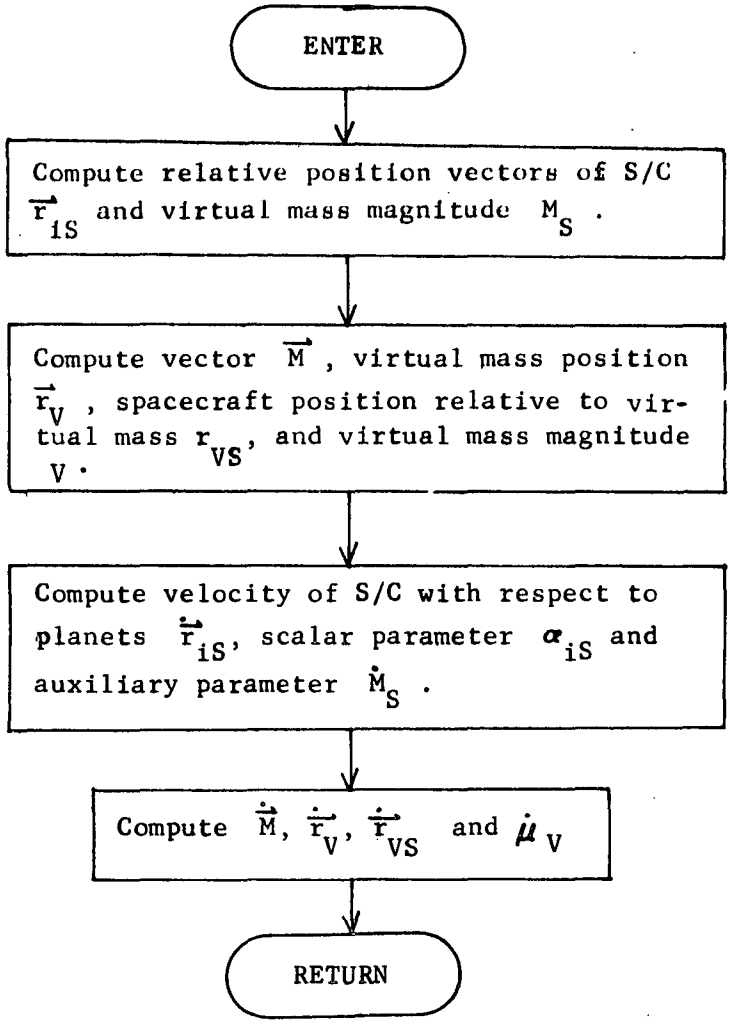
$$\dot{M}_S = -\sum \frac{\mu_i}{r_{iS}^3} \alpha_{iS} \tag{8}$$

where $$\alpha_{iS} = \frac{3\,\vec{r}_{iS} \cdot \dot{\vec{r}}_{iS}}{r_{iS}^{2}} \qquad (9)$$

Finally, the velocity of the spacecraft with respect to the virtual mass is

$$\dot{\vec{r}}_{VS} = \dot{\vec{r}}_{S} - \dot{\vec{r}}_{V} \qquad (10)$$

VMASS Flow Chart



VMP Analysis

VMP provides the logic to integrate an N-body trajectory from an initial spacecraft state ($\bar{r}_S$, $\bar{v}_S$) at time t_B to one of the following stopping conditions.

1. Target planet sphere of influence (SOI) is reached (ISP2 $\neq$ 0).

2. The closest approach to the target planet has been reached (ICL2 = 1).

3. The preset final trajectory time t_F has been exceeded.

The integration logic is controlled by ITRAT

ITRAT = 1 First pass through computation cycle (including ephemeris computation).

2 Second pass through computation cycle (excluding ephemeris).

3 Initialization flag.

To start the integration, appropriate variables are initialized (PRINTZ) and ITRAT is set equal to 3. The state of all gravitational bodies at t_B are found (ORB, EPHEM). The initial virtual mass position $\bar{r}_{V_B}$, velocity $\bar{v}_{V_B}$, magnitude μ_{V_B} and magnitude rate $\dot{\mu}_{V_B}$ are found by VMASS. Virtual mass dependent values are then initilized

$$\mu_{V_{AVE}} = \mu_{V_E} = \mu_{V_B} \tag{1}$$

$$\dot{\bar{r}}_{V_{AVE}} = \dot{\bar{r}}_{V_B} \tag{2}$$

$$\bar{r}_{VS_E} = \bar{r}_{VS_B} \tag{3}$$

$$\dot{\bar{r}}_{VS_E} = \dot{\bar{r}}_{VS_B} \tag{4}$$

$$(\Delta t)^2 = 1 \tag{5}$$

$$\text{ISPH1} = 0 \tag{6}$$

At this point the standard integration routine is entered by calling VECTOR.

In the standard integration routine, a new increment is initiated by calling ESTMT which:

1. Initializes all appropriate variables at the beginning of the increment (subscript B) to equal their values at the end of the previous increment.

2. Computes a Δt for the increment based on a modified true anomaly passage.

3. Computes the time at the end of the increment t_E.

4. Estimates the final (subscript E) position $\bar{r}_{V_E}$ and magnitude μ_{V_E} of the virtual mass.

Based on these estimates, the average magnitude and velocity of the virtual mass is computed

$$\mu_{V_{AVE}} = 1/2\ (\mu_{V_B} + \mu_{V_E}) \tag{7}$$

$$\bar{v}_{V_{AVE}} = (\bar{r}_{V_E} - \bar{r}_{V_B})/\Delta t \tag{8}$$

Subroutine VECTOR then computes the orbit relative to the virtual mass based on these estimates. It also refines the estimate of the spacecraft final state ($\bar{r}_{S_E}$, $\bar{v}_{S_E}$). ORB and EPHEM are called to determine the state at t_E of all gravitational bodies being considered. The virtual mass position $\bar{r}_{V_E}$, velocity $\bar{v}_{V_E}$, magnitude μ_{V_E} and magnitude rate $\dot{\mu}_{V_E}$ are determined by VMASS.

Using these refined values, the virtual mass average magnitude $\mu_{V_{AVE}}$ and velocity $\bar{v}_{V_{AVE}}$ are recomputed using equations (7) and (8). At this point a second pass is made through VECTOR to compute the spacecraft final state ($\bar{r}_{S_E}$, $\bar{v}_{S_E}$) which will be used in all subsequent calculations. VMASS is again called to make a final determination of the virtual mass

position, velocity, magnitude and magnitude rate at the end of the increment.

The virtual mass average accelerations are then computed

$$\ddot{\mu}_{V_{AVE}} = \left[\dot{\mu}_{V_E} - \dot{\mu}_{V_B} - \ddot{\mu}_{V_B}\,(\Delta t)\right] \Big/ (\Delta t)^2 \qquad (9)$$

$$\dot{\bar{v}}_{V_{AVE}} = \left[\bar{r}_{V_E} - \bar{r}_{V_B} - \bar{v}_{V_B}\,(\Delta t)\right] \Big/ (\Delta t)^2 \qquad (10)$$

These values are subsequently used by ESTMT to estimate the final position $\bar{r}_{VE}$ and magnitude μ_{VE} of the virtual mass for the next increment.

Tests are now made to determine whether the vehicle is inside a planetocentric sphere of radius 1.025 times larger than that of the SØI. If it is not, integration goes on as usual. If it is and yet is still outside the SØI, the integration step size is reduced to obtain an integration state near enough the SØI to permit accurate extrapolation to it. Finally, if the vehicle is inside of the sphere of influence, a refined sphere of influence (SØI) state is constructed by fitting an osculating planetocentric conic to the current state and extrapolating it to the sphere. The entire refinement process is carried out in subroutine SØIPS.

The refined state at the SØI is then used by IMPACT to compute B·T and B·R.

If trajectory data are to be printed at this point, the orbit inclination (assuming a hyperbolic orbit about the planet) is computed by first determining the "Kepler vector"

$$k = \bar{r}_{ST} \times \bar{v}_{ST} \qquad (11)$$

in planetocentric equatorial coordinates. Then

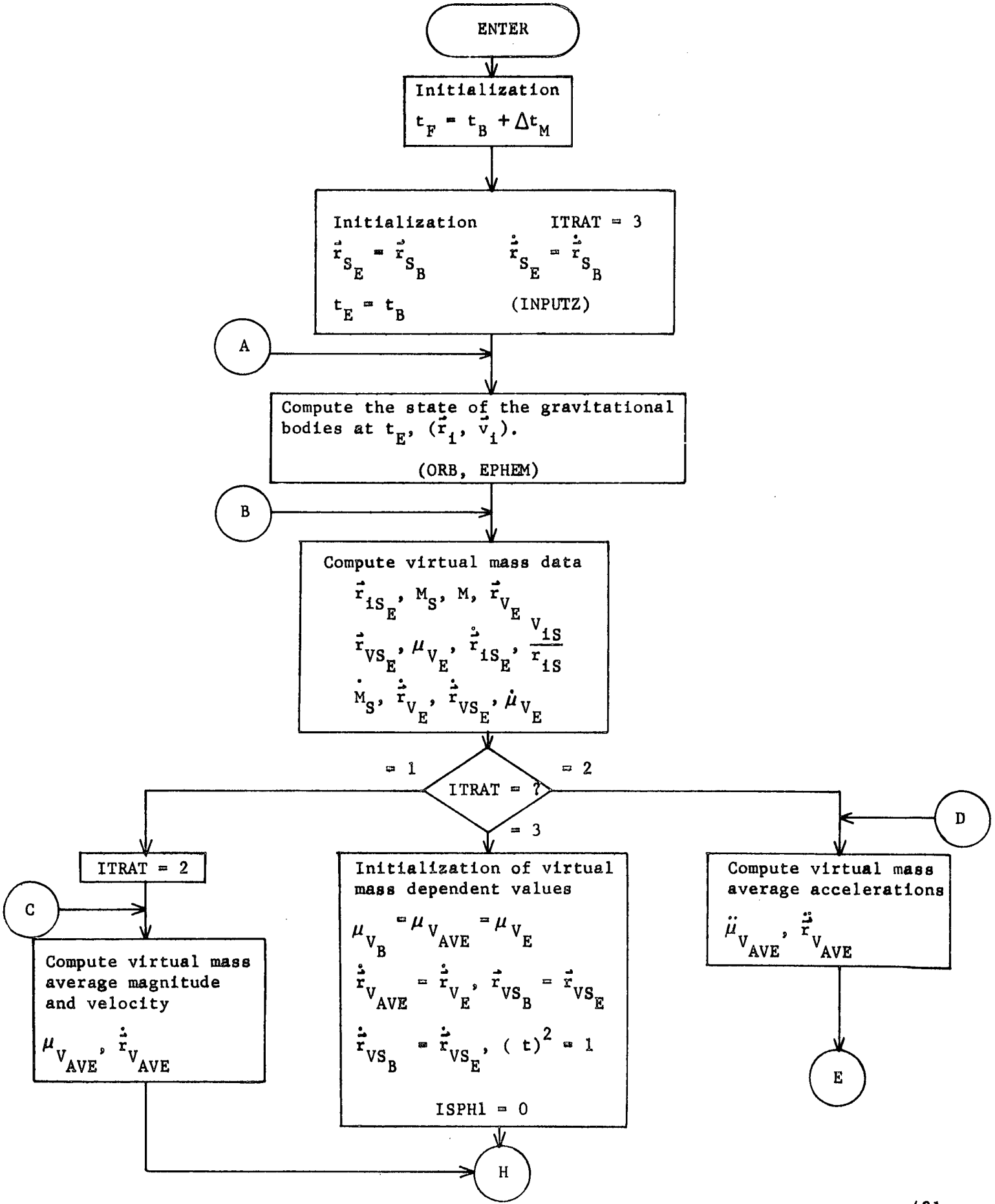
$$\cos i = \frac{k_z}{|\vec{k}|} \qquad (12)$$

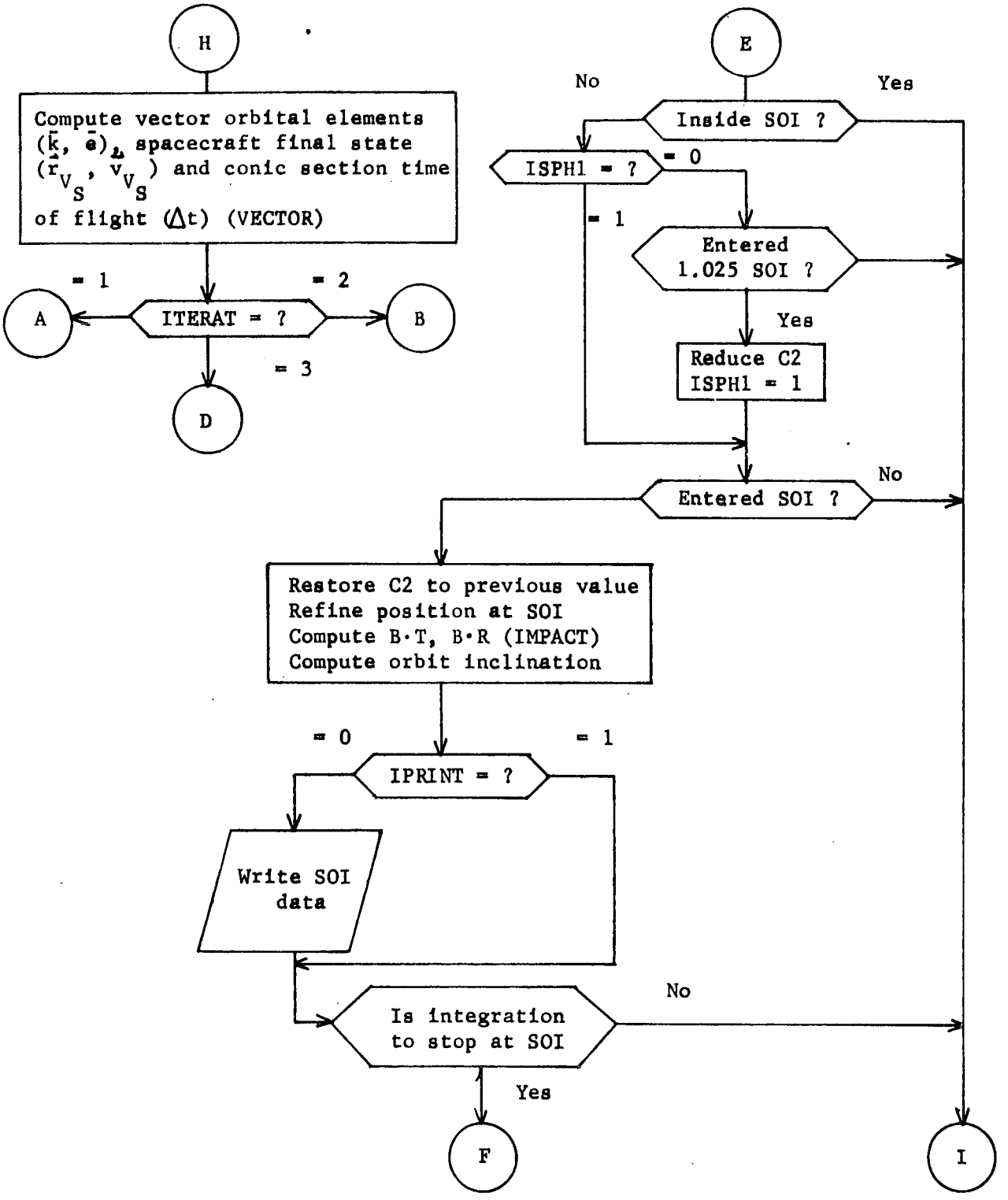
where i = orbit inclination and $\vec{k}_z$ = component of k normal to planet equatorial plane.

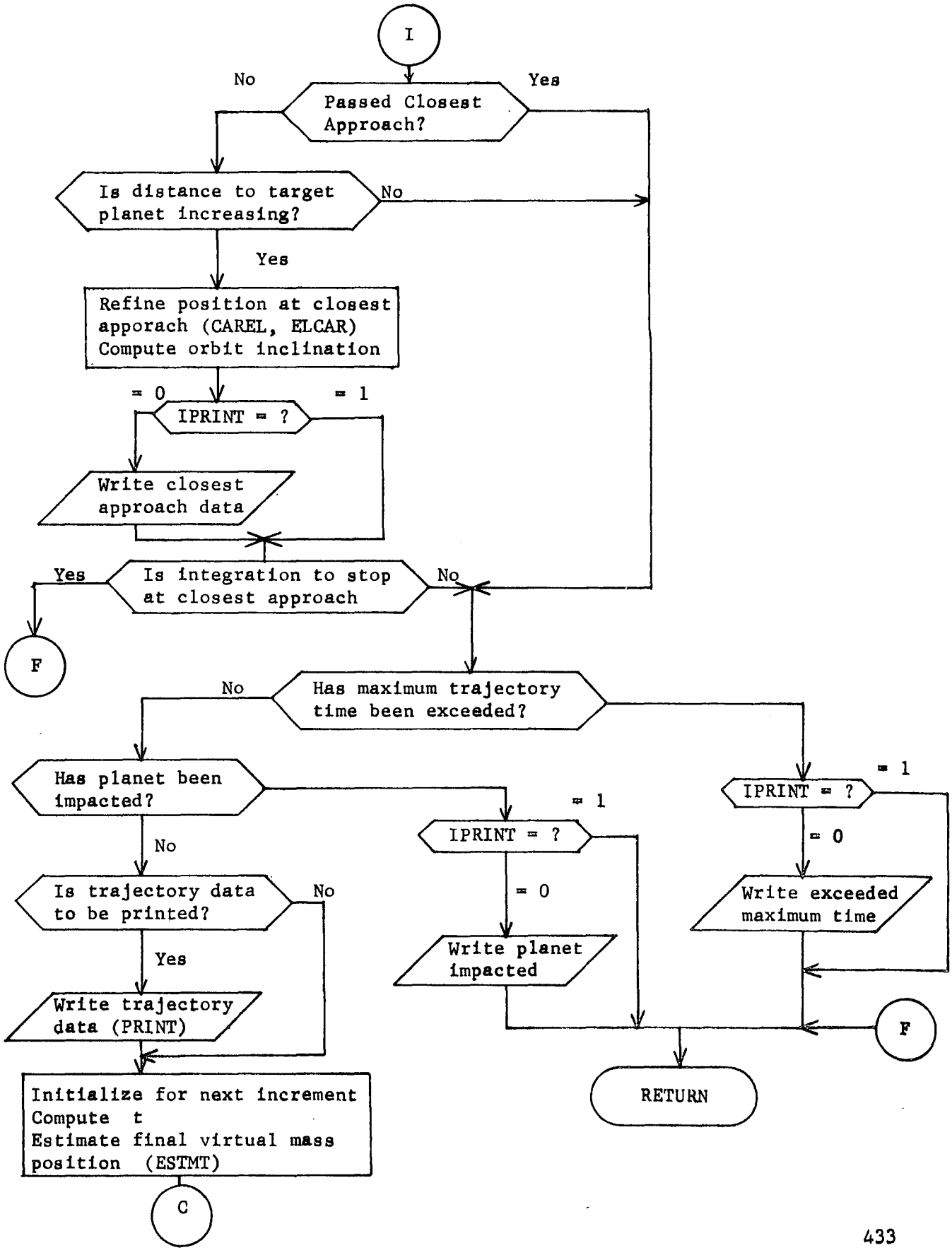
Tests are now made to determine if the spacecraft has reached a closest approach to the target planet. If it has, the interpolated state at closest approach $\left(\bar{r}_{CA}, \bar{v}_{CA}\right)$ is computed by calling CAREL with the spacecraft state just following closest approach. CAREL returns the element of the near planet conic. ELCAR is then called with these conic elements and returns the interpolated state at closest approach.

If the spacecraft is not within 10 SOI of the target planet, printout of closest approach data may occur; however, integration continues.

The final tests before starting a new integration increment determine if the maximum trajectory time t_F has been exceeded or a planet has been impacted. If the latter has occurred, the actual impact state is determined by fitting an osculating planetocentric conic to the current state and extrapolating to the planet surface. As was the case earlier at the SOI, this procedure is carried out in the subroutine SØIPS. If the impacted planet is the object of any type of probe targeting, the planet radius used in the impact state computation is the probesphere value input by the user. If these final two tests are passed, a new integration cycle is initiated by calling ESTMT.


ENTER
Initialization
$t_F = t_B + \Delta t_M$
Initialization
ITRAT = 3
$\vec{r}_{S_E} = \vec{r}_{S_B}$
$\dot{\vec{r}}_{S_E} = \dot{\vec{r}}_{S_B}$
$t_E = t_B$
(INPUTZ)
A
Compute the state of the gravitational bodies at t_E, $(\vec{r}_i, \vec{v}_i)$.
(ORB, EPHEM)
B
Compute virtual mass data
$\vec{r}_{iS_E}$, M_S, M, $\vec{r}_{V_E}$
$\vec{r}_{VS_E}$, μ_{V_E}, $\dot{\vec{r}}_{iS_E}$, $\frac{V_{iS}}{r_{iS}}$
$\dot{M}_S$, $\dot{\vec{r}}_{V_E}$, $\dot{\vec{r}}_{VS_E}$, $\dot{\mu}_{V_E}$
= 1
ITRAT = ?
= 2
= 3
D
ITRAT = 2
C
Compute virtual mass average magnitude and velocity
$\mu_{V_{AVE}}$, $\dot{\vec{r}}_{V_{AVE}}$
Initialization of virtual mass dependent values
$\mu_{V_B} = \mu_{V_{AVE}} = \mu_{V_E}$
$\dot{\vec{r}}_{V_{AVE}} = \dot{\vec{r}}_{V_E}$, $\vec{r}_{VS_B} = \vec{r}_{VS_E}$
$\dot{\vec{r}}_{VS_B} = \dot{\vec{r}}_{VS_E}$, $(t)^2 = 1$
ISPH1 = 0
Compute virtual mass average accelerations
$\ddot{\mu}_{V_{AVE}}$, $\ddot{\vec{r}}_{V_{AVE}}$
E
H


H
Compute vector orbital elements ($\bar{k}$, $\bar{e}$), spacecraft final state (r_{V_S}, v_{V_S}) and conic section time of flight (Δt) (VECTOR)
= 1
= 2
= 3
ITERAT = ?
A
B
D
E
No
Yes
Inside SOI ?
ISPH1 = ?
= 0
= 1
Entered 1.025 SOI ?
Yes
Reduce C2
ISPH1 = 1
Entered SOI ?
No
Restore C2 to previous value
Refine position at SOI
Compute B·T, B·R (IMPACT)
Compute orbit inclination
= 0
= 1
IPRINT = ?
Write SOI data
Is integration to stop at SOI
No
Yes
F
I

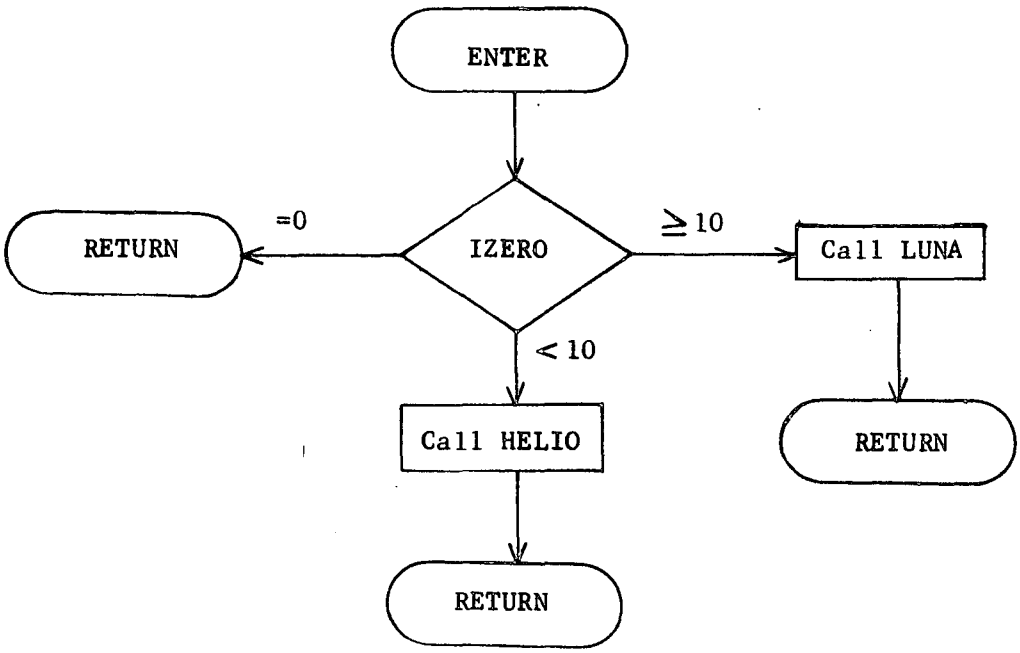

I
No
Yes
Passed Closest Approach?
Is distance to target planet increasing?
No
Yes
Refine position at closest apporach (CAREL, ELCAR)
Compute orbit inclination
= 0
= 1
IPRINT = ?
Write closest approach data
Yes
Is integration to stop at closest approach
No
F
No
Has maximum trajectory time been exceeded?
Has planet been impacted?
= 1
IPRINT = ?
= 1
IPRINT = ?
No
= 0
Write exceeded maximum time
Is trajectory data to be printed?
No
= 0
Write planet impacted
Yes
Write trajectory data (PRINT)
F
Initialize for next increment
Compute t
Estimate final virtual mass position (ESTMT)
RETURN
C

ZERIT Analysis

ZERIT is the executive subroutine handling the computation of the zero iterate values of time, position vector, and velocity vector.

The flag IZERO controls the operation of ZERIT. If IZERO = 0, no zero iterate computation is needed and so ZERIT is exited. If IZERO $<$ 10, the zero iterate is to be computed for a interplanetary trajectory so HELIO is called before returning. If IZERO $\geq$ 10, the zero iterate is to be computed for a lunar trajectory so LUNA is called for that computation.

ZERIT Flow Chart



BIBLIOGRAPHY

1. Baker, R.M.L. and Makemson, M. W.: An Introduction to Astrodynamics, Academic Press, New York, 1967.

2. Battin, R.H.: Astronautical Guidance, McGraw-Hill Book Company, Inc., New York, 1964.

3. Battin, R.H. and Fraser, D.C.: Space Guidance and Navigation, AIAA Professional Study Series, 1970.

4. Byrnes, D.V. and Hooper, H.L.: Multiconic: A Fast and Accurate Method of Computing Space Flight Trajectories, AAS/AIAA Astrodynamics Conference, Santa Barbara, Cal., 1970, AIAA Paper 70-1062.

5. Danby, J.M.A.: Matrizant of Keplerian Motion, AIAA J., vol 3, no. 4, April, 1965.

6. Hoffman, L.H. and Young, G.R.: Approximation to the Statistics of Midcourse Velocity Corrections, NASA TN D-5381, Clearinghouse for Federal Scientific and Technical Information, Springfield, Virginia.

7. Jazwinski, A.H.: Adaptive Filtering, Analytical Mechanics Associates, Lanham, Maryland, May 1968.

8. Joseph, A.E. and Richard, R.J.: Space Research Conic Program, Engineering and Planning Document 406, Jet Propulsion Laboratory, July 1966.

9. Kizner, W.: A Method of Describing Miss Distances for Lunar and Interplanetary Trajectories Under the Influence of Multiple Planet Attractions, TR-32-464, Jet Propulsion Laboratory, 1963.

10. Lee, B.G., Falce, R.R., and Hopper, F.: Interplanetary Trajectory Error Analysis, Volume I - Analytical Manual, Vol II - Computer Program Design, MCR-67-441, Martin Marietta Corporation, Denver, Colorado, December 1967. (Completed under Contract NAS8-21120)

11. Lee, B.G., Vogt, E.D., Falce, R.R., Pearson, S., and Demlow, E.: Simulated Trajectories Error Analysis Program, Vol I - Users Manual, Vol II - Analytical Manual, NASA CR-66818, Martin Marietta Corporation, Denver, Colorado. (Completed under Contract NAS1-8745)

12. Lesh, H.F. and Travis, C.: FLIGHT: A Subroutine to Solve the Flight Time Proglem, JPL Space Programs Summary 37-53, Vol II.

13. Mitchell, R.T. and Wong, S.K.: Preliminary Flight Path Analysis, Orbit Determination and Maneuver Strategy, Mariner Mars 1969, Project Document 138, Jet Propulsion Laboratory, 1968.

14. Myers, G.E.: Properties of the Conjugate Gradient and Davidon Methods, AAS Paper 68-081. Presented at 1968 AAS/AIAA Astrodynamics Specialist Conference, Jackson, Wyoming.

15. Novak, D.H.: Virtual Mass Technique for Computing Space Trajectories, ER-14045, Martin Marietta Corporation, Denver, Colorado, January 1966. (Completed under Contract NAS9-4370)

16. Scheid, F.: Theory and Problems of Numerical Analysis, McGraw-Hill Book Company, Inc., New York, 1968.

17. Sorenson, H.: Kalman Filtering, Advances in Control Systems, Vol 3, C.T. Leondes (Ed.), Academic Press, New York, 1966.

18. Wilde, D.J. and Beightler, C.S.: Foundations of Optimization, Englewood Cliffs, N.J., Prentice Hall, 1967.