NASA CONTRACTOR REPORT

NASA CR-793

STOP — A COMPUTER PROGRAM FOR SUPERSONIC TRANSPORT TRAJECTORY OPTIMIZATION

by Lawrence H. Stein, Malcolm L. Matthews, and Joel W. Frenk

Prepared by
THE BOEING COMPANY
Seattle, Wash.
for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • MAY 1967

NASA CR-793

[redacted]P — A COMPUTER PROGRAM FOR SUPERSONIC TRANSPORT TRAJECTORY OPTIMIZATION

By Lawrence H. Stein, Malcolm L. Matthews,
and Joel W. Frenk

Prepared under Contract No. NAS 1-5293 by
THE BOEING COMPANY
Seattle, Wash.

for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – CFSTI price $3.00

ABSTRACT

An IBM 7094 digital computer program using the steepest ascent technique has been developed for optimizing the flight path of a supersonic transport aircraft from the start of climb through cruise and descent. The program is sufficiently versatile so that other vehicles besides the SST may have their flight paths optimized. These vehicles include: space boosters, ICBM's, reentry vehicles, scramjets, and vehicles with air-augmented rocket propulsion.

The program incorporates a 3-D point mass simulation of a vehicle moving in relation to a spherical, rotating earth. The inverse-square law for gravity and 1962 U. S. standard atmosphere are used.

The optimization is accomplished with an automatic step-size controller and with automatic control variable weighting matrices to allow problem solution in a single computer run. Automatic plotting capability is included. Multistaged vehicles and problems involving variable initial conditions may be optimized.

Pitch angle, bank angle, wing sweep, and throttle setting are the control variables for the program. Inequality constraints are available on all control variables as well as on parameters affecting geopolitics, passenger comfort, structural loads, and engine operation. The geopolitical constraints include sonic boom overpressure and maximum and minimum altitude.

FOREWORD

This report was prepared by the Missile and Information Systems Division of The Boeing Company, Seattle, Washington. It presents the final documentation of the analytical development and user's manual for the Supersonic Transport Optimization Program (STOP). The program was developed by Boeing for the Langley Research Center under contract NAS 1-5293. The contract was administered by the National Aeronautics and Space Adminstration under the direction of Mr. J. R. Elliott, with Mr. David F. Thomas, Jr. acting as contract monitor.

The Supersonic Transport Optimization Program was obtained from the NASA Request for Proposal L-5347. Development of the program began in August 1965 and was completed in September 1966. Dr. L. H. Stein was responsible for program development. The closed loop guidance techniques and the control formulation were developed by Mr. M. L. Matthews. Mr. Matthews also assisted in a major part of the program development as well as in the solution of the twenty check cases. Mr. J. W. Frenk was in charge of programming and was assisted by Mr. D. A. Watson. Mr. Watson was also responsible for the plotting capability for the program. The work was performed under the direction of Mr. E. G. Haugseth of the Boeing Missile and Information Systems Division.

This report, together with a companion document, Supersonic Transport Trajectory Optimization - Example Solutions (NASA Contractor Report No. 66247) plus the FORTRAN source program listings, binary object deck, and symbolic object deck concludes the work prescribed under contract NAS 1-5293.

CONTENTS

CONTENTS (Cont.)

CONTENTS (Cont.)

STOP— A COMPUTER PROGRAM FOR SUPERSONIC TRANSPORT TRAJECTORY OPTIMIZATION

By Lawrence H. Stein, Malcolm L. Matthews, and Joel W. Frenk
Boeing Aerospace Group

SUMMARY

An IBM 7094 digital computer program using a steepest-ascent procedure has been developed for optimizing the flight path of supersonic transport aircraft from the start of climb through cruise and descent. This document describes the analytical development of the supersonic transport optimization program (STOP) and presents a manual for program users.

Program capability is summarized below.

1. The program is capable of optimizing the entire flight of an SST from a given low-speed, low-altitude condition at the start of the flight to a given low-speed, low-altitude condition at the end of the flight. The program will also optimize the climb, cruise, and descent phases of a mission separately.
2. The program incorporates a 3-D point-mass simulation for a vehicle moving in relation to a spherical, rotating earth. The inverse-square law for gravity is used. The 1962 U.S. standard atmosphere is included as a single subroutine.
3. The optimization is accomplished with an automatic step-size controller so that, in general, only one pass on the computer is required for a solution. The program will generate a nominal trajectory for starting the iterative procedure. Automatic plotting capability is included. STOP is sufficiently versatile so that other vehicles besides the SST may have their flight paths optimized. These vehicles include: space boosters, ICBM's, reentry vehicles, scramjets, and vehicles with air-augmented propulsion. Even problems completely divorced from flight path optimization may be solved with a minimum of reprogramming.
4. Payoff functions, terminal constraints, and stopping parameters may be selected from the list of 40 state variables (defined by equations of motion and enroute placards) given in figure 1. The program will optimize any one of the state variables while simultaneously satisfying 14 terminal constraints (one of which is considered to be the stopping condition). Inequality constraints, imposed by the user, are considered as terminal constraints.
5. Pitch angle, bank angle, wing sweep, and throttle setting are the control variables for the program. The user may select any subset of these variables for a given problem.

VARIABLE TYPE	VARIABLE INDEX	NC INDEX	PLOT INDEX	FORTRAN NAME	SYMBOL	DESCRIPTION	UNITS
STATE (EQUATION OF MOTION)	1	121	1	X(K1)	W	WEIGHT	LB
	2	122	2	X(K2)	H	ALTITUDE	FT
	3	123	3	X(K3)	γ	RELATIVE FLIGHT PATH ANGLE	DEG
	4	124	4	X(K4)	V	RELATIVE VELOCITY	FPS
	5	125	5	X(K5)	β	LATITUDE ANGLE	DEG
	6	126	6	X(K6)	ψ R	RELATIVE HEADING ANGLE	DEG
	7	127	7	X(K7)	λ	LONGITUDE ANGLE	DEG
	8	128	8	X(K8)	TD	DUMMY TIME	SEC
	9	129	9	X(K9)	RNG	PATH RANGE ALONG EARTH'S SURFACE	N MI
	10	130	10	X(K10)	AHI	AERODYNAMIC HEATING INTEGRAL	FT-LB/FT^2
	11	131	11	X(K11)	Δ V	IDEAL RELATIVE DELTA VELOCITY	FPS
	12	132	12	X(K12)	GL	GRAVITY LOSS	FPS
	13	133	13	X(K13)	DL	DRAG LOSS	FPS
	14	134	14	X(K14)	TVL	THRUST VECTORING LOSS	FPS
	15	135	15	X(K15)	ER	RELATIVE SPECIFIC ENERGY	FT
	16	136	16	X(K16)			
	17	137	17	X(K17)		AVAILABLE FOR EXPANSION NOT DEFINED	
	18	138	18	X(K18)			
	19	139	19	X(K19)			
	20	140	20	X(K20)			
STATE (ENROUTE PLACARD)	21	141	21	X(K21)	θ*	PITCH ANGLE PLACARD = F(t)	DEG^{2}SEC
	22	142	22	X(K22)	ϕ*	BANK ANGLE PLACARD = F(t)	DEG^{2}SEC
	23	143	23	X(K23)	η*	THROTTLE PLACARD = F(H, M)	SEC$_2$
	24	144	24	X(K24)	Λ*	WING SWEEP PLACARD = F(H, M)	DEG^{2}SEC
	25	145	25	X(K25)	α*	ANGLE OF ATTACK PLACARD = F(H, M)	DEG^{2}SEC
	26	146	26	X(K26)		NOT DEFINED	
	27	147	27	X(K27)	HD*	ALTITUDE RATE PLACARD = F(t)	FPS
	28	148	28	X(K28)	Q*	DYNAMIC PRESSURE PLACARD = F(t)	(PSF)^2 SEC
	29	149	29	X(K29)	Q*	DYNAMIC PRESSURE PLACARD = F(M)	(PSF)^{2}SEC
	30	150	30	X(K30)	Qα*	Qα PLACARD = F(M)	(PSF-DEG)^{2}SEC
	31	151	31	X(K31)	TEMT*	TOTAL TEMPERATURE PLACARD = F(t)	(°R)^{2}SEC
	32	152	32	X(K32)	N*	NORMAL LOAD FACTOR PLACARD = F(H, M)	SEC
	33	153	33	X(K33)	RPA*	RESULTANT PHYSIOLOGICAL ACCEL. PLACARD = F(t)	(FT/SEC^2)^{2}SEC
	34	154	34	X(K34)	H*	ALTITUDE PLACARD = F(M)	FT^{2}SEC
	35	155	35	X(K35)	ΔP*	SONIC BOOM OVERPRESSURE PLACARD = F(λ, β)	(PSF)^{2}SEC
	36	156	36	X(K36)	M*	MACH NUMBER PLACARD = F(H)	SEC
	37	157	37	X(K37)		NOT DEFINED	
	38	158	38	X(K38)	γ*	GAMMA PLACARD = F(H, M)	DEG^2 SEC
	39	159	39	X(K39)		NOT DEFINED	
	40	160	40	X(K40)		NOT DEFINED	

FIGURE 1: STATE VARIABLES

6. Aerodynamic and engine options are available for receiving the data in the forms commonly used for most types of vehicles.

7. Multistaged vehicles and those where external stores are jettisoned as a function of time may be optimized.

8. Inequality constraints may be imposed on parameters affecting geopolitics, passenger comfort, control limitations, structural loads, and engine operation. The geopolitical constraints include sonic boom overpressure and maximum and minimum altitude.

9. The initial conditions can be varied by the program to obtain increased performance. This option eliminates the need for a preliminary search to determine the neighborhood of initial conditions for an optimal flight path.

INTRODUCTION

The need to optimize transport aircraft flight paths is becoming apparent because (1) the next generation will include supersonic transports for which the climb and acceleration phase consumes an appreciable part of the vehicle fuel, and (2) a large part of the flight is constrained by enroute placards.

The steepest-ascent method has been successfully used to optimize rocket-boost trajectories, reentry-vehicle trajectories, and orbital transfers giving substantial performance gains. There have been numerous attempts to optimize flight paths of airbreathing interceptor-type aircraft with a relatively high thrust-to-weight ratio using the steepest-ascent technique (ref. 1). These methods have been successful to a degree, but, when applied to low-thrust-to-weight-ratio aircraft, have resulted in flight path instability problems that have made convergence difficult.

Recognizing that, in the past, optimization techniques have produced significant performance gains for many classes of vehicles, NASA initiated the present study under RFP L-5347 (ref. 2). The purpose of the study was development of a digital computer program that would optimize the flight path of a supersonic transport with realistic operational constraints. A sufficient number of cases would be run to substantiate the program and define the most appropriate flight paths for selected SST configurations. A computer program general enough to optimize the SST would have the capability to optimize trajectories for many classes of vehicles including rockets, air-augmented rockets, airbreather vehicles, and gliders.

The technique used for optimization is the steepest-ascent method given by Bryson (ref. 1), which follows the direct approach of determining a maximum. The optimum flight path for the present study is the solution to the nonlinear differential equations of motion that satisfy the imposed constraints and maximizes or minimizes one of the state variables.

The development of a computer program to satisfy the SST requirements and overcome problems associated with the flight path instabilities and complex engine characteristics required some changes in the usual methods used in optimization programs. Significant contributions to the program were made by NASA/LRC personnel. The use of pitch angle as a control variable was suggested by J. R. Elliott as a technique to overcome the flight path instabilities. The formulation of jet engine data into a form compatible with the optimization procedure was suggested by W. E. Foss, Jr. This form of data uses thrust input as T/P and weight flow as $\dot{W}/P$, where P is the ambient atmospheric pressure. Contribution to the form of the SST aerodynamic data representation was made by C. M. Jackson, Jr.

SYMBOLS

A	axial force (pounds)
a	speed of sound (feet/second)
$\bar{a}$	inertial acceleration vector (feet/second^2)
$A\ (t - \tau)$	indicial response at time t due to a unit step in f(t) at time τ (see Duhamel's integral, equation 81)
A_e	nozzle exit area (square inches)
AHI	aerodynamic heating integral (foot-pounds/feet^2)
a_n	component of acceleration normal to relative velocity vector
a_t	component of acceleration along the velocity vector
a_p	component of acceleration completing the right-hand set $(a_n, a_t, a_p) = \bar{a}$
C_A	axial force coefficient, A/q S
C_D	drag force coefficient, D/q S
C_{D_0}	minimum drag force coefficient
C_L	lift force coefficient, L/q S
C_{L_α}	$\partial C_L/\partial\alpha$ (per degree)
$C_{L_{\alpha=0}}$	lift force coefficient at $\alpha = 0$
C_{L_M}	lift force coefficient for minimum drag
C_N	normal force coefficient, N/q S
C_{N_α}	$\partial C_N/\partial\alpha$ (per degree)
$C_{N_\alpha 3}$	$\partial C_N/\partial\alpha^3$ (per degree^3)
C_T	thrust force coefficient, T/q (square feet)
D	drag force (pounds)
D_0, D_1, D_2	drag constants (defined by equation 151)
DL	velocity loss caused by drag forces (feet/second)
$(dU)^2$	control variable perturbation magnitude (see equation 100)
df	predicted change in a function f (see fig. 8)
df_o	trial value for df
df^*	actual change in a function f (see fig. 8)
df_o^*	trial value for df^*

$d\bar{\beta}$	combined changes in the constraints and initial state variables
$d\varphi$, $d\bar{\psi}$, $d\Omega$	change in payoff, constraint, and stopping functions
$d\varphi_o$, $d\bar{\psi}_o$	trial values for $d\varphi$ and $d\bar{\psi}$
F, F(t)	n x n matrix of partial derivatives, $\partial f_i/\partial x_j$
f, $f\left(X(t), \bar{u}(t), t\right)$	an n x 1 matrix of functions defining the time derivatives of the state variables
f_{NL}	a value determining the degree of nonlinearity of the function f
f(t)	forcing function of Duhamel's integral (see equation 81)
G, G(t)	n x m matrix of partial derivatives, $\partial f_i/\partial u_j$
g	local acceleration of gravity (feet/second)
g(t)	forcing function of the convolution integral (see equation 99)
g_o	gravitational acceleration at earth's surface (feet/second^2)
GL	velocity loss due to earth's gravitational field (feet/second)
GF	function defined by equation 140
h	altitude above earth's surface (feet)
h_{ref}	reference altitude used for skin friction drag calculations (feet)
h $(t - \tau)$	response of a system at time t, due to a unit control variable impulse at time τ (see convolution integral, equation 99)
I_{sp}	specific impulse, $T/\dot{W}$ (pounds thrust/pounds fuel per second)
$I_{\varphi\varphi}$, $I_{\psi\varphi}$, $I_{\psi\psi}$	integrals defined by equations 108b, c, and d
k	step size coefficient (see fig. 8)
K_m	fuel flow correction factor for rockets
K_R	constant used in sonic boom calculations
K_T	total impulse correction factor for rockets
L	lift force (pounds); reference length for sonic boom calculations
M	Mach number
m	number of control variables
N	force normal to body axis (pounds)
n	number of state variables plus enroute placards
p	number of terminal and enroute constraints
ΔP	sonic boom overpressure (pounds/feet^2)

P, P_∞	ambient pressure (pounds/feet^2)
P_{ref}	$\sqrt{P_{SL}\ P_\infty}$ (pounds/feet^2)
P_{SL}	sea-level ambient pressure (pounds/feet^2)
q	dynamic pressure, $1/2\ \rho\ V^2$ (pounds/feet^2)
$q\alpha$	dynamic pressure times angle of attack (pounds-degree/feet^2)
$\bar{r}$	position vector measured from earth's center to vehicle (feet)
r_D	desired final altitude for circular orbit (feet)
RPA	resultant physiological acceleration (feet/second^2)
R_o	radius of a spherical earth (feet)
S	aerodynamic reference area (square feet)
S_u^{ω}	integrated payoff function sensitivities (see equation 134)
s_u^{ω}	instantaneous payoff function sensitivities (see equation 130)
SFC	specific fuel consumption (pounds fuel per hour/pounds thrust)
T	actual thrust component along body axis (pounds); static temperature (degrees Rankine); final trajectory time (seconds)
t	independent trajectory variable, time (seconds)
t_o	initial trajectory time (seconds)
t_D	dummy time (seconds)
T_{ref}	uncorrected thrust as input for airbreathers (pounds)
T_T	total corrected thrust for rockets (pounds)
T_{total}	total temperature (degrees Rankine)
T_{vac}	uncorrected vacuum thrust as input for rockets (pounds)
TVL	velocity loss caused by thrust vectoring (feet/second)
$\bar{u}$, $\bar{u}(t)$	an m x 1 matrix of control variables
$\bar{V}$, $\bar{V}_I$	inertial velocity vector (feet/second)
V_{cs}	circular satellite velocity (feet/second)
$\bar{V}_R$	relative velocity vector (feet/second)
VF	function defined by equation 150
W	weight (pounds); an m x m weighting matrix
w_{ii}	elements of the weighting matrix W (equation 137)
W_f	fuel weight (pounds)

x	path range (feet)
x, y	coordinates of points on the forward shock signature (see fig. 7)
$\bar{x}, \bar{x}(t)$	an n x 1 matrix of state variables and enroute placards
y_c	lateral extent of forward shock signature (see fig. 7)
α	angle of attack (degrees)
α_o	angle of attack of the previous integration step (degrees)
β	latitude of vehicle (degrees)
β_G	latitude at which the overpressure intersects the ground (degrees) (see fig. 7)
β_{local}	latitude of a point on the forward shock signature (see fig. 7)
β_m	$\sqrt{M^2 - 1}$
γ	flight path angle (degrees); specific heat ratio
δ	Kronecker delta; indicates variance
δ_c	thrust cant angle (degrees)
η	throttling control variable
θ	pitch angle; control variable, $\gamma + \alpha$ (degrees)
Λ	sweepback control variable (degrees)
λ	longitude of vehicle (degrees)
$\lambda, \lambda(t)$	adjoint variables
λ_G	longitude at which the overpressure intersects the ground (see fig. 7)
λ_{local}	longitude of a point on the forward shock signature (see fig. 7)
$\lambda_\varphi, \lambda_\varphi(t)$	payoff function adjoint variables, measures sensitivity of φ at time T, to state variable changes at time t
$\lambda_\psi, \lambda_\psi(t)$	constraint function adjoint variables, measures sensitivity of a constraint at time T, to state variable changes at time t
$\lambda_\Omega, \lambda_\Omega(t)$	stopping function adjoint variables, measures sensitivity of the stopping function at time T, to state variable changes at time t
$\lambda_{\varphi\Omega}, \lambda_{\psi\Omega}$	see equations 98a and b
μ, μ_s	Lagrange multiplier (see equation 112); first Lagrange multiplier (see equation 113)

$\overline{\nu}$, $\overline{\nu}_s$	a 1 x p matrix of Lagrange multipliers (see equations 108 and 114)
ρ	ambient density (slugs/foot^3)
τ	dummy time used in Duhamel's and the convolution integral
ϕ	payoff function; bank angle (degrees)
ψ	heading angle, measured in degrees north of east
$\overline{\psi}$	a p x 1 constraint vector
Ω	stopping function
ω	angular velocity of earth (radians/second)

Subscripts

o	sea level or earth surface; initial; variable based on the angle of attack of the previous integration step and current state variables
R	refers to relative coordinate system
I	refers to inertial coordinate system
max	maximum
req	required
vac	vacuum
ref	reference
∞	ambient condition

Superscripts

$( )'$	transpose of a matrix ( )
$( )^{-1}$	inverse of a matrix ( )
$( )^*$	indicates nominal value

ANALYTICAL DEVELOPMENT

Equations of Motion

Coordinate system. — The flight-path coordinate system is used to define the vehicle position and attitude with respect to a spherical, rotating earth. This system is sometimes regarded as the "natural" coordinate system because the state variables of flight-path angle and velocity are explicit coordinates. The basic state variable coordinates are longitude (λ), latitude (β), altitude (h), relative velocity (V_R), relative flight-path angle (γ_R), and relative heading angle (ψ_R) (fig. 2).

Control variables. — The total force acting on the vehicle has three distinct sources: (1) gravitational force as a result of mutual mass attraction between the vehicle and earth; (2) aerodynamic force resulting from the vehicle motion through the atmosphere; and (3) thrust force from the vehicle propulsion system. There is no way of controlling the gravitational force since it is a function only of the state of the system.

The aerodynamic forces are determined by the geometry of the vehicle and its attitude with respect to the free-stream air mass. For fixed-geometry aircraft, the aerodynamic force is dependent on the angle of attack (defined as the angle between the vehicle longitudinal axis and the velocity vector). The angle of attack is used primarily to establish lift that is normally in a vertical plane. Out-of-plane maneuvers are made by banking the vehicle to direct the lift vector out of the vertical plane, thus permitting lateral translations. The bank angle, φ, is defined as the angle between the vertical plane containing the velocity vector and the vehicle plane of symmetry as viewed along the velocity vector (fig. 3).

A problem exists with angle of attack as a control variable for the point mass equations of motion when simulating the flight of a low thrust/weight, winged, air-breathing vehicle. The angle-of-attack control variable has been satisfactorily used for high thrust/weight vehicles, but the use of angle of attack for SST-type vehicles results in an unstable, oscillatory flight path during cruise which is not representative of the actual path. The use of a pitch angle (θ) given by

$$\theta = \gamma + \alpha \tag{1}$$

as a control variable resolves the problem and results in a stable, well behaved flight path in all cases checked. The analysis of this problem is given in appendix A. Note that if the vehicle is banked, θ is not the pitch angle in the usual sense (that is, the angle between the longitudinal axis and horizontal plane), but instead is the algebraic sum of the flight-path angle and the angle of attack as given by equation 1. Therefore, for nonzero bank, θ is not a physical angle.

Since some of the more advanced aircraft permit changing vehicle geometry by varying the wing sweep, the effect of this variable must be considered in

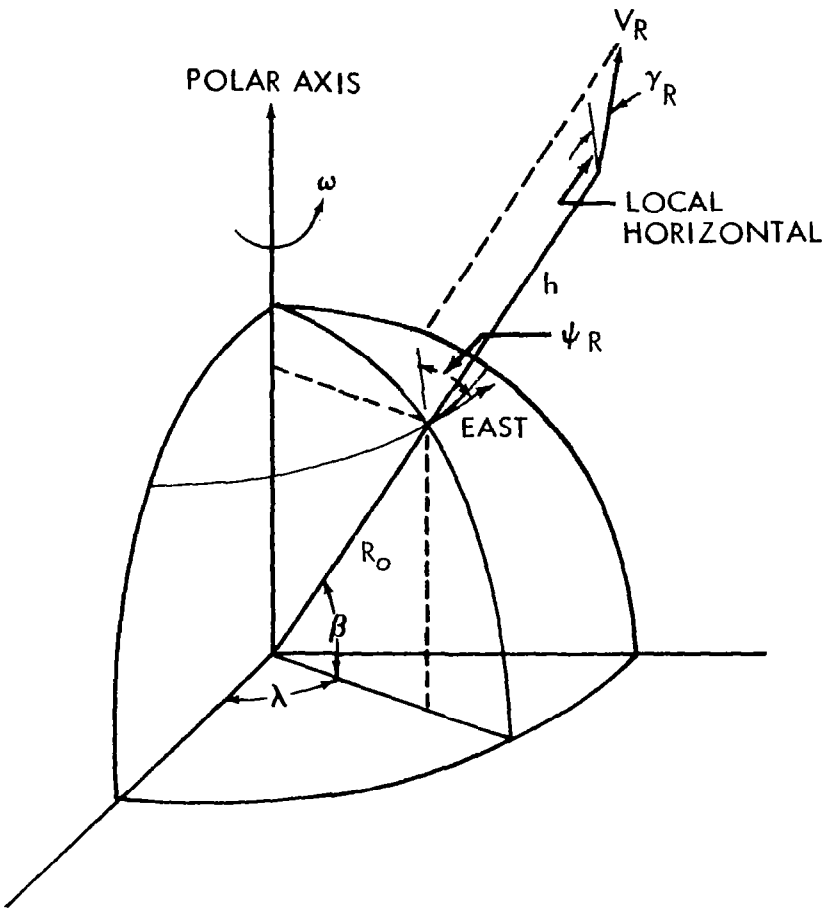


Figure 2. BASIC COORDINATE SYSTEM

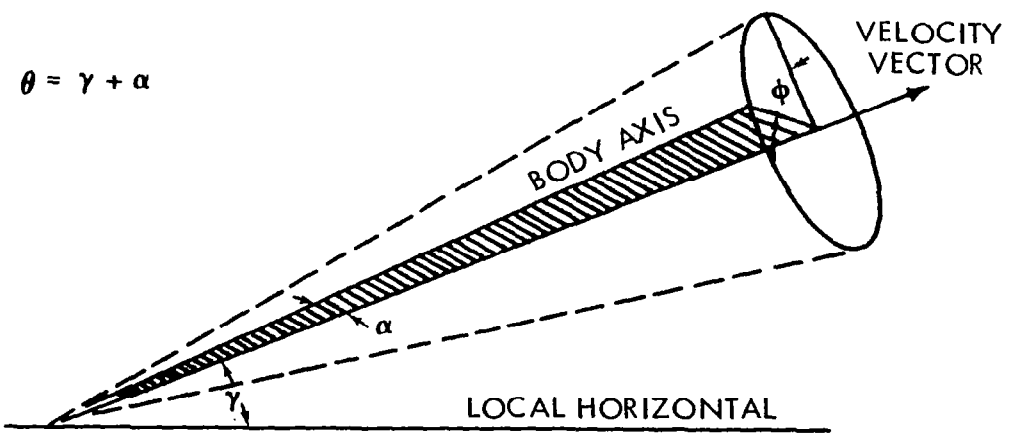


Figure 3. VEHICLE ORIENTATION

establishing the aerodynamic force on the vehicle. The wing sweep, Λ, is defined as the angle between the lateral body axis and a reference line on the wing (e.g., leading edge, quarter chord, and half chord).

The remaining control variable to be considered is the throttle setting, η, for controlling the thrust. This parameter is defined as the ratio of thrust at a given state to the reference thrust at that same state. Thrust is considered to act along the vehicle body axis.

Therefore, to determine the aerodynamic and thrust forces required for controlling the vehicle, four control variables are required: θ, φ, Λ, and η.

Kinematic equations. — The basic state equations of motion are defined in this section. Details of the development are not given here because the methods are standard.

The kinematics of a point mass vehicle for a rotating, central-force field is given by

$$\overline{V} = \overline{V}_R + \overline{\omega} \times \overline{r} \tag{2}$$

$$\overline{a} = \frac{d\overline{V}}{dt} + \overline{\omega} \times \overline{V} \tag{3}$$

where $\overline{V}$ is the inertial velocity vector

$\overline{V}_R$ is the relative velocity vector

$\overline{\omega}$ is the earth's rotation rate

$\overline{r}$ is the position vector of the vehicle

$\overline{a}$ is the inertial acceleration vector.

Substituting equation 2 into equation 3 gives

$$\overline{a} = \frac{d\overline{V}_R}{dt} + 2\,\overline{\omega} \times \overline{V}_R + \overline{\omega} \times (\overline{\omega} \times \overline{r}) \tag{4}$$

Using the coordinate system of figure 2, equation 4 can be written in components along the flight path, normal to the flight path in the plane containing the $\overline{r}$ and the $\overline{V}$ vector, and perpendicular to the plane containing $\overline{r}$ and $\overline{V}$.

$$
\begin{aligned}
a_t &= \dot{V}_R + \omega^2 \, r \cos\beta \, (\sin\beta \cos\gamma_R \sin\psi_R - \cos\beta \sin\gamma_R) \\
a_n &= V_R \dot{\gamma}_R - \frac{V_R^2}{r} \cos\gamma_R - 2\omega V_R \cos\beta \cos\psi_R \\
&\quad - \omega^2 \, r \cos\beta \, (\cos\beta \cos\gamma_R + \sin\beta \sin\gamma_R \sin\psi_R) \\
a_p &= V_R \cos\gamma_R \, \dot{\psi}_R + \frac{V_R^2}{r} \tan\beta \cos^2\gamma_R \cos\psi_R \\
&\quad - 2\omega V_R \, (\sin\gamma_R \sin\psi_R \cos\beta - \sin\beta \cos\gamma_R) \\
&\quad + \omega^2 \, r \sin\beta \cos\beta \cos\psi_R
\end{aligned}
\qquad (5)
$$

where γ_R is the relative flight-path angle

ψ_R is the relative heading measured north of east.

Other equations which follow from the procedures are

$$
\begin{aligned}
\dot{\beta} &= \frac{V_R \cos\gamma_R \sin\psi_R}{r} \\
\dot{\lambda} &= \frac{V_R \cos\gamma_R \cos\psi_R}{r \cos\beta} \\
\dot{h} &= V_R \sin\gamma_R \\
\dot{x} &= V_R \frac{R_o}{r} \cos\gamma_R
\end{aligned}
\qquad (6)
$$

where λ is the longitude angle

h is the altitude

x is the path range.

The change in weight of the vehicle is given by the differential equation

$$\dot{W} = - \text{ weight flow} \qquad (7)$$

where the weight flow is a combination of fuel flow and inerts.

The gravitational, aerodynamic, and thrust forces are required to complete the system. Other "auxiliary" state variables that are required will be defined in a later section.

Applied forces. — The forces acting on the vehicle which are not a result of the vehicle kinematics are discussed in this section.

Gravitational forces: The gravitational forces are determined for the central force field by representing the gravitational acceleration by the inverse square law

$$g = \frac{\mu}{r^2} = g_o \left(\frac{R_o}{r}\right)^2 \tag{8}$$

Aerodynamic forces: The aerodynamic forces for aircraft are written for the wind axes system with lift (L) normal to the flight path and drag (D) parallel to the flight path. Side-slip is assumed to be zero, i.e., all turns are assumed to be coordinated such that no side-slip results.

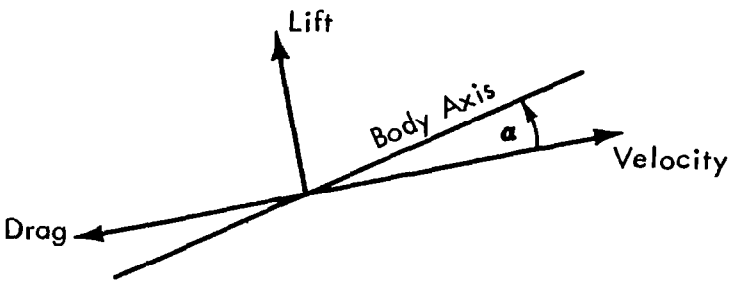


Figure 4. AERODYNAMIC FORCES - WIND AXES

If the bank angle is not zero, the lift vector is rotated about the velocity vector through the angle φ (bank angle).

The aerodynamic forces may be expressed as

$$L = C_L \, q \, S \tag{9}$$

$$D = C_D \, q \, S \tag{10}$$

where L is the lift force

D is the drag force

C_L is the lift coefficient

C_D is the drag coefficient

q is the dynamic pressure $= \frac{1}{2} \rho \, V^2$ or $\frac{\gamma \, P_\infty}{2} M^2$

S is the reference area.

Aerodynamic data input options are available for accepting data in the most common forms. For vehicles with fixed wings, the data are considered as

$$C_L = f \, (h, \, M, \, \alpha) \tag{11}$$

$$C_D = f\ (h,\ M,\ \alpha) \tag{12}$$

if not represented in polar form. If available as drag polars, the data are represented as

$$C_L = C_{L_\alpha}\ \alpha \tag{13}$$

$$C_D = C_{D_0} + \frac{\partial C_D}{\partial C_L^{\,2}}\ C_L^{\,2} \tag{14}$$

where C_{L_α} = lift curve slope, f (M) per degree

C_{D_0} = minimum drag coefficient, f (M)

$\frac{\partial C_D}{\partial C_L^{\,2}}$ = induced drag constant, f (M).

The aerodynamics for variable-geometry aircraft with wing sweep as a control variable are formulated as

$$C_L = f\ (\Lambda,\ M,\ \alpha) \tag{15}$$

$$C_D = f\ (\Lambda,\ M,\ \alpha) \tag{16}$$

or if the data are given as drag polars

$$C_L = C_{L_{\alpha=0}} + C_{L_\alpha}\ \alpha \tag{17}$$

$$C_D = C_{D_0} + \frac{\partial C_D}{\partial C_D^{\,2}}\ (C_L - C_{L_M})^2 \tag{18}$$

where $C_{L_{\alpha=0}}$ = lift coefficient for $\alpha = 0$, f (Λ, M)

C_{L_α} = lift curve slope, f (Λ, M) per degree

C_{D_0} = minimum drag coefficient, f (Λ, M)

$\frac{\partial C_D}{\partial C_L^{\,2}}$ = induced drag constant, f (Λ, M)

C_{L_M} = lift coefficient for minimum drag, f (Λ, M).

The effect of altitude on drag (skin friction, ΔC_{D_0}) is included as a linear function of altitude,

$$\Delta C_{D_0} = \frac{\Delta C_D}{\Delta h}\ (h - h_{ref}) \tag{19}$$

where $\Delta C_D/\Delta h$ is a function of the reference altitude.

The aerodynamic forces on a missile are quite often referenced to the body axis with the normal force perpendicular to the body axis and the axial force along the body axis.

$$N = C_N \, q \, S \tag{20}$$

$$A = C_A \, q \, S \tag{21}$$

where N is the normal force

A is the axial force

C_N is normal force coefficient

C_A is axial force coefficient.

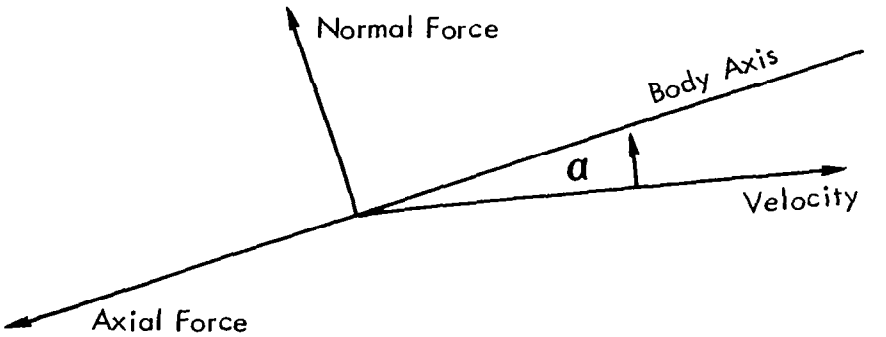


FIGURE 5. AERODYNAMIC FORCES - BODY AXES

The normal and axial force coefficients are given by

$$C_N = C_{N_\alpha} \, \alpha + C_{N_{\alpha^3}} \, \alpha^3 \tag{22}$$

$$C_A = C_{A_0} \text{ (not dependent on } \alpha) \tag{23}$$

where C_{N_α} = f (M) per degree

$C_{N_\alpha 3}$ = f (M) per degree^3

C_{A_0} = f (M).

The forces in the body axis system are transformed to the wind axis by

$$C_L = C_N \cos \alpha - C_A \sin \alpha \tag{24}$$

$$C_D = C_N \sin \alpha + C_A \cos \alpha \tag{25}$$

for use in the program.

Thrust forces: The thrust forces on a vehicle are considered to act in a direction fixed relative to the body axis so that the resultant thrust along the body axis is

$$T = T_T \cos \delta_c \tag{26}$$

where T is the thrust along body axis

T_T is the total thrust force

δ_c is the thrust cant angle.

(The angle δ_c is used only for symmetric nozzle configurations where the normal thrust component from two opposing engines cancel.)

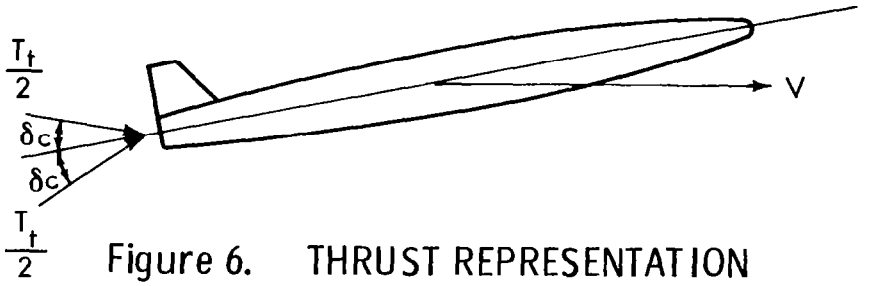


Figure 6. THRUST REPRESENTATION

The computation of thrust for airbreathing engines follows the form

$$T_T = T_{ref}\, \eta \tag{27}$$

where T_{ref} is the reference thrust = F (α, h, M, or V)

η is the throttling parameter.

The thrust may also be given by

$$T_{ref} = C_T\, q \tag{28}$$

where C_T is thrust coefficient = F (α, h, M, or V)

A thrust option that has particular use for the jet engine is

$$\frac{T_{ref}}{P} = f\ (\alpha,\ h,\ M) \tag{29}$$

which was formulated to give a smooth, consistent set of data over the entire operating range of the engine. P is the atmospheric pressure (psf).

The reference thrust for rocket engines is based on vacuum conditions. The total thrust is given by

$$T_T = K_T\, T_{vac}\, \eta - P_\infty\, A_e/144 \tag{30}$$

where T_{vac} is the vacuum thrust (pounds), f (time)

P_∞ is atmospheric ambient pressure (psf)

A_e is the exit area (square inches), f (time)

K_T is the total impulse correction factor.

Fuel flow: The fuel flow for airbreathing engines can be specified in terms of fuel flow directly or in terms of thrust specific fuel consumption.

$$\dot{W}_f = \frac{SFC}{3600} \times T_{ref} \tag{31}$$

The fuel flow and specific fuel consumption can be expressed as

$$\left.\begin{aligned} \dot{W}_f,\ SFC &= f\ (\eta,\ h,\ V) \\ &= f\ (\alpha,\ h,\ V) \\ &= f\ (\eta,\ M,\ h) \\ &= f\ (\alpha,\ M,\ h) \end{aligned}\right\} \tag{32}$$

depending on the independent variables required for adequate representation. The fuel flow for jet engines is also formulated as

$$\frac{\dot{W}_f}{P} = f\ (h,\ M,\ \eta) \tag{33}$$

and is designed to give a consistent set of data from idle speeds through maximum augmentation. Again P is the atmospheric pressure (psf).

For rocket engines, the fuel flow is obtained from the specific impulse by the relation

$$\dot{W}_f = K_m\ T_{vac}/I_{sp} \tag{34}$$

where K_m is fuel flow correction factor.

I_{sp} is specific impulse (seconds), f (time).

The options of thrust and fuel flow available will handle most types of engine data formulations.

Basic equations of motion. — The basic equations are summarized below with the effects of gravity, aerodynamics, and propulsion included.

$$\dot{V}_R = \frac{T \cos \alpha - D}{m} - g \sin \gamma_R + \omega^2 r \cos \beta\ (\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R) \tag{35}$$

$$\dot{\gamma}_R = \frac{(T \sin \alpha + L) \cos \varphi}{m V_R} - \left(\frac{g}{V_R} - \frac{V_R}{r}\right) \cos \gamma_R + 2\omega \cos \beta \cos \psi_R$$

$$+ \frac{\omega^2 r}{V_R} \cos \beta \, (\cos \beta \cos \gamma_R + \sin \beta \sin \gamma_R \sin \psi_R) \tag{36}$$

$$\dot{h} = V_R \sin \gamma_R \tag{37}$$

$$\dot{W} = - \dot{W}_f - \dot{W}_{inert} \tag{38}$$

$$\dot{\psi}_R = - \frac{(T \sin \alpha + L) \sin \varphi}{m V_R \cos \gamma_R} - \frac{V_R}{r} \tan \beta \cos \gamma_R \cos \psi_R$$

$$- 2\omega (\sin \beta - \tan \gamma_R \sin \psi_R \cos \beta)$$

$$- \frac{\omega^2 r \sin \beta \cos \beta \cos \psi_R}{V_R \cos \gamma_R} \tag{39}$$

$$\dot{\beta} = \frac{V_R \cos \gamma_R \sin \psi_R}{r} \tag{40}$$

$$\dot{\lambda} = \frac{V_R \cos \gamma_R \cos \psi_R}{r \cos \beta} \tag{41}$$

These equations are sufficient to define a vehicle trajectory for a spherical, rotating earth. The equation of motion for path range on the earth's surface is

$$\dot{x} = \frac{R_o}{r} V_R \cos \gamma_R \tag{42}$$

Auxiliary state variables. —Variables required for optimization, stopping or constraining a flight path, but which do not fall in the category of basic state variables, are defined as auxiliary state variables. The differential equations for these variables are integrated when desired by the user. The additional variables and their defining equations are given below.

Dummy time: The dummy time, introduced so that it may be used as a state variable, is defined by

$$\frac{dt_D}{dt} = 1 \ . \tag{43}$$

Aerodynamic heating integral: The aerodynamic heating integral, a measure of the heat encountered by a vehicle during flight, is defined by

$$\frac{d\,AHI}{dt} = \frac{1}{2}\rho V_R^3 \tag{44}$$

where ρ is the atmospheric density.

Ideal relative ΔV: The ideal relative ΔV, the velocity change a vehicle would experience in the absence of atmospheric and gravitational effects, is defined by

$$\frac{d\,\Delta V}{dt} = \frac{T}{m} + \omega^2 r \cos\beta\,(\cos\beta \sin\gamma_R - \sin\beta\cos\gamma_R \sin\psi_R) \tag{45}$$

where m is the vehicle mass.

Drag loss: Drag loss, the velocity decrement resulting from motion through the atmosphere, is defined by

$$\frac{d\,DL}{dt} = \frac{D}{m} \tag{46}$$

Gravity loss: Gravity loss, the velocity decrement resulting from motion in the earth's gravitational field, is defined by

$$\frac{d\,GL}{dt} = g \sin\gamma_R \tag{47}$$

Thrust vector loss: Thrust vector loss, the velocity decrement resulting from the thrust vector not aligned with the flight path velocity vector, is defined by

$$\frac{d\,TVL}{dt} = \frac{T}{m}(1 - \cos\alpha) \tag{48}$$

Inequality constraints. —During the course of a flight path, a function may be required to be less than or equal to and/or greater than or equal to a particular value or set of values. Inequality constraints arise from a number of different considerations: (1) geopolitical limitations; (2) passenger comfort; (3) control limits; (4) structural limits; and (5) engine limits.

Geopolitical: This class of inequality constraints is related to consideration of other than aircraft limitations. The maximum and minimum altitude may be imposed by the FAA for traffic control. Aircraft noise and sonic boom (ref. 7) are important in vicinity of cities, airports, etc. The sonic boom is of special consideration to SST flight paths because the boom signature and overpressure (ΔP) on the ground may design the flight path. The sonic boom overpressure on the ground, in the plane of the velocity vector, may be found from the following expression:

$$\frac{\Delta P\,(h/L)^{3/4}}{K_R\,\beta_m^{1/4}\,P_{ref}} = f(M,\ h,\ \beta_m\ C_L S/2L^2) \tag{49}$$

where

$$\beta_m = \sqrt{M^2 - 1}$$

$$P_{ref} = \sqrt{P_{SL}\, P_\infty}$$

$$K_R = 1.9$$

and f (M, h, β_m C_L S/2L^2) is a tabular function. The overpressure generated by an airplane at a latitude and longitude (β, λ) will intersect the ground ahead of the airplane in the plane of the velocity vector at a latitude and longitude of

$$\beta_G = \beta + \frac{h \sin \psi_R}{\beta_m R_o} \tag{50}$$

$$\lambda_G = \lambda + \frac{h \cos \psi_R}{\beta_m R_o \cos \beta} \tag{51}$$

The shock pattern on the ground is considered to be the intersection of a cone (with the half apex angle equal to the complement of the Mach angle) with the ground plane (fig. 7). This intersection, or shock signature, is a hyperbola with the equation

$$h^2 + y^2 = x^2 \beta_m^2 \tag{52}$$

where h is the vehicle altitude

y is lateral distance from the flight path

x is distance along the flight path from the vehicle to the point on the shock signature.

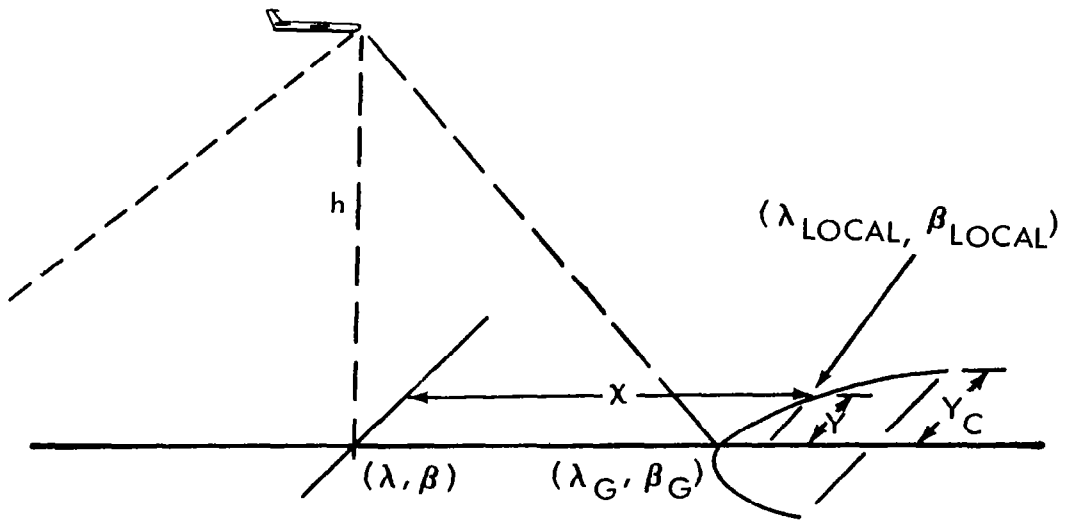


Figure 7: SONIC BOOM GEOMETRY

The lateral extent of the shock wave is given as a tabular function

$$y_c = f\ (h, M) \tag{53}$$

The geographical location of a point along the shock signature is expressed in terms of longitude, latitude, heading angle, and altitude

$$\beta_{local} = \beta + \frac{1}{R_o}\left[\frac{\sin \psi_R}{\beta_m}\sqrt{h^2 + y^2} + y \cos \psi_R\right] \tag{54}$$

$$\lambda_{local} = \lambda + \frac{1}{R_o \cos \beta}\left[\frac{\cos \psi_R}{\beta_m}\sqrt{h^2 + y^2} - y \sin \psi_R\right] \tag{55}$$

where y can go from $- y_c$ to $+ y_c$ to cover the entire shock signature.

The overpressure along the shock signature is related to that given in equation 49 ($\Delta P_{y=0}$) by

$$\Delta P(y) = \Delta P_{y=0}\left[\frac{h}{\sqrt{h^2 + y^2}}\right]^{3/4} \tag{56}$$

The maximum allowable overpressure over the ground is given by

$$\Delta P_{max} = \Delta P_{max}\ (\lambda_{local}, \beta_{local}) \tag{57}$$

where λ_{local} and β_{local} are the points along the shock signature.

The next step is to derive a function in terms of the instantaneous vehicle state that represents the integral of the overpressure violation along the shock signature from $- y_c$ to $+ y_c$. The overpressure violation is given by

$$\Delta P_s - \Delta P_{max} > 0 \tag{58}$$

where the ΔP_s is given by equation 56. A true integral of the violation along the signature is not practical, therefore an approximation is made by summing the overpressure violations for several selected points as follows.

$$\frac{d\ BOOM\ PF}{dt} = \delta\left[\sum_{i=1}^{NP} (\Delta P_s - \Delta P_{max})_i\right]^2 \tag{59}$$

$\delta = 0$ if $\Delta P_s < \Delta P_{max}$

$\delta = 1$ if $\Delta P_s \geq \Delta P_{max}$

NP is the number of points selected for the approximation. At present 7 points are used spaced at an interval of $y_c/3$. Equation 59 which defines the sonic boom penalty function, is integrated as one of the equations of motion along the

path with a terminal constraint value of zero. The partial derivatives of equation 59 with respect to the state and control variables are determined numerically (appendix B).

Passenger comfort: The comfort of a passenger is significant to the airlines who are trying to please the customer. To insure that the vehicle does not fly into areas of discomfort, placards may be imposed on normal load factor (n), pitch angle (θ), bank angle (φ), resultant physiological acceleration (RPA), and altitude rate ($\dot{h}$). The altitude rate controls such items as rate of change of cabin pressure. The magnitude of cabin pressure is determined by altitude and mechanical or compressor limitations.

Control: These placards result from physical or mechanical limitations on the system. The control variables limited are pitch angle, bank angle, throttle setting, and wing sweep.

Structural: The structural requirement on airframe requires that the state of the system be constrained so that dynamic pressure, dynamic pressure times angle of attack ($Q\alpha$), stagnation temperature, and cabin pressure differential do not exceed prescribed limits. The stagnation temperature is given by the adiabatic equation

$$T_{total} = T\left(1 + \frac{\gamma - 1}{2} M^2\right) \tag{60}$$

where γ = specific heat ratio and T is the ambient temperature. Cabin pressure differential can be formulated as a placard on altitude rate.

Engine: The engine is protected from flight conditions that would produce undesirable results by bounding the altitude Mach number region in which the vehicle can operate. In addition, maximum and minimum throttle setting may be bounded by a control placard as a function of altitude and Mach number.

Several methods of handling inequality constraints have been devised, i.e., integral method (ref. 3), penalty function method (ref. 4), and a method by Bryson, Denham, and Dreyfus (refs. 5 and 6). The method used during this study is based on the square of the violation. This method is similar to the integral method but produces a form more satisfactory for use with analytical partial derivatives. Consider a typical problem as shown in figure 8.

Let C be the time history of the function to be constrained, which violates the placard shown as P. The region of constraint violation is shaded. A measure of the total constraint violation is the areas shaded, or

$$A = \int_{to}^{T} (C - P)\ \delta\ dt \tag{61}$$

where $\delta = 0$ when there is no violation

$\delta = 1$ when there is a violation.

This formulation is not satisfactory for analytical partials since the partial derivative of $\dot{A}/dt$ with respect to the state or control variables does not have a measure of the violation. A form which does have the required nature is

$$A = \int_{to}^{T} (C - P)^2 \, \delta \, dt \tag{62}$$

where the partial derivative with respect to the state variables does have the proper form

$$\frac{\partial \dot{A}}{\partial X} = 2 \, (C - P)\left(\frac{\partial C}{\partial X} - \frac{\partial P}{\partial X}\right) \tag{63}$$

which is still dependent on the violation.

Each of the inequality constraints is formulated in the same manner and permits a minimum as well as a maximum limit. Each is treated in the same manner. Inequality constraints can be imposed for the placards shown in figure 1.

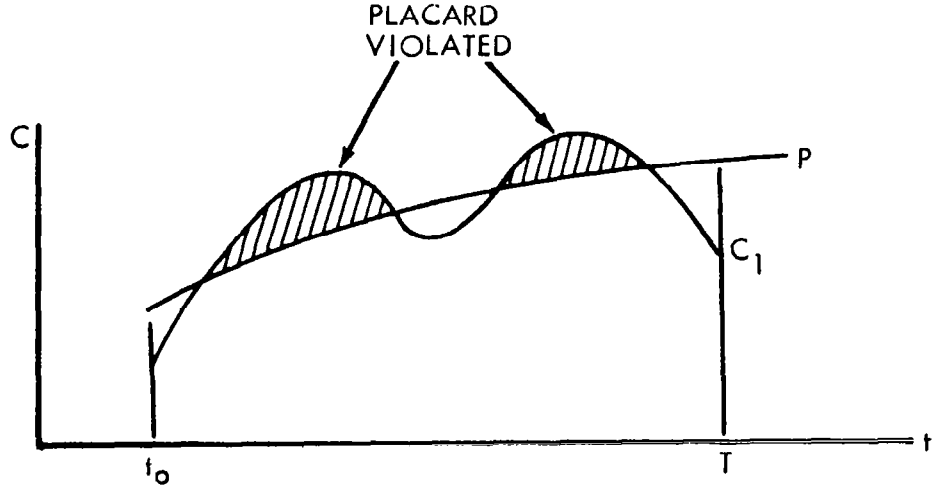


Figure 8: INEQUALITY CONSTRAINT

Steepest-Ascent Technique

The analysis of trajectories by the steepest-ascent method has been treated extentensively in the literature (refs. 1, 3, and 4). The derivation given here follows that of Bryson and Denham, reference 1. Since the steepest-ascent method is basic to this study, the derivation is included here for completeness.

Statement of problem. — The steepest-ascent solution is formulated to determine the control variable history, u(t), so as to maximize a function (payoff function)

$$\varphi = \varphi\left[\bar{x}(T),\ T\right] \tag{64}$$

and meet the terminal constraints

$$\bar{\psi} = \bar{\psi}\left[\bar{x}(T),\ T\right] = 0 \tag{65}$$

where φ = performance index to be optimized

$\bar{x}$ = state variable matrix (order n x 1)

T = time at the stopping condition

$\bar{\psi}$ = a matrix of terminal constraint functions (order p x 1).

The stopping time, T, is established by the condition

$$\Omega = \Omega\left[\bar{x}(T),\ T\right] = 0 \tag{66}$$

where Ω is referred to as the stopping condition.

The equations of motion defining the state of the system can be written in terms of first-order differential equations. The form of these equations is

$$\dot{\bar{x}}(t) = \bar{f}\left[\bar{x}(t),\ \bar{u}(t),\ t\right] \tag{67}$$

where $\bar{u}(t)$ is an (m x 1) matrix of control variables

$\bar{f}$ is an (n x 1) matrix of functions defining the time derivatives of the state variables.

The initial values of the state variables, $\bar{x}(t_o)$, are generally known for a given problem. If some of the initial values are not specified, they may be determined along with the control matrix to optimize φ. If it is desired to optimize some function other than one of the normal state variables, it is necessary to introduce an additional state variable and another differential equation defining the state variable and solve it simultaneously with the required equations of motion.

The steepest-ascent process is started by assuming a control variable time history, $\bar{u}(t)$. About the only requirement on $\bar{u}(t)$ is that the trajectory generated must reach the stopping condition.

The nominal trajectory is generated using the assumed $\bar{u}(t)$, vehicle and engine characteristics, and the initial conditions. The equations of motion (equation 67) are integrated numerically to obtain the nominal state variable program until the stopping condition is satisfied. For convenience, the nominal conditions will be designated by ( )*, i.e.,

$$\bar{u}(t)_{\text{nominal}} = \bar{u}^*(t) \tag{68}$$

$$\bar{x}(t)_{\text{nominal}} = \bar{x}^*(t) \tag{69}$$

The steepest-ascent procedure uses the concept of local linearization about a nominal flight path. For small perturbation in the control variables

$$\bar{u}(t) = \bar{u}^*(t) + \delta\bar{u}(t) \tag{70}$$

the state variables are perturbed

$$\bar{x}(t) = \bar{x}^*(t) + \delta\bar{x}(t) \tag{71}$$

Substituting equations 70 and 71 into equation 67, the linear differential equation for $\delta\bar{x}$ is obtained

$$\frac{d}{dt}\,\delta\bar{x}(t) = F\,\delta\bar{x} + G\,\delta\bar{u} \tag{72}$$

where

$$F = \begin{bmatrix} \left(\frac{\partial f_1}{\partial x_1}\right)^* & \cdots & \left(\frac{\partial f_1}{\partial x_n}\right)^* \\ \vdots & & \vdots \\ \left(\frac{\partial f_n}{\partial x_1}\right)^* & \cdots & \left(\frac{\partial f_n}{\partial x_n}\right)^* \end{bmatrix} \quad \text{(order n x n)} \tag{73}$$

$$G = \begin{bmatrix} \left(\frac{\partial f_1}{\partial u_1}\right)^* & \cdots & \left(\frac{\partial f_1}{\partial u_m}\right)^* \\ \vdots & & \vdots \\ \left(\frac{\partial f_n}{\partial u_1}\right)^* & \cdots & \left(\frac{\partial f_n}{\partial u_m}\right)^* \end{bmatrix} \quad \text{(order n x m)} \tag{74}$$

Again the ( )* indicates that the partial derivatives are evaluated along the nominal flight path.

Adjoint equations. — The effects of the control variable perturbations on the payoff quantity, stopping condition, and constraint functions must now be determined.

Equation 72 is a linear equation that describes the small perturbations about the nominal path. To this is added a linear differential equation adjoint to equation 72

$$\frac{d\lambda}{dt} = - F' \lambda \tag{75}$$

where the λ's are a matrix of adjoint variables. The significance of the λ's and their role as influence functions for φ, $\bar{\psi}$, and Ω will be demonstrated. To show this, premultiply equation 72 by λ', equation 75 by $\delta\bar{x}'$ (where $(\ )'$ indicates the transpose of the matrix), and add the transpose of the second product to the first

$$\lambda' \frac{d\delta\bar{x}}{dt} + \frac{d\lambda'}{dt} \delta\bar{x} = \lambda' F \delta\bar{x} + \lambda' G \delta\bar{u} - \lambda' F \delta\bar{x} \tag{76}$$

which can be written as

$$\frac{d}{dt} (\lambda' \delta\bar{x}) = \lambda' G \delta\bar{u} \tag{77}$$

Integrating equation 77,

$$(\lambda' \delta\bar{x})_{t=T} = \int_{t_o}^{T} \lambda' G \delta\bar{u} \, dt + (\lambda' \delta\bar{x})_{t=t_o} \tag{78}$$

Examining this equation gives much information on the nature of the adjoint variables. The product $G \delta\bar{u}$ gives the rate of change of the state variables due to changes of the control variables only:

$$\delta\dot{\bar{x}} = G \delta\bar{u} \tag{79}$$

therefore, equation 78 can be written

$$(\lambda' \delta\bar{x})_{t=T} = \int_{t_o}^{T} \lambda' \delta\dot{\bar{x}} \, dt + (\lambda' \delta\bar{x})_{t=t_o} \tag{80}$$

but this is exactly the form of the Duhamel integral, which is

$$z(t) = \int_{t_o}^{t} A (t - \tau) \frac{df(\tau)}{d\tau} \, d\tau + A (t - t_o) f(t_o) \tag{81}$$

where $A (t - \tau)$ is the indicial response at time t due to a unit step in $f(t)$ at time τ. The $f(t)$ is the arbitrary forcing function; $z(t)$ is the response of the system to the arbitrary forcing function. Making a comparison between the two equations for $t = T$, the λ's are the indicial response of the system due to a unit step in each element of the state variable vector, $\delta\bar{x}$. The variables of interest for this work are the payoff function φ, the stopping condition Ω, and the terminal constraints $\bar{\psi}$. A set of λ's can now be defined corresponding to φ, Ω, and the $\bar{\psi}$'s. The adjoint variables are identified as

$$\lambda'_{\varphi} = \frac{\partial \varphi}{\partial \bar{x}}, \qquad \lambda'_{\psi} = \frac{\partial \bar{\psi}}{\partial \bar{x}}, \qquad \lambda'_{\Omega} = \frac{\partial \Omega}{\partial \bar{x}} \tag{82}$$

where λ'_{φ} is a (1 x n) matrix, λ'_{ψ} is a (p x n) matrix, and λ'_{Ω} is a (1 x n) matrix. The λ's are influence functions and may be considered as

$$\lambda_{\varphi_i} = \frac{\partial \varphi}{\partial x_i} \tag{83}$$

or the change of φ due to a unit step change of x_i. Similar interpretation can be made for λ_{ψ} and λ_{Ω}. A set of boundary conditions can now be written for the λ's at t = T giving

$$\lambda_{\varphi_j}(T) = \left(\frac{\partial \varphi}{\partial x_j}\right)_T \qquad j = 1, \ldots, n \tag{84}$$

$$\lambda^i_{\psi_j}(T) = \left(\frac{\partial \psi_i}{\partial x_j}\right)_T \qquad i = 1, \ldots, p \qquad j = 1, \ldots, n \tag{85}$$

$$\lambda_{\Omega_j}(T) = \left(\frac{\partial \Omega}{\partial x_j}\right)_T \qquad j = 1, \ldots, n \tag{86}$$

If the payoff, constraint, and stopping functions are restricted to state variables, the $\lambda(T)$'s are either zeroes or ones.

Starting from the terminal values of the λ's, equation 75 may be integrated backwards along the flight path (i.e., from T to t_o) using partial derivatives which were stored during the forward trajectory to obtain λ'_{φ}, λ'_{ψ}, and λ'_{Ω} as functions of time.

Substituting the functions from equations 84, 85, and 86 into equation 78 and noting that

$$(\lambda'_{\varphi}\ \delta\bar{x})_T = \delta\varphi(T) \tag{87}$$

$$(\lambda'_{\psi}\ \delta\bar{x})_T = \delta\bar{\psi}(T) \tag{88}$$

$$(\lambda'_{\Omega}\ \delta\bar{x})_T = \delta\Omega(T) \tag{89}$$

the following are obtained

$$\delta\varphi(T) = \int_{t_o}^{T} \lambda'_{\varphi}\ G\ \delta\bar{u}\ dt + (\lambda'_{\varphi}\ \delta\bar{x})_{t=t_o} \tag{90}$$

$$\delta\bar{\psi}(T) = \int_{t_o}^{T} \lambda'_{\psi}\ G\ \delta\bar{u}\ dt + (\lambda'_{\psi}\ \delta\bar{x})_{t=t_o} \tag{91}$$

$$\delta\Omega(T) = \int_{t_o}^{T} \lambda'_{\Omega}\, G\, \delta\bar{u}\, dt + (\lambda'_{\Omega}\, \delta\bar{x})_{t=t_o} \tag{92}$$

These equations would give the changes in the payoff function, stopping conditions, and terminal constraints if the final time (T) at the stopping condition did not change. In general, this is not true and, due to small perturbations, the stopping time is at some time, $T + \Delta T$. To account for this change equations 90, 91, and 92 become

$$d\varphi = \int_{t_o}^{T} \lambda'_{\varphi}\, G\, \delta\bar{u}\, dt + (\lambda'_{\varphi}\, \delta\bar{x})_{t=t_o} + \dot{\varphi}(T)\, \Delta T \tag{93}$$

$$d\psi = \int_{t_o}^{T} \lambda'_{\psi}\, G\, \delta\bar{u}\, dt + (\lambda'_{\psi}\, \delta\bar{x})_{t=t_o} + \dot{\bar{\psi}}(T)\, \Delta T \tag{94}$$

$$d\Omega = \int_{t_o}^{T} \lambda'_{\Omega}\, G\, \delta\bar{u}\, dt + (\lambda'_{\Omega}\, \delta\bar{x})_{t=t_o} + \dot{\Omega}(T)\, \Delta T \tag{95}$$

Note that these equations are now total differentials. This set of equations gives the changes in φ, $\bar{\psi}$, and Ω for the perturbed trajectory.

The stopping condition does not change from one iteration to the next, therefore $d\Omega(T) = 0$, giving the relation

$$\Delta T = -\frac{1}{\dot{\Omega}(T)} \int_{t_o}^{T} \lambda'_{\Omega}\, G\, \delta\bar{u}\, dt - \frac{1}{\dot{\Omega}(T)} (\lambda'_{\Omega}\, \delta\bar{x})_{t=t_o} \tag{96}$$

the equation for φ and $\bar{\psi}$ from equations 93, 94, and 95 can now be written as

$$d\varphi = \int_{t_o}^{T} \lambda'_{\varphi\Omega}\, G\, \delta\bar{u}\, dt + (\lambda'_{\varphi\Omega}\, \delta\bar{x})_{t=t_o} \tag{97}$$

$$d\bar{\psi} = \int_{t_o}^{T} \lambda'_{\psi\Omega}\, G\, \delta\bar{u}\, dt + (\lambda'_{\psi\Omega}\, \delta\bar{x})_{t=t_o} \tag{98}$$

where

$$\lambda'_{\varphi\Omega} = \lambda'_{\varphi} - \frac{\dot{\varphi}(T)}{\dot{\Omega}(T)}\, \lambda'_{\Omega} \tag{98a}$$

$$\lambda'_{\psi\Omega} = \lambda'_{\psi} - \frac{\dot{\bar{\psi}}(T)}{\dot{\Omega}(T)}\, \lambda'_{\Omega} \tag{98b}$$

The $\lambda_{\varphi\Omega}$ and $\lambda_{\psi\Omega}$ are modified influence functions, which account for a change in the time at the stopping condition.

The integral terms of equations 97 and 98 may be compared with the convolution integral used for determining the response of a system to a continuous control variable forcing function by first obtaining the impulse response functions, i.e.,

$$z(t) = \int_{t_o}^{t} h\ (t - \tau)\ g(\tau)\ d\tau \tag{99}$$

where $h\ (t - \tau)$ is the response of a system at time t due to a unit control variable impulse at time τ. The function $g(\tau)$ is the continuous control variable forcing function and $z(t)$ is the response at some arbitrary time t to that forcing function.

If the arbitrary time is taken to be the final time T, and the forcing functions are the perturbed control variables $\delta\bar{u}$, the functions $\lambda'_{\varphi\Omega}\ G$ and $\lambda'_{\psi\Omega}\ G$ are shown to be the responses of the final values of the payoff function and terminal constraints to a unit impulse of each of the elements of $\delta\bar{u}$ at some time t.

Variational equations. —For steepest ascent, the control variable history, $\delta\bar{u}$, that maximizes $d\varphi$ given by equation 97 for a given $d\bar{\psi}$ and $d\Omega = 0$ is desired. An additional requirement is that dU defined from

$$(dU)^2 = \int_{t_o}^{T} \delta\bar{u}'(t)\ W\ \delta\bar{u}(t)\ dt \tag{100}$$

be chosen to ensure that the perturbations, $\delta\bar{u}$, will be small enough for the linearizations leading to equations 84, 85, and 86 to be within reason. W is an (m x m) symmetric weighting matrix chosen to improve convergence during the steepest-ascent procedure. The automation of logic for the selection of $(dU)^2$ and the weighting matrix are discussed in later sections.

The values of $d\bar{\psi}$ are chosen to bring the nominal solution closer to the desired terminal constraints, $\bar{\psi} = 0$. It is desirable, in many cases, to restrict the change of constraints, $d\bar{\psi}$, allowed for each iteration.

The procedure now follows the calculus of variations using the method of Lagrange multipliers. A linear combination of equations 98 and 100 with equation 97 gives

$$d\varphi = \int_{t_o}^{T} \lambda'_{\varphi\Omega}\, G\, \delta\bar{u}\, dt + \lambda'_{\varphi\Omega}\,(t_o)\, \delta\bar{x}\,(t_o)$$
$$+ \nu' \left[d\bar{\psi} - \int_{t_o}^{T} \lambda'_{\psi\Omega}\, G\, \delta\bar{u}\, dt - \lambda'_{\psi\Omega}\,(t_o)\, \delta\bar{x}\,(t_o) \right]$$
$$+ \mu \left[(dU)^2 - \int_{t_o}^{T} \delta\bar{u}'\, W\, \delta\bar{u}\, dt \right] \qquad (101)$$

or combined gives

$$d\varphi = \int_{t_o}^{T} \left[\lambda'_{\varphi\Omega}\, G - \nu'\, \lambda'_{\psi\Omega}\, G - \mu\, \delta\bar{u}'\, W \right] \delta\bar{u}\, dt$$
$$+ \left[\lambda'_{\varphi\Omega}\,(t_o) - \nu'\, \lambda'_{\psi\Omega}\,(t_o) \right] \delta\bar{x}\,(t_o)$$
$$+ \nu'\, d\bar{\psi} + \mu\, (dU)^2 \qquad (102)$$

where ν is a (1 x p) matrix of constant Lagrange multipliers, and μ is a constant. Both ν and μ can be chosen for convenience, since they were assumed arbitrary.

Taking the variation of equation 102 with respect to the control variable gives

$$\delta(d\varphi) = \int_{t_o}^{T} \left[(\lambda'_{\varphi\Omega}\, G - \nu'\, \lambda'_{\psi\Omega}\, G - \mu\, \delta\bar{u}'\, W)\, \delta^2\bar{u} - \mu\, \delta^2\bar{u}'\, W\, \delta\bar{u} \right] dt \qquad (103)$$

where $\delta\bar{x}\,(t_o)$, $d\bar{\psi}$, and dU are considered to be constants. Since this is a scalar equation

$$\delta^2\bar{u}'\, W\, \delta\bar{u} \equiv \delta\bar{u}'\, W\, \delta^2 u \qquad (104)$$

i.e., the transpose is equal to the matrix where W is a symmetric matrix. Therefore the expression for the variance of $d\varphi$ is reduced to

$$\delta(d\varphi) = \int_{t_o}^{T} \left[\lambda'_{\varphi\Omega}\, G - \nu'\, \lambda'_{\psi\Omega}\, G - 2\mu\, \delta\bar{u}'\, W \right] \delta^2\bar{u}\, dt \qquad (105)$$

The optimal $d\varphi$ will occur where its variance is zero. Therefore, since $\delta^2\bar{u} \neq 0$,

$$\lambda'_{\varphi\Omega} G - \nu' \lambda'_{\psi\Omega} G - 2\mu\, \delta\bar{u}'\, w = 0 \qquad (106)$$

Solving for $\delta\bar{u}$ gives

$$\delta\bar{u} = \frac{1}{2\mu} W^{-1} G' (\lambda_{\varphi\Omega} - \lambda_{\psi\Omega}\nu) \qquad (107)$$

where the $(\ )^{-1}$ indicates the inverse of a matrix. Substituting the expression back into equation 98 and solving for ν gives

$$\nu = I^{-1}_{\psi\psi} I_{\psi\varphi} - 2\mu I^{-1}_{\psi\psi} d\bar{\beta} \tag{108}$$

where

$$d\bar{\beta} = d\bar{\psi} - \lambda'_{\psi\Omega}(t_o)\ \delta\bar{x}(t_o) \tag{108a}$$

$$I_{\psi\psi} = \int_{t_o}^{T} \lambda'_{\psi\Omega} G W^{-1} G' \lambda_{\psi\Omega}\, dt \tag{108b}$$

$$I_{\psi\varphi} = \int_{t_o}^{T} \lambda'_{\psi\Omega} G W^{-1} G' \lambda_{\varphi\Omega}\, dt \tag{108c}$$

$$I_{\varphi\varphi} = \int_{t_o}^{T} \lambda'_{\varphi\Omega} G W^{-1} G' \lambda_{\varphi\Omega}\, dt \tag{108d}$$

In the computer program, reference is made to the "I" matrix, which is defined as

$$I = \int_{t_o}^{T} \Lambda' G W^{-1} G' \Lambda\, dt \tag{109}$$

where Λ is the n x (p + 1) matrix, which may be represented as a partitioned matrix as

$$\Lambda = \underset{n \times 1}{\left[\lambda_{\varphi\Omega}\right.} \vdots \underset{n \times p}{\left.\lambda_{\psi\Omega}\right]} \tag{110}$$

The I matrix can be partitioned as follows

$$I = \left[\begin{array}{c|c} I_{\varphi\varphi} & I_{\varphi\psi} \\ \hline I_{\psi\varphi} & I_{\psi\psi} \end{array}\right] \tag{111}$$

This is a symmetric matrix since $I_{\varphi\psi} = I'_{\psi\varphi}$ and $I_{\psi\psi}$ is symmetric, where $I_{\varphi\varphi}$, $I_{\psi\varphi}$, and $I_{\psi\psi}$ are defined by equation 108b, c, d.

Using the expression for ν (equation 108) and the equation for $\delta\bar{u}$ (equation 107), substituting into equation 100 and solving for μ gives

$$\mu = \pm \frac{1}{2} \left[\frac{I_{\varphi\varphi} - I'_{\psi\varphi} I^{-1}_{\psi\psi} I_{\psi\varphi}}{(dU)^2 - d\bar{\beta}' I^{-1}_{\psi\psi} d\bar{\beta}}\right]^{1/2} \tag{112}$$

The Lagrange multipliers referred to in STOP are of different form than those given by equations 108 and 112. For convenience, the "first Lagrange multiplier" in the program, μ_s, is

$$\mu_s = \frac{1}{2\mu} = \pm\left[\frac{(du)^2 - d\bar{\beta}' I_{\varphi\varphi}^{-1} d\bar{\beta}}{I_{\varphi\varphi} - I'_{\psi\varphi} I_{\bar{\psi}\bar{\psi}}^{-1} I_{\psi\varphi}}\right]^{1/2} \tag{113}$$

and the array of Lagrange multipliers are denoted by

$$\nu_s = -\frac{\nu}{2\mu} = -\mu_s \nu \tag{114}$$

The form of the perturbed control variable (equation 107) therefore can be written as

$$\delta\bar{u} = W^{-1} G' \Lambda \begin{bmatrix} \mu_s \\ \nu_s \end{bmatrix} \tag{115}$$

The complete expression for $\delta\bar{u}$ can now be obtained by substituting the equations for μ and ν into equation 107, which gives

$$\delta\bar{u}(t) = \pm W^{-1} G' (\lambda_{\varphi\Omega} - \lambda_{\psi\Omega} I_{\psi\psi}^{-1} I_{\psi\varphi}) \left[\frac{(dU)^2 - d\bar{\beta}' I_{\psi\psi}^{-1} d\bar{\beta}}{I_{\varphi\varphi} - I'_{\psi\varphi} I_{\psi\psi}^{-1} I_{\psi\varphi}}\right]^{1/2} + W^{-1} G' \lambda_{\psi\Omega} I_{\psi\psi}^{-1} d\bar{\beta} \tag{116}$$

This equation is the foundation for changing the control variable to improve the performance for each iteration. The payoff function change predicted for an iteration is obtained by substituting equation 116 for $\delta\bar{u}(t)$ into equation 97 for $d\varphi$, giving

$$d\varphi = \pm\left[\left((dU)^2 - d\bar{\beta}' I_{\psi\psi}^{-1} d\bar{\beta}\right)\left(I_{\varphi\varphi} - I'_{\psi\varphi} I_{\psi\psi}^{-1} I_{\psi\varphi}\right)\right]^{1/2} + I'_{\psi\varphi} I_{\psi\psi}^{-1} d\bar{\beta} + \lambda'_{\varphi\Omega}(t_o)\, \delta\bar{x}(t_o) \tag{117}$$

where the + sign is used to increase φ and the - sign to decrease φ. This procedure will give the optimum trajectory as discussed under the statement of the problem. The method for perturbing the free initial condition $\delta\bar{x}(t_o)$ will be discussed in a later section. The effects of varying the initial conditions enter the problem through its effect on $d\bar{\beta}$ as well as $\delta\bar{x}(t_o)$.

Iterative procedure. — A major problem encountered in steepest-ascent calculations is the choice of the step size, $(dU)^2$. Too small a step results in regular, but slow convergence; too large a step results in irregular progress and probable convergence failure. Therefore, to insure convergence in a minimum amount of computer time, it is desirable to automatically select the largest step size possible, per iteration, consistent with the linearity requirements of the steepest-ascent technique. The selection technique used in this program is patterned after a method developed by D. S. Hague (ref. 3).

A function f_i (a payoff or constraint function) rarely is — nor does it have to be — perfectly linear to be acceptable for steepest-ascent calculations. The degree of linearity or nonlinearity is, however, important and is defined as

$$f_{NL_i} = \left| \frac{df_i^* - df_i}{df_i} \right| \tag{118}$$

where f_{NL_i} is the degree of nonlinearity representing the percentage error of the linear prediction to a parabolic approximation of the actual change of a payoff variable or a constraint. df_i^* is the actual change in f_i and df_i is the linear prediction of the change in f_i. Both the degree of linearity and choice of the function used to measure linearity must be chosen with care and are discussed below.

On a perturbed trajectory the predicted payoff function change is given by

$$d\varphi = \left[\left(I_{\varphi\varphi} - I'_{\psi\varphi} I_{\bar{\psi}\bar{\psi}}^{-1} I_{\psi\varphi} \right) \left((dU)^2 - d\bar{\psi}' I_{\bar{\psi}\bar{\psi}}^{-1} d\bar{\psi} \right) \right]^{1/2} + I'_{\psi\varphi} I_{\bar{\psi}\bar{\psi}}^{-1} d\bar{\psi} \tag{119}$$

Essentially the step size may be defined as a choice of (dU^2) and $d\bar{\psi}$. Consider the one parameter set of perturbations

$$(dU)^2 = k^2 (dU_o)^2 \tag{120}$$

$$d\bar{\psi}_i = k\, d\bar{\psi}_{oi} \tag{121}$$

where dU_o^2 and $d\bar{\psi}_{oi}$ are arbitrary nominal changes. It follows on substituting equations 120 and 121 into equation 119 that

$$d\varphi\ (k) = k\, d\varphi_o \tag{122}$$

Here $d\varphi_o$ is the change in performance resulting from the nominal step, $(dU_o)^2$ and $d\bar{\psi}_{oi}$, which results from the choice of $k = 1$. Consider figure 9 where the nominal step is denoted by the "trial" step and f_i is any of the functions φ or $\bar{\psi}$.

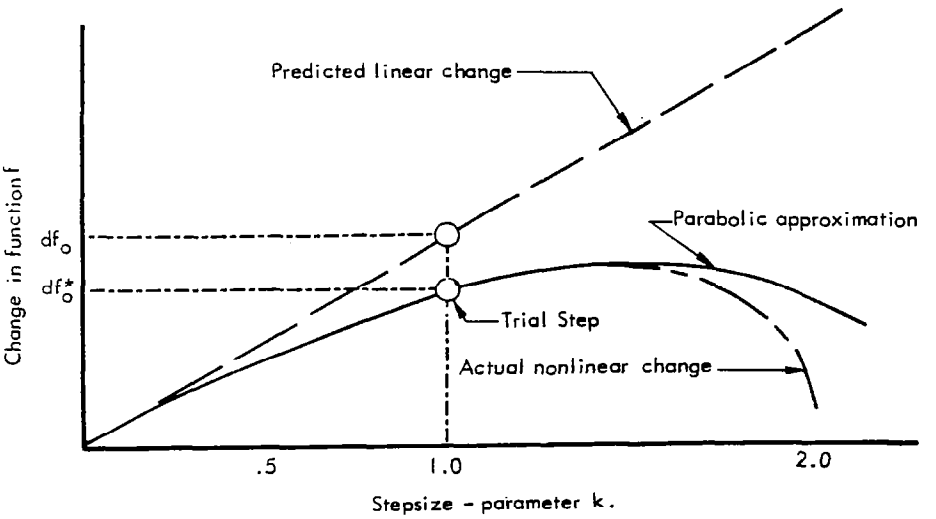


Figure 9. PARABOLIC APPROXIMATION

The predicted linear variation with k is the straight line through the origin and the point (1.0, df_o). For a small enough step the actual nonlinear change will substantially agree with this line. As the step size increases the two results will, however, tend to diverge. Suppose a parabola is fitted through the origin and the point (1.0, df_o^*) having the same slope at the origin as the linear change. The equation of this parabola is

$$df^* = (df_o^* - df_o)k^2 + df_o k \qquad (123)$$

Provided a trial step is reasonably chosen, this expression will approximate the actual nonlinear change over a considerably greater region of step sizes than did the linear prediction. Essentially, the combination of a trial result and the linear prediction allow a second-order approximation to the variation of a function with step size.

Assuming for the moment that $df_o > df_o^*$, and $df_o\,(0)/k > 0$, we can anticipate from equation 123 that the greatest change in f will occur when

$$k \approx -\frac{df_o}{2\,(df_o^* - df_o)} \qquad (124)$$

Further, on substituting this result into equations 123 and 118, it follows that at this point the nonlinearity, f_{NL}, will be about 0.5. If the actual nonlinear variation were to be parabolic this value would be the desired nonlinearity. In practice, it has been found that this value is too high. Using 0.5 in conjunction with the parabolic approximation leads to steps that are "over the hump." Experience has shown that using a nonlinearity of $f_{NL} = 0.3$ practically eliminates this problem and leads to satisfactory steps.

Now that a satisfactory degree of nonlinearity has been determined, the second question, "which of the f_{NL} to use?" has still to be answered. This is equivalent to deciding which of the functions of interest should be used to control the step size. One approach is to control with the worst-behaved function, i.e., the one whose nonlinearity is greatest. If this approach is taken, difficulties immediately appear. For example, it may be that only one function is behaving badly, the remainder being extremely linear. In this case, small steps with resulting slow convergence, or failure to converge, may result. Failure to converge here, however, does not mean divergence, but that $(dU)^2$ is so significantly reduced due to the nonlinearity of the control function that the changes in constraints and payoff functions become lost in computer noise. This results in negligible or random variation of the constraint and payoff variables from iteration to iteration. The conclusion is made, therefore, that the control must be made with the best behaved function, that is, the one having the most linearity.

To ensure that the remaining functions do not wander too far, various tests are designed to maintain convergence consistent with the linearity requirements of the steepest-ascent method. These tests, though based on experience to some extent, have proven to be a successful means of ensuring convergence.

Majority vote test: The majority vote test consists of the examination of the changes in the constraint and the payoff functions based upon the results of the previous trial. Only those functions that are more than a specified tolerance from their desired value are examined. If at least the same number of functional changes are in the proper direction as those with adverse travel, then the test is satisfied. Otherwise, the step size, $(dU)^2$, is reduced and a new trial is computed. The majority vote test, in addition to the adverse φ test discussed below, is also used to determine whether or not a valid step is acceptable, prior to going into a reverse integration of the adjoint equations of motion. A valid step is performed after a limited number of trials or a successful trial and consists of the forward integration of the equations of motion together with the partial derivative calculations.

Step size coefficients: The step size coefficients are based on the functional nonlinearities as shown in figure 9. Using the parabolic approximation, the step size that causes the best-behaved function to have a nonlinearity of 0.3 is computed. If the resulting step size coefficient, k, fails to satisfy the condition $0.5 < k < 2.0$, then a further trial at the upper or lower limit is undertaken. Experience has indicated that these bounds are reasonable ones to impose on interpolation or extrapolation using the parabolic approximation.

Adverse φ test: The adverse φ test follows the calculation of the step size coefficient. The test ensures that, if adverse travel occurs in the payoff function φ, it will not exceed a specified tolerance. When the adverse travel exceeds this tolerance, the problem is regarded as too nonlinear and $(dU)^2$ is again reduced, followed by a new trial computation.

Successful completion of the above tests is necessary to perform a valid step. Other logic, however, is incorporated to force a valid step under certain circumstances. Such tests are discussed in the user's manual.

Figure 10 is presented to illustrate the flow of the automatic convergence process.

Variable initial conditions. — The steepest-ascent technique determines, as a by-product of the optimization process, a set of influence coefficients that defines the perturbation of the payoff quantity per unit-step-change in the initial state vector. The problem is to determine the correct direction to perturb the initial conditions in order to increase performance on the next iteration. In addition, logic must be incorporated to guarantee convergence of the initial conditions to their optimum values. It is usually desirable to allow the initial condition vector to be free only in a bounded region.

The performance predicted for an iteration, given by equation 117, provides the basis for selecting the optimum initial conditions. Consider the equation

$$d\varphi\ (T) = \pm\left[\left((dU)^2 - d\bar{\beta}'\ I_{\psi\psi}^{-1}\ d\bar{\beta}\right)\left(I_{\varphi\varphi} - I_{\psi\varphi}'\ I_{\psi\psi}^{-1}\ I_{\psi\varphi}\right)\right]^{1/2} + I_{\psi\varphi}'\ I_{\psi\psi}^{-1}\ d\bar{\beta} + \lambda'_{\varphi\Omega}\ (t_o)\ \delta\bar{X}\ (t_o) \tag{125}$$

For a given problem solution, the payoff function has been optimized; that is, $d\varphi\ (T) = 0$, the constraints have been met, $d\bar{\psi} = 0$, and the initial conditions are at their optimum values so that $\delta\bar{X}\ (t_o) = 0$. Then, since $(dU)^2 \neq 0$, the relation must hold that

$$I_{\varphi\varphi} - I'_{\psi\varphi}\ I_{\psi\psi}^{-1}\ I_{\psi\varphi} = 0 \tag{126}$$

Now restricting this to the case where the constraints are met but the initial conditions have not yet been optimized, equation 125 can be reduced to

$$d\varphi\ (T) = \left[\left(\lambda'_{\varphi\Omega}\ (t_o) - I_{\psi\varphi}'\ I_{\psi\psi}^{-1}\ \lambda'_{\psi\Omega}\ (t_o)\right)\right]\delta\bar{X}\ (t_o) \tag{127}$$

if it is assumed (as is the case for the optimum) that equation 126 holds.

This expression allows the variation of the free initial conditions for each iteration but is exact only for the optimum path. The correct direction to perturb $\bar{X}\ (t_o)$ is the direction to improve performance. In obtaining optimum performance, perturbation of each free initial condition is considered independently since equation 127 shows that the effects of the perturbations are uncoupled.

Convergence is guaranteed by bounding the perturbations. The perturbation size is controlled using the rule that if the perturbation is in the same direction

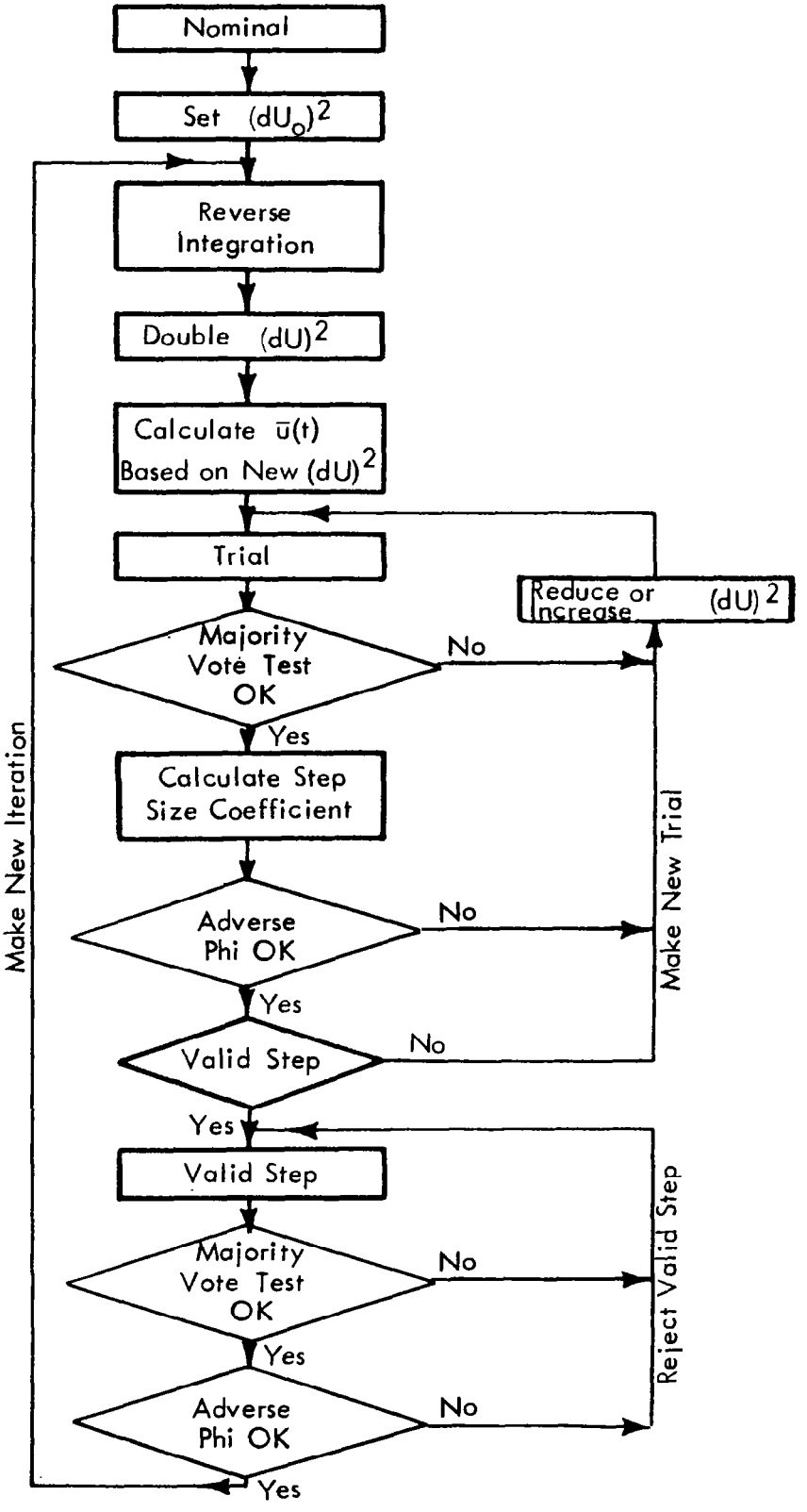


Figure 10 AUTOMATIC CONVERGENCE FLOW

for two successive iterations, then the value of the perturbation is doubled, or halved if the perturbation changes sign. The maximum initial condition change per iteration and the limiting values are specified for each free variable.

Weighting functions matrix. — A cursory glance at the steepest-ascent method might lead to the conclusion that the choice of weighting matrices is not a significant problem, but in practice this is not always so. The control variables, for example, may have widely differing abilities to affect the final values of the payoff and constraint functions. It is necessary, therefore, to differentiate between powerful and weak control variables, based on their ability to affect the optimization functions, particularly in their ability to affect the payoff function. Even with a single control variable, this problem remains, since along some subarcs the control may be weak and on others strong.

Two types of weighting are implied, one resulting from the differences in sensitivity of the multiple control variables and the other resulting from differences in sensitivity of a single control variable along a trajectory. There are two basic reasons for the use of weighting matrices. One is to speed up convergence and the other is to ensure convergence. It is fairly obvious that a well-chosen weighting matrix can improve convergence even in a well-behaved problem that involves both a weak and a strong control variable. The steepest-ascent process can be speeded up by forcing larger perturbations in the weak control variable in the initial iterations, rather than waiting for the stronger one to converge before making significant changes in the weaker one. What is not so obvious is the fact that, without a reasonably chosen weighting matrix, it is possible that the steepest-ascent method may fail to converge entirely. Generally, it is a relatively straightforward process to develop the control history of a single control variable until it lies within the neighborhood of the optimal solution, provided the problem does not involve weak and powerful subarcs.

However, the control history can only be developed into the neighborhood of the optimum since there is a region about the optimal history in which the control variable is ineffective by definition. If the problem involves terminal constraints, however, it is only ineffective for those perturbations that leave the terminal constraints unchanged. For a perturbation that merely seeks to gain performance, it may well, and usually does, remain powerful. If now there are two control variables, one weak, the other strong, it may be that even though the strong variable is in the neighborhood of the optimal solution, it still dominates the perturbations that occur as it alternates about the ideal solution. In this case, the method may fail to converge the weak control variable at all, since the effect of the weaker control can be completely obscured by the noise level of the stronger variable. Further discussion on weighting functions is given in reference 3.

Control variable power: The control variable power is used as a basis for determining a weighting function that will ensure convergence with respect to the payoff function.

The change in the payoff function resulting from a perturbation in the control variable was given by equation 97

$$d\varphi = \int_{t_o}^{T} \lambda'_{\varphi\Omega} G \, \delta\bar{u} \, dt + \left(\lambda'_{\varphi\Omega} \, \delta\bar{X} \right)_{t = t_o} \tag{128}$$

For a unit impulse perturbation in the control variables $\delta\bar{u}$, it was shown in a previous section that the response of the payoff function was

$$\delta \, (d\varphi) = \lambda'_{\varphi\Omega} G \tag{129}$$

where again $\lambda'_{\varphi\Omega}$ is a 1 x n matrix and G is an n x m matrix resulting in $\delta \, (d\varphi)$ as a 1 x m matrix giving the response of the payoff function to a unit impulse of each of the m control variables. The elements of $\delta \, (d\varphi)$ are referred to as the instantaneous payoff function sensitivites and will be designated as

$$\left(s_u^{\varphi} (t) \right)' = \lambda'_{\varphi\Omega} G \tag{130}$$

These quantities measure the power of a control variable with respect to the payoff function, provided no restrictions are imposed on the terminal constraint changes.

The control variable perturbations are closely related to the instantaneous payoff sensitivities. For the case with no terminal constraints, the control perturbations are

$$\delta\bar{u} = \pm W^{-1} G' \lambda_{\varphi\Omega} \sqrt{\frac{(dU)^2}{I_{\varphi\varphi}}} \tag{131}$$

or in terms of the instantaneous payoff sensitivities

$$\delta\bar{u} = \pm W^{-1} s_u^{\varphi} \sqrt{\frac{(dU)^2}{I_{\varphi\varphi}}} \tag{132}$$

Thus the optimum (steepest-ascent) perturbation varies directly with the inverse weighting matrix and the instantaneous sensitivities. If terminal constraints are considered in the problem, the control variable perturbations are (from equation 116 with fixed initial conditions)

$$\delta\bar{u} = \pm W^{-1} \left(s_u^{\varphi} - G' \lambda_{\psi\Omega} I_{\psi\psi}^{-1} I_{\psi\varphi} \right) \left[(dU)^2 - d\bar{\psi}' I_{\psi\psi}^{-1} d\bar{\psi} \right]^{1/2} \tag{133}$$

$$+ W^{-1} G' \lambda_{\psi\Omega} I_{\psi\psi}^{-1} d\bar{\psi}$$

The results above suggest an approach to the problem of false convergence. The problem is due to small perturbations in the weak control variable. The

inverse weighting matrix based on the control variable sensitivities can be used to amplify the effect of the weaker control variables. Using this type of weighting matrix, effectively the basis of optimization is changed from that perturbation having the greatest change in the payoff function to that perturbation having the greatest change in φ with all control variables being equally important and must therefore be perturbed by a reasonable amount.

The possibility of failing to converge to the desired constraints is more remote than that of failing to converge the payoff function. The dominant control variables for the payoff function are very often the dominant control variables for the constraints and hence will continue to be perturbed until the constraints are achieved. The terminal constraints may often be met without the optimum control history. The failure to meet the terminal constraints is immediately recognized whereas the optimum performance is verified only by starting from a considerably different nominal. Weighting matrices based on constraint sensitivities ($G' \lambda_{\psi\varphi}$) and a mixed payoff and constraint sensitivity have been discussed by Hague in reference 3 but are not considered here.

The overall control variable power may be determined by integrating the instantaneous payoff sensitivities where the constraints are again ignored

$$S_u^{\varphi} = \int_{t_o}^{T} \left| s_u^{\varphi}(t) \right| dt \tag{134}$$

The elements of this column matrix are referred to as the integrated payoff sensitivities. Both the integrated payoff sensitivities and the instantaneous payoff sensitivities may be used as the basis for the weighting functions.

The instantaneous payoff sensitivities can be used to construct a weighting matrix to balance weak and strong subarcs for a given control variable. The integrated payoff sensitivities are used to obtain the weighting matrix to balance the power of the multiple control variables. The matrix is of the form

$$\left[w_{ii} \right]^{-1} = \frac{1}{m+1} \left[1 + \frac{\sum_{j=1}^{m} S_{u_j}^{\varphi}}{S_{u_i}^{\varphi}} \right] \quad i = 1, 2, \ldots, m \tag{135}$$

where the w_{ii} are the diagonal elements of a square matrix. Note that when all the control variables are equally powerful, i.e., $S_{u_j}^{\varphi} = S^{\varphi}$

$$\left[w_{ii} \right]^{-1} = \frac{1}{m+1} \left[1 + \frac{m S^{\varphi}}{S^{\varphi}} \right] = I \tag{136}$$

the identity matrix.

This weighting matrix will ensure that each control variable is perturbed to the same order. A time-varying weighting matrix based on the instantaneous payoff sensitivity can have the form

$$\left[w_{ii} \right]^{-1} = \left[1 + \frac{s^{\varphi}_{u_{max}}}{s^{\varphi}_{u_i}} \right] \tag{137}$$

where $s^{\varphi}_{u_{max}}$ is the largest value of s^{φ}_{u} along the trajectory. This weighting matrix was not automated in STOP since it was felt that a satisfactory matrix could be input based on the knowledge of the problem by the user. This also allows the user to weight regions of the trajectory as his experience dictates.

Nominal Trajectory Generation

The steepest-ascent method begins with a nominal flight generated by an initial choice for the control variable history. Essentially, the only requirements on the nominal flight path are that it meet the stopping condition and be in the neighborhood of the optimum. A reasonable flight path can sometimes be generated open loop, that is, by inputing a table for the control variable history. In many cases, however, it is difficult and time consuming to determine a control table that will even meet the stopping condition. Therefore, capability is included to generate closed-loop trajectories for the nominal. The nominal guidance modes are shown in table I.

Table I: NOMINAL GUIDANCE OPTIONS

Mode	Guidance Type	Control Variable
0	$\theta = f(t)$	θ
1	$\alpha = 0$	θ
2	$\alpha = f(t)$	θ
3	$\gamma = f(t)$	θ
4	$\dot{\gamma} = f(t)$	θ
5	$h = f(V)$	θ
6	$h = f(M)$	θ
7	$\dot{\gamma} = 0$	θ
8	$\gamma = 0$	θ
9	$\dot{V} = 0$	θ
10	$\dot{M} = 0$	θ
11	$\dot{V} = 0$	η
12	$\dot{M} = 0$	η

The basic method for determining the control variable in STOP is a table lookup of all the selected variables as functions of time. The function of the guidance mode is to override the tabulated input value for one of the control

variables. Therefore a control table must always be input with values for each control variable. The values of the control variables in the table which will be overriden may be a dummy value since it is not used. A brief discussion of several methods for obtaining angle of attack or throttling is given prior to discussing each guidance mode. The three schemes are: (1) determine angle of attack to produce a given $\dot{\gamma}$; (2) determine the angle of attack to give a required $\dot{V}$; and (3) determine the throttling parameter to produce a given $\dot{V}$.

Angle of attack from known $\dot{\gamma}$. — The equation of motion for flight path angle is given by

$$\dot{\gamma}_R = \frac{(T \sin \alpha + L) \cos \varphi}{m V_R} - \left(\frac{g}{V_R} - \frac{V_R}{r}\right) \cos \gamma_R + \frac{\omega^2 r}{V_R} \cos \beta \left(\cos \beta \cos \gamma_R + \sin \beta \sin \gamma_R \sin \psi_R\right) + 2 \omega \cos \beta \cos \psi_R \tag{138}$$

or for simplicity

$$\dot{\gamma}_R = \frac{(T \sin \alpha + L) \cos \varphi}{m V_R} + GF \tag{139}$$

where

$$GF = -\left(\frac{g}{V_R} - \frac{V_R}{r}\right) \cos \gamma_R + \frac{\omega^2 r}{V_R} \cos \beta \, (\cos \beta \cos \gamma_R + \sin \beta \sin \gamma_R \sin \psi_R) + 2 \omega \cos \beta \cos \psi_R \tag{140}$$

If the angle of attack and bank angle are small so that $\alpha \doteq \sin \alpha$ and $\cos \varphi \doteq 1$, equation 139 can be written as

$$\dot{\gamma} = \frac{\left(T + L_\alpha\right) \alpha + L_{\alpha=0}}{m V_R} + GF \tag{141}$$

For a given $\dot{\gamma}$, the angle of attack required is obtained by solving equation 141 for α,

$$\alpha = \frac{\left(\dot{\gamma}_{req} - GF\right) m V_R - L_{\alpha=0}}{T + L_\alpha} \tag{142}$$

Angle of attack for a known $\dot{V}$. — The angle of attack for a required $\dot{V}$ is based on equation 35

$$\dot{V}_{req} = \frac{T \cos \alpha - D}{m} - g \sin \gamma_R + \omega^2 r \cos \beta (\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R) \quad (143)$$

or for simplicity

$$\dot{V}_{req} = \frac{T \cos \alpha - D}{m} + VF \quad (144)$$

where

$$VF = - g \sin \gamma_R + \omega^2 r \cos \beta (\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R) \quad (145)$$

Making the assumption that $\cos \alpha \approx 1 - \alpha^2/2$ and expanding the drag in terms of angle of attack gives

$$D = D_0 + D_1 \alpha + D_2 \alpha^2 \quad (146)$$

where

$$D_0 = \left[C_{D_0} + \frac{\partial C_D}{\partial C_L^2} \left(C_{L_{\alpha=0}} - C_{L_M}\right)^2\right] q S$$

$$D_1 = 2 \frac{\partial C_D}{\partial C_L^2} C_{L_\alpha} \left(C_{L_{\alpha=0}} - C_{L_M}\right) q S$$

$$D_2 = \frac{\partial C_D}{\partial C_L^2} C_{L_\alpha}^2 q S$$

Equation 144 can be written as

$$\dot{V}_{req} = \frac{T - D_0 - D_1 \alpha - \left(\frac{T}{2} + D_2\right) \alpha^2}{m} + VF \quad (147)$$

For a given $\dot{V}_{req}$, the angle of attack can be obtained by solving the quadratic equation

$$\left(\frac{T}{2} + D_2\right) \alpha^2 + D_1 \alpha + (\dot{V}_{req} - VF) m - (T - D_o) = 0 \quad (148)$$

Equation 148 yields two values for α. The larger value is used since it represents the usual case.

In some cases the $\dot{V}$ required may be larger than the maximum $\dot{V}$ that can be attained. In this case the best that can be done is to fly at the angle of attack

for maximum $\dot{V}$. This indicates that the input h - V or h - M curves exceed the capability of the vehicle through adjusting $\dot{V}$. The angle of attack for maximum $\dot{V}$ is obtained by setting

$$\frac{-D_1 - 2\left(\frac{T}{2} + D_2\right)\alpha}{m} = 0 \tag{149}$$

and solving for α,

$$\alpha = -\frac{D_1}{2\left(\frac{T}{2} + D_2\right)} \tag{150}$$

Throttling for a known $\dot{V}$. — The equation for thrust is given in general as

$$T = \left(K_T\, T_{vac}\, \eta - \frac{P_\infty\, A_e}{144}\right) \cos\,\delta \tag{151}$$

Solving for η gives

$$\eta = \frac{\frac{T}{\cos\,\delta c} + \frac{P_\infty\, A_e}{144}}{K_T\, T_{vac}} \tag{152}$$

where T is obtained by solving equation 144

$$T = \frac{(\dot{V}_{req} - VF)\, m + D}{\cos\,\alpha} \tag{153}$$

Substituting into equation 152 gives

$$\eta = \frac{\frac{(\dot{V}_{req} - VF)\, m + D}{\cos\,\alpha\,\cos\,\delta c} + \frac{P_\infty\, A_e}{144}}{K_T\, T_{vac}} \tag{154}$$

There are no approximations here since T is defined as a linear function of η.

Guidance Modes. — Each of the nominal guidance modes, given in table I, is detailed below. Modes 3 through 10 calculate the θ control variable from $\theta = \alpha + \gamma$.

Mode 0: This open-loop guidance table is used as it was input by the user.

Mode 1: The angle of attack is zero. The control variable θ is obtained by

$$\theta = \gamma \tag{155}$$

Mode 2: The program interprets the first control variable input in the control table as angle of attack for the nominal trajectory only. The control variable θ is calculated as

$$\theta = \alpha_{input} + \gamma \tag{156}$$

Mode 3: This mode calculates θ where the flight path angle is given as a function of time by a guidance table input, and uses the techniques of calculating the angle of attack required to fly a given $\dot{\gamma}$ (equation 142). The required $\dot{\gamma}$ is obtained by

$$\dot{\gamma}_{req} = \dot{\gamma}_I - \frac{(\gamma - \gamma_I)}{\Delta t} \tag{157}$$

where $\dot{\gamma}_I$ is the slope of the input γ with respect to time

γ is the state variable

γ_I is the desired flight path angle from the input table

Δt is an increment of time to correct for the γ error (currently $\Delta t = 1$ sec)

Mode 4: This mode calculates θ when the $\dot{\gamma}$ is given as a function of time by a guidance table input. The angle of attack is calculated by equation 142 from the $\dot{\gamma}$ in the guidance table.

Mode 5: This mode calculates θ where the altitude is given as a function of velocity by a guidance table input. The required $\dot{V}$ is obtained from

$$\dot{V}_{req} = \frac{h}{(dh/dV)_{req}} \tag{158}$$

The desired h - V curve is flown by adjusting angle of attack to give the proper $\dot{V}$. Inaccuracies in the numerical procedure may cause drift from the desired h - V. This is compensated by adding a feedback term as

$$\left(\frac{dh}{dV}\right)_{req} = \left(\frac{dh}{dV}\right)_{input} \pm \frac{(h - h_{input})}{25} \tag{159}$$

The + sign is used for descent phases and the - sign is used for ascent along the h - V or h - M curves. The angle of attack is obtained from equation 148.

Mode 6: This mode calculates θ where altitude is given as a function of Mach number by a guidance table input. The slope dh/dM is obtained from the input data. The technique of mode 5 is used where

$$\left(\frac{dh}{dV}\right)_{input} = \frac{1}{a}\left(\frac{dh}{dM}\right)_{input} \tag{160}$$

where a is the speed of sound along the flight path

Mode 7: This mode calculates θ which produces $\dot{\gamma} = 0$. Equation 142, for calculating angle of attack for a given $\dot{\gamma}$, is used as in mode 4 but no input guidance table is required.

Mode 8: This mode calculates θ which will give $\gamma = 0$. The method described under mode 3 is used but no guidance tables are input. If the flight path angle is not zero at the start of the phase, the feedback term causes the vehicle to pitch until $\gamma = 0$.

Mode 9: This mode calculates θ for which the flight path acceleration ($\dot{V}$) is zero. Equation 148 is used to calculate the angle of attack for which $\dot{V} = 0$.

Mode 10: This mode calculates θ for which the Mach number is constant. The requirement of $\dot{M} = 0$ is obtained through a $\dot{V}_{req}$. Since $V = M\,a$

$$\dot{V} = \dot{M}\,a + M\,\frac{\partial a}{\partial h}\,\dot{h} \tag{161}$$

which, since $\dot{M} = 0$ for this mode, reduces to

$$\dot{V}_{req} = M\,\frac{\partial a}{\partial h}\,\dot{h} \tag{162}$$

The angle of attack is obtained from equation 148.

Mode 11: This mode calculates the throttling (η) for constant velocity. The throttling is calculated from equation 154 for $\dot{V}_{req} = 0$.

Mode 12: This mode calculates the throttling (η) for constant Mach number. The throttling is calculated using equation 162 for $\dot{V}_{req}$ and equation 154.

Use of guidance options — A nominal trajectory generation relies on the user to input various modes that will take the vehicle to the stopping condition. The flight path is divided into various guidance phases, each controlled by one of the guidance modes described above. A phase ends upon reaching an input value of any of the 85 state, auxiliary, and control variables or time. The number of phases is limited to 12, of which only 5 can use the input tables reserved for guidance modes. The tables required for modes 0 and 2 are not included in the five reserved tables. An example statement of a guidance phase buildup is:

Phase 1. Fly a prescribed h - V (mode 5) to an altitude of 30,000 feet

Phase 2. Fly a $\dot{\gamma} = -0.2$ degree per second (mode 4) and decrease to a γ of 0 degree

Phase 3. Fly at constant velocity using throttling as the control variable (mode 11) to a range of 1000 nautical miles

Phase 4. Fly a $\dot{\gamma} = -0.2$ degree per second (mode 4) decreasing to a γ of -1 degree

Phase 5. Fly a prescribed h - V (mode 5) to a velocity of 500 feet per second.

The generation of nominal trajectories using modes 3 through 10 is limited to vehicles with aerodynamics given by

$$C_L = C_{L_{\alpha=0}} + C_{L_\alpha}\,\alpha \tag{163}$$

$$C_D = C_{D_0} + \frac{\partial C_D}{\partial C_L{}^2}\left(C_L - C_{L_M}\right)^2 \tag{164}$$

or

$$C_L = C_{L_\alpha}\,\alpha \tag{165}$$

$$C_D = C_{D_0} + \frac{\partial C_D}{\partial C_L{}^2}\,C_L{}^2 \tag{166}$$

and thrust is independent of angle of attack. This restriction is made because the amount of computer time required to generate a nominal can be reduced by using the simpler representations given in equations 141 and 147.

Additional Options

Several options are included in STOP which permit the user to optimize special problems with fewer inputs than required in the usual sense. These are referred to as the gamma tilt, maximum payload, and circular satellite options. Each is discussed briefly below.

Gamma tilt. — The gamma tilt option is required for vertical takeoff (VTO) boosters. The tilt angle, γ_{tilt}, is input as a function of time in a table. Termination of the tilt maneuver is accomplished by a stage stopping time; consequently, the tilt maneuver must be counted as a stage. If flight path angle, γ, is selected as a free initial condition, then the perturbation on γ occurs at the end of tilt.

Circular satellite. — This option permits the usage of relative circular satellite velocity, V_{cs_R}, as a stopping condition or constraint. The option eliminates the calculations by the user as given below

$$V_{cs_R} = \left[\left(r_D\,\omega\cos\beta - V_{cs_I}\cos\psi_I\right)^2 + \left(V_{cs_I}\sin\psi_I\right)^2\right]^{1/2} \tag{167}$$

where $V_{cs_I} = \sqrt{\frac{\mu}{r_D}}$

and

$\mu = 14.081718 \times 10^{15}$

r_D = desired final radius

Maximum Payload. — If weight is the payoff variable and also a free initial condition, then the program will add payload and adjust the last-stage propellant weight at launch so that the burnout weight will be a maximum. A plot of payload versus last-stage propellant illustrates the manner in which this option operates (figure 11).

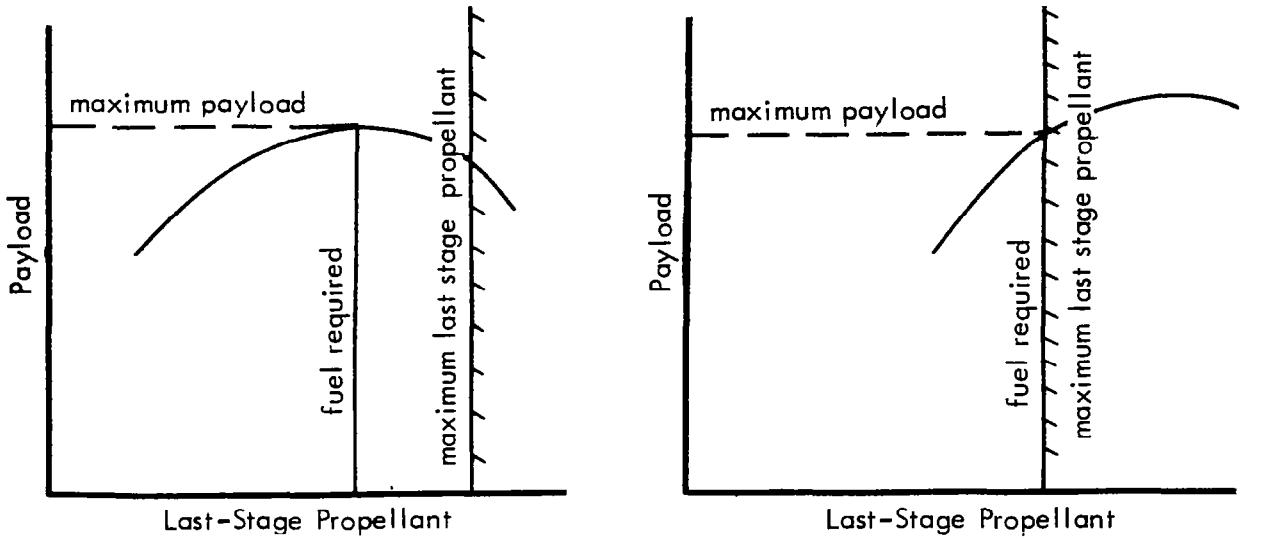


Figure II. MAXIMUM PAYLOAD OPTION

Notice that when the fuel required for maximum payload is less than the maximum last-stage propellant, the true maximum payload is attainable. Otherwise, the payload limit is restricted by the amount of fuel the last stage may hold.

Conclusions and Recommendations

The computer program developed for the contract meets all requirements of the statement of work (ref. 1) and in addition has the capability to select the optimum initial conditions for a case with free initial conditions.

For multistage vehicles, the staging times must be input in the present program. It would be desirable, in a future modification, to have the optimal staging capability using the algebraic steepest-ascent method that allows staging on variables other than time. This option would permit optimization of coast and burn times to increase performance and give added flexibility to the program.

Multiple-vehicle capability is also desirable in a program of this type. This would enable optimization of intercept problems involving fighter aircraft, rendezvous between spacecraft and satellites, space rescue, etc.

Program Assumptions and Limitations

The assumptions and limitations that were made in the development of STOP are detailed in this section. The manner in which the assumptions affect the utility of the program are indicated. The effects of the limitations on the program are apparent.

Assumptions. —Simulation: The basic assumption for STOP is that the vehicle is represented as a point mass lifting body. This precludes any effect of the vehicle short-period dynamics on the optimum flight path. The results of the program in optimizing the flight path for a vehicle having high pitch rates have not been thoroughly substantiated. For aircraft and most boosters, the pitch rates do not affect the optimal results appreciably.

Earth model: The earth is assumed to be a rotating sphere with an inverse-square gravity field. The effect of the differences between the actual and assumed earth representation on the optimum flight path are small for the type of vehicles for which the program is designed.

Aerodynamics and propulsion: The aerodynamics and propulsion data for the vehicle can be input in several ways, depending on the availability of the data or type of vehicle. The most used aerodynamic representation for aircraft is the drag polar and lift as a linear function of angle of attack.

Generation of nominal flight paths: The generation of nominal flight paths by STOP for modes 3 through 10 assumes the drag polar and the linear lift representation and that thrust is not dependent on angle of attack. This restriction is not considered serious because this is the most common aerodynamic and thrust representation for aircraft. A method using more general aerodynamics and engine data can be formulated but the calculations require iterative procedures and are very time consuming.

Control variables: The control variables selected for STOP are pitch angle, bank angle, throttling, and wing sweep. The pitch angle is defined as $\theta = \gamma + \alpha$. The bank angle is used to perform out-of-plane maneuvers and assumes coordinated turns with no slideslip. Note that, for nonzero bank, θ is not a physical angle.

Payoff, constraints, and stopping conditions: The steepest-ascent procedure as formulated for STOP allows only state variables to be used as the payoff, constraint, and stopping functions. Enroute placards are formulated as auxiliary state variables for which terminal constraints are applied.

Atmosphere: The earth atmosphere is input as a single subroutine, using a calling sequence to transmit the data. Any subroutines using the same calling sequence can be used for the atmosphere calculation.

Staging: The trajectory can be staged only on time. Weight jettisons can be performed only at staging.

Limitations. — State variables: The number of state variables is limited to 15. These may be selected from the 20 state variables defined by equations of motion and 20 state variables defined by enroute constraints. Altitude, flight path angle, and velocity must always be selected.

Control variables: The maximum number of control variables is four. Pitch angle, bank angle, throttling, and wing sweep are incorporated.

Constraints: A maximum of 13 constraints plus a stopping condition can be imposed on a problem. Of the 13, the number of enroute constraints is limited only by the number of data tables allowed for input of maximum and minimum placards. Twenty tables are reserved for the placards, which allows a maximum of 10 placards.

Stages: A maximum of 14 computer stages are permitted. The number of physical stages is not necessarily the same as the number of computer stages.

Data deck: A maximum of 1000 data cards are allowed for tables 1 through 30 plus 31 through 34 for any stage.

Equations of motion: The point mass equations of motion include the following limitations

$$\dot{\gamma} = 0 \quad \text{if } |V_R| < 100 \text{ fps or during a tilt maneuver}$$

$$\dot{\psi} = 0 \quad \text{if } |V_R| < 100 \text{ fps or}$$

$$|90 \pm \gamma_R| < 0.1 \text{ degrees or}$$

$$|90 \pm \beta| < 0.1 \text{ degrees}$$

$$\dot{\lambda} = 0 \quad \text{if } |90 \pm \beta| < 0.1 \text{ degrees.}$$

Input Data Preparation

The following section presents a brief listing of the purpose of each of the nine (9) card sets that make up the Input Data Deck. This is followed by detailed instructions for preparing the data deck for a problem solution. Comments and notes are included to give the user further information to aid the setup. Prior to sending the data through the computer, it should be checked for the most common errors made in data preparation, as discussed under "Trouble Shooting."

Input Deck Summary — The Input Data Deck is composed of nine (9) card sets of which card sets 1 through 5 and 9 are required. Card sets 6, 7, and 8 may or may not be included in the Input Data Deck, depending on the problem to be solved.

Card Set	Number of Cards	Reference Pages	Purpose
1	1	53	Title
2	1	53	Print and Storage frequencies, type of solution, number of iterations and computer stages.
3	1-23	54-63 (See also figs. 12 and 13)	These cards, by use of control (NC) numbers, define the input options, output options, the state, auxiliary, and control variables as well as the use of a number of special-purpose options. Included in this selection are the equations of motion and the enroute placards.
4	3-5	64-65	Initial condition data including various physical constants, value of stopping condition, integration error limits, stopping condition tolerance, DUSQ, and initial values of the various state variables.
5	8-16	65-66	Computer stage dependent parameters such as: initial weight, final stage time (trajectory time), aerodynamic reference area, nozzle cant angle, total impulse correction factor, weight flow correction factor, inert weight flow and minimum allowable integration step.
6	0-13	66-67	Constraint dependent parameters; desired values and final tolerances for the various terminal constraints and enroute placards.
7	0-7	67	Free initial conditions, inputs maximum and minimum value and initial increment per iteration of each free initial condition.
8	0-12	67-68	Nominal guidance data; defines the various options used in the nominal trajectory generation for each stage.

Card Set	Number of Cards	Reference Pages	Purpose
9	2-58	68-76	All the tabulated input such as control function, weighting function, gamma tilt program, en-route placards, aerodynamic data (lift, drag, etc.), and engine data.

Preparation — The following are detailed instructions for preparing the data deck for a problem solution:

Card Set 1

Title Card — Format A72 — 1 Card

Card 1

The contents of this card will be printed at the start of the case. It may contain any desired information in columns 1-72 or left blank, but it must be the first card in the data case.

Card Set 2

Controls — Format 14I5 — 1 Card

Card 1

Column	Name	Description
1-5	IPRNT1	Printout of forward trajectory occurs every IPRNT1 integration steps.
6-10	IPRNT2	Printout of influence coefficients occurs every IPRNT2 integration steps during backward integration if NC(7) = 1.
11-15	ISTOR1	Storage of time, partials, and plot data occurs every ISTOR1 integration steps along forward trajectory. (Recommended value = 1 for rockets, 2 for airbreathers.)
16-20	ISTOR2	Storage of impulse response functions occurs every ISTOR2 integration steps along backward trajectory. (Recommended value = 1 for rockets, 2 for airbreathers.)
21-25	NARBY	Degree of problem solution indicator = 0 Integrate nominal trajectory only and quit = 1 Integrate nominal trajectory and backward trajectory only and quit = 2 Iterate (NARBY = 2 required for optimization)

Column	Name	Description
26-30	NITC	Maximum number of iterations
31-35	MSTAGE	Number of (computer) stages
		(1 ≤ MSTAGE ≤ 14)

Card Set 3

NC Controls — Format 14I5 — 1-23 Cards

Each card contains seven pairs of integers. The first integer is the index of the NC to be set and the second integer is the desired value of the NC. Every card must contain seven pairs of integers, except the last card. The NC's may be input in any order. The reading of the NC's is terminated by a blank field in columns 61-65 of the last card. Thus, if the number of NC's to be read is not a multiple of 7, the last card will contain one to six pairs of integers. Otherwise, the last card will be blank. The explanation of the NC's is as follows. NC's not read in will be zero.

NC Index	Name	Description
1	NC(1)	Plot paper option
		= 0 use 11 x 17 millimeter paper
		= 1 use 8-1/2 x 11 millimeter paper
2	NC(2)	Not used
3	NC(3)	Plot identification option
		ORTHOMAT (NC(11) < 0)
		= 0 Use symbols
		= 1 Use colors
		SC 4020 (NC(11) > 0)
		= 0 9 x 9 vellum only
		= 1 microfilm only
		= 2 9 x 9 vellum only
		= 3 9 x 9 vellum and microfilm plots
4	NC(4)	Number of constraints + 2
		(Count both terminal constraints and enroute placards)
		(2 ≤ NC(4) ≤ 15)

NC Index	Name	Description
5	NC(5)	Number of free initial conditions $0 \leq NC(5) \leq 7$ Note: If a tilt maneuver is selected (NC(14) = 1), and if gamma is a free initial condition, then the free initial condition will be applied at the end of tilt (start of computer stage 2)
6	NC(6)	Maximum payload option = 0 Ignore option = 1 Last-stage propellant will be limited to the maximum last-stage propellant (MLSP, input in CARD SET 4). Weight must be a free initial condition for this option and NC(19) = 1 is required. Note: If NC(6) = 0 and NC(19) = 1, final weight is maximized by minimizing last-stage fuel.
7	NC(7)	Influence coefficient print option = 0 Do not print influence coefficients = 1 Print influence coefficients (Usual value = 0)
8	NC(8)	Punch option = 0 Do not punch control and restart tables used for last iteration = 1 Punch control and restart tables used for last iteration (Usual value = 1) Notes: 1) If a guidance mode is selected (NC(13) > 0) for a nominal trajectory (NARBY=0), then if NC(8)=1, STOP will punch a control table (table 0), which may be used to generate the same trajectory open loop (NC(13)=0). 2) The control table punched by STOP is in octal (Format 6 Ø 12).
9	NC(9)	Extremal option = 1 maximize payoff variable = -1 minimize payoff variable = 0 solve boundary value problem

NC Index	Name	Description
10	NC(10)	Circular satellite option = 0 Ignore option = -1 Stop on V_{cs} at desired final altitude. (NC(18) = 4 is required for this option = N Nth constraint is $V = V_{cs}$ This requires NC(N+19)=4
11	NC(11)	Plot option = 0 Do no plot = N Plot first, last, and every Nth iteration on SC 4020 plotter = -N plot first, last, and every Nth iteration on ORTHO-MAT plotter Note: Only stored points are plotted.
12	NC(12)	Numerical partial check option = 0 Do not check partials = N Check partials every Nth stored integration step and printout analytical and numerical values (Usually NC(12) = 0)
13	NC(13)	Nominal trajectory guidance mode option = 0 Open loop (control history from table 0) = N Closed loop, N phases (see card set 8 if NC(13) > 0) (Usually NC(13) = 0), $1 \leq N \leq 12$
14	NC(14)	Tilt maneuver option = 0 Do not use tilt maneuver = 1 Use tilt maneuver for stage 1 (table 2) (NC(14) = 1 for VTO systems) and the tilt maneuver counts as a stage
15	NC(15)	Skin friction option = 0 No CDF = 1 H_{ref}, $\partial C_D / \partial H = f(M)$. This requires that table 23 be input
16	NC(16)	Control table input option = 0 input decimal table = 1 input octal table Note: NC(16) = 1 is required to accept tables punched by the program

NC Index	Name	Description
17	NC(17)	Restart table option = 0 No restart table (use for nominal trajectory) = 1 Input restart table after table 0 Note: A restart table is obtained whenever a control variable table is generated by the program as a result of previous iterations. See punched card output discussion.
18	NC(18)	State variable index for stopping condition (figure 12, column 2). See input instructions check list.
19	NC(19)	State variable index for payoff quantity (figure 12, column 2)
20	NC(20)	State variable index for first constraint (figure 12, column 2)
21	NC(21)	State variable index for second constraint (figure 12, column 2)
'	'	
'	'	
'	'	
'	'	
32	NC(32)	State variable index for 13th constraints (figure 12, column 2) Note: Enroute placards are state variables and count as constraints. See figure 12 for list of state variables.
33	NC(33)	Engine option for stage 1
34	NC(34)	Engine option for stage 2
'	'	
'	'	
46	NC(46)	Engine option for stage 14 Note: The engine option for each stage is input as a two-digit number. The left digit specifies the type of thrust table as follows:

VARIABLE TYPE	VARIABLE INDEX	NC INDEX	PLOT INDEX	FORTRAN NAME	SYMBOL	DESCRIPTION	UNITS
STATE (EQUATION OF MOTION)	1	121	1	X(K1)	W	WEIGHT	LB
	2	122	2	X(K2)	H	ALTITUDE	FT
	3	123	3	X(K3)	γ	RELATIVE FLIGHT PATH ANGLE	DEG
	4	124	4	X(K4)	V	RELATIVE VELOCITY	FPS
	5	125	5	X(K5)	β	LATITUDE ANGLE	DEG
	6	126	6	X(K6)	ψ R	RELATIVE HEADING ANGLE	DEG
	7	127	7	X(K7)	λ	LONGITUDE ANGLE	DEG
	8	128	8	X(K8)	TD	DUMMY TIME	SEC
	9	129	9	X(K9)	RNG	PATH RANGE ALONG EARTH'S SURFACE	N MI
	10	130	10	X(K10)	AHI	AERODYNAMIC HEATING INTEGRAL	$FT\text{-}LB/FT^2$
	11	131	11	X(K11)	Δ V	IDEAL RELATIVE DELTA VELOCITY	FPS
	12	132	12	X(K12)	GL	GRAVITY LOSS	FPS
	13	133	13	X(K13)	DL	DRAG LOSS	FPS
	14	134	14	X(K14)	TVL	THRUST VECTORING LOSS	FPS
	15	135	15	X(K15)	ER	RELATIVE SPECIFIC ENERGY	FT
	16	136	16	X(K16)		AVAILABLE FOR EXPANSION NOT DEFINED	
	17	137	17	X(K17)			
	18	138	18	X(K18)			
	19	139	19	X(K19)			
	20	140	20	X(K20)			
STATE (ENROUTE PLACARD)	21	141	21	X(K21)	θ^*	PITCH ANGLE PLACARD = F(t)	DEG^2/SEC
	22	142	22	X(K22)	$\dot{\varphi}^*$	BANK ANGLE PLACARD = F(t)	DEG^2/SEC
	23	143	23	X(K23)	η^*	THROTTLE PLACARD = F(H, M)	SEC
	24	144	24	X(K24)	Λ^*	WING SWEEP PLACARD = F(H, M)	DEG^2/SEC
	25	145	25	X(K25)	α^*	ANGLE OF ATTACK PLACARD = F(H, M)	DEG^2/SEC
	26	146	26	X(K26)		NOT DEFINED	
	27	147	27	X(K27)	HD*	ALTITUDE RATE PLACARD = F(t)	FPS
	28	148	28	X(K28)	Q*	DYNAMIC PRESSURE PLACARD = F(t)	$(PSF)^2$ SEC
	29	149	29	X(K29)	Q*	DYNAMIC PRESSURE PLACARD = F(M)	$(PSF)^2SEC$
	30	150	30	X(K30)	$Q\alpha^*$	$Q\alpha$ PLACARD = F(M)	$(PSF\text{-}DEG)^2SEC$
	31	151	31	X(K31)	TEMT*	TOTAL TEMPERATURE PLACARD = F(t)	$(^\circ R)^2SEC$
	32	152	32	X(K32)	N*	NORMAL LOAD FACTOR PLACARD = F(H, M)	SEC
	33	153	33	X(K33)	RPA*	RESULTANT PHYSIOLOGICAL ACCEL. PLACARD = F(t)	$(FT/SEC^2)^2SEC$
	34	154	34	X(K34)	H*	ALTITUDE PLACARD = F(M)	FT^2SEC
	35	155	35	X(K35)	Δ P*	SONIC BOOM OVERPRESSURE PLACARD = F(λ, β)	$(PSF)^2SEC$
	36	156	36	X(K36)	M*	MACH NUMBER PLACARD = F(H)	SEC
	37	157	37	X(K37)		NOT DEFINED	
	38	158	38	X(K38)	γ^*	GAMMA PLACARD = F(H, M)	DEG^2 SEC
	39	159	39	X(K39)		NOT DEFINED	
	40	160	40	X(K40)		NOT DEFINED	

FIGURE 12: STATE VARIABLES

Thrust Option

Digit	Thrust Table
0	Coast (no thrust table)
1	Vacuum thrust (lb), nozzle exit area (in^2), vacuum specific impulse (sec) = F (stage time). Table format 1 for 3 dependent variables
2	Vacuum thrust (lb), nozzle exit area (in^2), vacuum specific impulse (sec) = F (total time). Table format 1 for 3 dependent variables
3	$T = F(\alpha, H, V)$
4	$T = F(\alpha, H, M)$
5	$T = F(\alpha, M, H)$
6	$T/P = F(\alpha, H, M)$
7	$CT = F(\alpha, H, V)$
8	$CT = F(\alpha, H, M)$
9	$CT = F(\alpha, M, H)$

Table format 2 for one, two, or three independent variables

Note: (1) P is atmospheric pressure for option 6; (2) For thrust options 3 through 9, the number of engines is input in columns 11 to 20 on the first data card in table 31. If this field is left blank, STOP will assume one engine. The thrust from table 31 (and fuel flow from table 32 if a WDOT option is selected) will then be multiplied by the number of engines.

The right digit specifies the type of fuel flow table as follows:

Fuel Flow Option

Digit	Fuel Flow Table
0	Coast (no fuel flow table)
1	$WDOT = F(H, M, \eta)$
2	$SFC = F(\eta, H, V)$
3	$SFC = F(\alpha, H, V)$

Table format 2 for one, two, or three independent variables

Fuel Flow Table (Continued)

Digit		
4	WDOT = F (η, H, V)	Table format 2 for one, two, or three independent variables
5	WDOT/P = F (H, M, η)	
6	SFC = F (η, M, H)	
7	SFC = F (α, M, H)	
8	WDOT = F (η, M, H)	
9	WDOT = F (α, M, H)	

Note: P is atmospheric pressure for option 5

NC Index	Name	Description
47	NC(47)	Aerodynamics option for stage 1
48	NC(48)	Aerodynamics option for stage 2
⋮	⋮	
60	NC(60)	Aerodynamics option for stage 14

Note: The aerodynamics options available are given below and are identified by the value assigned to NC(47) - NC(60).

Aerodynamic Option

NC Value	Aerodynamic Table
0	Vacuum (no aerodynamic table)
1	CD = F (H, M, α) CL = F (H, M, α)
2	CD = F (Λ, M, α) CL = F (Λ, M, α)
3	CD_o, $\partial CD/\partial CL^2$, $\partial CL/\partial \alpha$ = F (M)
4	CA, $\partial CN/\partial \alpha$, $\partial CN/\partial \alpha^3$ = F (M)
5	CD_o, $\partial CD/\partial CL^2$, $CL_{\substack{min \\ drag}}$, $\partial CL/\partial \alpha$, $CL_{\alpha=0}$ = F (Λ, M)

Note: (1) Options 1 and 2 use table format 2 for one, two, or three independent variables; (2) Options 1 and 2 require two tables — the first is a CD table; the second, a CL table; (3) Options 3 and 4 are table format 1 for three dependent variables; (4) Option 5 uses table format 3 for two independent and five dependent variables; (5) CD_o in option 5 is CD_{min}; (6) All angles and slopes are in degrees.

NC Index	Name	Description
61	NC(61)	Index of dependent variable for 1st plot
62	NC(62)	Index of independent variable for 1st plot
63	NC(63)	Index of dependent variable for 2nd plot
64	NC(64)	Index of independent variable for 2nd plot
⋮	⋮	
73	NC(73)	Index of dependent variable for 7th plot
74	NC(74)	Index of independent variable for 7th plot
		Note: See figures 12 and 13 for plot index
75	NC(75)	Sonic boom placard option (Table 30 must be input) = 0 Planar sonic boom = N Area sonic boom N is number of points to be used along the shock to determine the placard violation. N must be odd and greater than one. (Recommended value is 7.)
76	NC(76)	Not used
77	NC(77)	Pitch angle option = 0 Do not use pitch angle = 1 Use pitch angle
78	NC(78)	Bank angle option = 0 Do not use bank angle = 1 Use bank angle
79	NC(79)	Engine throttling option = 0 Do not use throttling = 1 Use throttling
80	NC(80)	Wing sweep option = 0 Do not use wing sweep = 1 Use sweep
81	NC(81)	AK (1) print option = 0 Do not print AK (1) = 1 Print AK (1)
82	NC(82)	AK (2) print option = 0 Do not print AK (2) = 1 Print AK (2)
⋮	⋮	

VARIABLE TYPE	VARIABLE INDEX	NC INDEX	PLOT INDEX	FORTRAN NAME	SYMBOL	DESCRIPTION	UNITS
AUXILIARY PRINTOUT	1	81	41	AK(1)	T	THRUST	LB
	2	82	42	AK(2)	L	LIFT	LB
	3	83	43	AK(3)	D	DRAG	LB
	4	84	44	AK(4)	Q	DYNAMIC PRESSURE	PSF
	5	85	45	AK(5)	M	MACH NUMBER	—
	6	86	46	AK(6)	GCR	GREAT CIRCLE RANGE	N MI
	7	87	47	AK(7)	WDOT	WEIGHT FLOW	LB/SEC
	8	88	48	AK(8)	VI	INERTIAL VELOCITY	FPS
	9	89	49	AK(9)	γI	INERTIAL FLIGHT PATH ANGLE	DEG
	10	90	50	AK(10)	ψI	INERTIAL HEADING ANGLE	DEG
	11	91	51	AK(11)	α	ANGLE OF ATTACK	DEG.
	12	92	52	AK(12)	TEMP	TOTAL TEMPERATURE	°R
	13	93	53	AK(13)	VA	SPEED OF SOUND	FPS
	14	94	54	AK(14)	P	AMBIENT PRESSURE	PSF
	15	95	55	AK(15)	ρ	AMBIENT DENSITY	SLUG/FT^2
	16	96	56	AK(16)	TEM	AMBIENT TEMPERATURE	°R
	17	97	57	AK(17)	WM	WEIGHT (METRIC)	NEWTONS
	18	98	58	AK(18)	HM	ALTITUDE (METRIC)	METERS
	19	99	59	AK(19)	RPA	RESULTANT PHYSIOLOGICAL ACCELERATION	FT/SEC^2
	20	100	60	AK(20)	N	NORMAL LOAD FACTOR	—
	21	101	61	AK(21)	Qα	DYNAMIC PRESSURE TIMES ANGLE OF ATTACK	PSF-DEG
	22	102	62	AK(22)	G	LOCAL GRAVITY	FT/SEC^2
	23	103	63	AK(23)	R	RADIUS VECTOR	FT
	24	104	64	AK(24)	ΔP	SONIC BOOM OVERPRESURE	PSF
	25	105	65	AK(25)	CL	LIFT COEFFICIENT	PSF
	26	106	66	AK(26)	CD	DRAG COEFFICIENT	
	27	107	67	AK(27)	L/D	LIFT/DRAG RATIO	
	28	108	68	AK(28)	SFC	SPECIFIC FUEL CONSUMPTION	1/SEC
	29	109	69	AK(29)	VM	RELATIVE VELOCITY (METRIC)	M/SEC
	30	110	70	AK(30)	TM	THRUST (METRIC)	NEWTONS
	31	111	71	AK(31)	LM	LIFT (METRIC)	NEWTONS
	32	112	72	AK(32)	DM	DRAG (METRIC)	NEWTONS
	33	113	73	AK(33)	QM	DYNAMIC PRESSURE (METRIC)	NWTS/M^2
	34	114	74	AK(34)	WDOTM	WEIGHT FLOW (METRIC)	KGF/SEC
	35	115	75	AK(35)	VIM	INERTIAL VELOCITY (METRIC)	M/SEC
	36	116	76	AK(36)	QαM	DYNAMIC PRESSURE TIMES ANGLE OF ATTACK (METRIC)	NWT-RAD/M^2
	37	117	77	AK(37)	ΔPM	SONIC BOOM OVER PRESSURE (METRIC)	NWTS/M^2
	38	118	78	AK(38)	SFCM	SPECIFIC FUEL CONSUMPTION (METRIC)	KFG/NWT-SEC
	39	119	79	AK(39)		NOT DEFINED	
	40	120	80	AK(40)			
CONTROL	1	77	81	U(1)	θ	PITCH ANGLE	DEG
	2	78	82	U(2)	ϕ	BANK ANGLE	DEG
	3	79	83	U(3)	η	ENGINE THROTTLE	—
	4	80	84	U(4)	Λ	WING SWEEP ANGLE	DEG
INDEPENDENT	-	-	85	T	(t)	TIME	SEC

FIGURE 13 AUXILIARY, CONTROL AND INDEPENDENT VARIABLES

NC Index	Name	Description
120	NC(120)	AK (40) print option = 0 Do not print AK (40) = 1 Print AK (40)
121	NC(121)	X (1) state variable option = 0 Do not integrate X (1) = 1 Integrate X (1)
122	NC(122)	X (2) state variable option = 0 Do not integrate X (2) = 1 Integrate X (2)
⋮	⋮	
140	NC(140)	X (20) state variable option = 0 Do not integrate X (20) = 1 Integrate X (20)
141	NC(141)	X (21) state variable option = -1 Observe minimum placard = 0 Ignore placard = 1 Observe maximum placard = 2 Observe both maximum and minimum placards
142	NC(142)	X (22) state variable option = -1 Observe minimum placard = 0 Ignore placard = 1 Observe maximum placard = 2 Observe both maximum and minimum placards
⋮	⋮	
160	NC(160)	X (40) state variable option = -1 Observe minimum placard = 0 Ignore placard = 1 Observe maximum placard = 2 Observe both maximum and minimum placards

Note: (1) Figures 12 and 13 list state variables and AK's; (2) X(2), X(3), and X(4) must always be selected; (3) Two tables are reserved for each placard in the order selected, beginning with table 3; (4) Two tables must be input for a placard only if table format 2 is required and both max and min are to be observed. Otherwise, one table is required; (5) For table format 2, the odd numbered table is reserved for the min placard, the even numbered table for the max placard; (6) For table format 1, both limits are always input in the odd numbered table with min placard first; (7) A maximum of 15 state variables may be used.

Card Set 4

Initial Condition Data | Format 7F10 | 3-5 Cards

Card 1

Column	Name	Description
1-10	XSTP	Desired value of stopping condition (checked in last stage only)
11-20	GR	Earth gravitational constant times 10^{-15} (Recommended value = 14.081718)
21-30	RO	Earth radius (feet) (Recommended value = 20,902,992)
31-40	OMEGA	Earth rotational rate (rad/sec) times 10^{4} (Recommended value = 0.72921152)
41-50	GO	Sea level gravity (ft/sec^2) (Recommended value = 32.174)

Card 2

1-10	DUSQ	Initial value of $\int_{t_o}^{T} \delta \bar{u}' W \delta \bar{u} \, dt$ (may be left blank)
11-20	FINNER	Forward trajectory integration error limit for variable step integration (Recommended value = 18 for rockets, 15 for airbreathers)
21-30	BINNER	Backward trajectory integration error limit for variable step integration (Recommended value = 18 for rockets, 15 for airbreathers)
		Note: FINNER and BINNER are in some respect the number of binary bits of accuracy to be held during integration. Eighteen yields four to five significant figures of accuracy. Changing FINNER and BINNER by 3 amounted to changing accuracy by about 1 significant figure. Decreasing FINNER and BINNER is the direction of decreasing accuracy and decreasing computer time
31-40	EPSLN	Maximum allowable tolerance in stopping condition. (Usually EPSLN should be adjusted to give six to eight significant digits for XSTP. For example, if XSTP = 25,000, EPSLN should be about 0.01. If XSTP = 0, EPSLN should be about 0.000001)

Card 2

Column	Name	Description
41-50	MLSP	Maximum last-stage propellant (A value required if NC(6) = 1)
51-60	TSTART	Trajectory starting time (Usual value = 0)
61-70	L	Reference length for sonic boom calculations

Card 3

Column	Name	Description
1-10	XO (1)	Initial value of first state variable selected from figure 12
11-20	XO (2)	Initial value of second state variable selected from figure 12
⋮	⋮	
61-70	XO (7)	Initial value of seventh state variable selected from figure 12
		(Additional cards are required if more than seven state variables are selected.)
		Note: These values are input in ascending variable index order (as selected from figure 12) with no blank fields.

Card Set 5

Stage-Dependent Parameters Format 7F10 8-16 Cards

Each card subset contains the stage-dependent parameter values in fields of 10; i.e., columns 1 to 10, value for stage 1; 11 to 20, value for stage 2, etc. Two cards are required in each subset if MSTAGE > 7.

Card Subset	Name	Parameters
1	WO (NSTAGE)	Initial weight (lb) of each stage Note: (1) A positive value will be used as an initial weight; (2) A zero value will use the end weight of the previous stage at the start weight of the stage; (3) A negative value will use the end weight of the previous stage minus the input value for the stage.
2	TSTP (NSTAGE)	Stopping trajectory time for each stage Note: The last stage does not stop on time (except dummy time if it is a state variable) unless the trajectory exceeds the input stopping time for the last stage, in which case the run will be aborted.

Card Set 5

Card Subset	Name	Parameters
3	SREF (NSTAGE)	Aerodynamic reference area (ft^2) for each stage
4	DELC (NSTAGE)	Nozzle cant angle (deg) for each stage measured from body centerline
5	XKT (NSTAGE)	Total impulse correction factor (XKT) for each stage Note: These values may be used to correct total impulse or each stage since $I_{TOT} = \int_{TSTART}^{T_{BO}} (XKT)\,(T_{VAC})\,dt$ (Usually use XKT's = 1)
6	XKM (NSTAGE)	Weight flow correction factor (XKM) for each stage Note: These values may be used to correct the final weight of each stage since $W_{final} = W_{initial} - \int_{TSTART}^{T_{BO}} \left[(XKM)\,(WDOT) - WDOT_{inerts}\right] dt$ (Usually use XKM's = 1)
7	WDOTI (NSTAGE)	Inert weight flow (lb/sec) for each stage
8	DTMIN (NSTAGE)	Minimum allowable integration step for each stage (Recommended Value = 0.1 second)

Card Set 6

Constraint-Dependent Parameters Format 7F10 0-13 Cards

Card 1		Parameters
1-10	PSI	Desired value for first terminal constraint
11-20	DPSI	Desired final tolerance for first terminal constraint
Card 2		
1-10	PSI	Desired value for second terminal constraint
11-20	DPSI	Desired final tolerance for second terminal constraint
		Note: (1) Enroute placards are treated as terminal constraints and require a desired terminal value of zero; (2) The desired

Card Set 6 (Continued)

value of the constraints and the tolerances must be ordered as selected by the NC(20) - NC(32) controls. For instance, if NC(20) = 5, then the first terminal constraint will be latitude as indicated in figure 12; (3) One card required for each constraint.

Card Set 7

Free Initial Conditions — Format I10, 3F10 — 0-7 Cards

One card is input for each desired variable initial condition as follows:

Column	Name	Description
1-10	IX	Index of state variable with free initial condition (See figure 12) (Right adjusted)
11-20	XOIMAX	Maximum allowable value
21-30	XOIMIN	Minimum allowable value
31-40	DELX	Initial increment per iteration

Note: (1) This card set will be missing if NC(5) = 0; (2) If γ_{TILT} is a free initial condition, then the maximum allowable value must be 200.

Card Set 8

Nominal Guidance Data — Format 2I5, 6F10 — 0-12 Cards

Column	Name	Description
1-5	MODE	Guidance mode index
6-10	NSW	Index of variable used to switch guidance modes (see figures 12 and 13, plot index). NSW is input positive if SWVAL is to be approached from below, or negative if it is to be approached from above
11-20	SWVAL	Value of the variable on which guidance mode is switched
21-30	AMIN	Minimum allowable angle of attack
31-40	AMAX	Maximum allowable angle of attack
41-50	TMIN	Minimum allowable throttling
51-60	TMAX	Maximum allowable throttling

Note: (1) The guidance modes available (MODE) are as follows:

Card Set 8 (Continued)

Guidance Mode Options

Mode	Descriptions	Table Format
0	θ = f(t), open loop	1
1	α = 0, calculate θ	
2	α = f(t), calculate θ	1
3	γ = f(t), calculate θ	1
4	$\dot{\gamma}$ = f(t), calculate θ	1
5	h = f(V), calculate θ	1
6	h = f(M), calculate θ	1
7	$\dot{\gamma}$ = 0, calculate θ	
8	γ = 0, calculate θ	
9	$\dot{V}$ = 0, calculate θ	
10	$\dot{M}$ = 0, calculate θ	
11	$\dot{V}$ = 0, calculate η	
12	$\dot{M}$ = 0, calculate η	

(2) The maximum number of phases is limited to 12, of which only five may be phases requiring tables; (3) Modes 0 and 2 use the input control variable table (table 0) and therefore are not included in the five reserved guidance tables (Note 2); (4) TMIN and TMAX are input only for MODES 11 and 12; (5) If SWVAL occurs before XSTP on the last phase, the remainder of the trajectory will be flown open loop using table 0; (6) The trajectory will always stop on XSTP; (7) The guidance mode option is for the nominal trajectory only. If NARBY = 2, STOP will iterate from the nominal in the usual open loop mode; (8) This card set will be missing if NC(13) = 0; (9) Aerodynamic option 3 or 5 must be used when modes 3 through 10 are used; (10) For modes 5 and 6, MODE is input as a positive number for ascent or negative for descent. In either case, the H-M or H-V table must be monotonically increasing in the independent variable.

Card Set 9

Tables — Format 2I5, A60/ (7F10) — 2-58 Tables

Every table must begin with a title card as follows:

Column	Description
1-5	Table number
6-10	Number of data cards in the table (do not count this card)
11-70	Any desired information

Card Set 9 (Continued)

Input Data Tables

Table Number	Description
0	Control function table (U) Table format 1, NU dependent variables vs time where NU is the number of control variables selected. This table must always be present as the first table and will not be extrapolated. Note: Restart table follows table 0 if NC(17) = 1 and is also assigned as table 0.
1	Weighting function table (W^{-1} diagonal elements). Table format 1, NU, dependent variables vs time. This table must always be present as the second table and allows the user to aid in assigning the value of DUSQ along the trajectory and among the control variables. A zero value will cause no change in a U, and a 1. will allow the full change determined by the gradient. As an example, it is a good idea to restrict pitch angle changes during a tilt maneuver and during the low-speed portion of a VTO trajectory. For instance, this table might be as follows for a VTO liquid rocket tilting at 8 seconds after launch (θ control): 0. 0. 15. 0. 30. 1. 10000. 1.
2	Gamma tilt table Table format 1, 1 dependent variable vs time (Used only if NC(14) = 1)
3	First table for 1st enroute placard (Format specified by card set 3 and figure 12)
4 ⋮	Second table for 1st enroute placard
21	First table for 10th enroute placard
22	Second table for 10th enroute placard
23	Skin friction drag table (required if NC(15) = 1) H_{REF}, $\frac{\Delta CD}{H\text{-}H_{REF}} = F(M)$ Table format 1, 2 dependent variables vs Mach number
24	First guidance table (Format specified by card set 8)
25 ⋮	Second guidance table

Table Number	Description
28	Last guidance table
29	Lateral cutoff distance (required if NC(75) > 1), YCO = F(M, H) Table Format 2
30	Sonic boom table (required if NC(104) or NC(135) = 1

$\Delta PJ = F(H, M, DENOM)$ Table Format 2

where:

$$\Delta PJ = (\Delta P)\left(\frac{H}{L}\right)^{3/4} / K_R \beta^{1/4} \; P_{ref} \quad .$$

ΔP = Overpressure ~ psf

K_R = Sonic boom constant ≈ 1.9

$$P_{ref} = \sqrt{P_o * P}$$

$$\beta = \sqrt{M^2 - 1}$$

$$DENOM = \frac{\beta}{2} C_L \frac{S}{L^2}$$

31	1st engine table for stage 1
	Note: Engine options 10 and 20 require only one table
32	2nd engine table for stage 1
33	1st aerodynamics table for stage 1
	Note: Aero options 3, 4, and 5 require only one table
34	2nd aerodynamics table for stage 1
31	1st engine table for stage 2
32	2nd engine table for stage 2
33	1st aerodynamics table for stage 2
34	2nd aerodynamics table for stage 2

(Repeat tables 31 through 34 for each stage)

Note: (1) Some or all of tables 2 through 34 may be missing, depending on the options selected from card set 3; (2) The first-stage dependent table (31-34) of each stage must be preceded by a card with the word STG in columns 1-3 (remaining columns must be blank); (3) The last-stage dependent table for the last stage must be followed by a card with the word END in columns 1-3 (remaining columns must be blank); (4) The STG cards are required even for stages with no tables; (5) All tables except table 0 may be extrapolated; (6) The total number of data cards for tables 1 through 30 + 31 through 34 for any stage is limited to 1000.

Card Set 9 (Continued)

Three formats are available for data table input. The table lookup routines use linear interpolation and will extrapolate, although the extrapolation feature should be avoided if possible.

Table Format 1 — 1 Independent by 1, 2, 3, 4, or 5 Dependent

X = independent variable

Y_i = dependent variables

NDV = number of dependent variables,

X, Y_1, Y_2, --- Y_{NDV}, X, Y_1, Y_2, --- Y_{NDV}, X, ---

7 entries per card

Figure 14 shows an example for two dependent variables

Table Format 2 — 1, 2, or 3 Independent by 1 Dependent

Cardset 1

Column	Name	Description
1-10	NPL	Number of planes (may = 1)
11-20	ENG	Number of engines (used only for thrust options 3-9)

Cardset 2 (All data for first plane)

Card Subset 1

Column	Name	Description
1-10	X1	Value of the first plane (X1 = 0, if only one plane)
11-20	NCV	Number of curves on first plane (may = 1)

Card Subset 2 (All data for first curve)

Card 1

Column	Name	Description
1-10	X2	Value of the first curve (X2 = 0, if only one curve)
11-20	NPT	Number of data points on first curve (must be > 1)
21-30	X3	Value of the 1st point
31-40	Y	Value of the dependent variable
41-50	X3	Value of the 2nd point
51-60	Y	Value of the dependent variable
61-70	X3	Value of the 3rd point

Card Set 9 (Continued)

Card Subset 2

Card 2

1-10	Y	Value of the dependent variable
11-20	X3	Value of the 4th point
!	!	!
!	!	

7 values per card until all points for the first curve are input

Card Subset 3 (All data for the 2nd curve)

Same format as card subset 2.

Card subsets continue until all curves on the first plane are input.

The next plane is input in the same manner, beginning with card set 2.

Note: (1) Each plane begins with a new card; (2) Each curve begins with a new card; (3) All data in the table is input floating point (Every value must have a decimal point); (4) The table lookup routine uses linear interpolation and will extrapolate linearly in all directions.

Figure 15 shows an example for Table Format 2.

Table Format 3 2 Independent by 5 Dependent

This format is used only for the aerodynamics option 5 as follows:

Cardset 1

Column	Name	Description
1-10	NSWP	Number of constant sweep curves (may be 1)

Cardset 2 (All data for first sweep)

Card Subset 1

Column	Name	Description
1-10	S1	Value of first sweep curve
11-20	NM1	Number of Mach number points on first curve

Card Subset 2

Card 1

Column	Name	Description
1-10	M1	Value of first Mach number
11-20	CDO	

Cardset 9 (Continued)

Card Subset 2

Card 1

Column	Name	Description
21-30	$\partial CD/\partial CL^2$	
31-40	CL_{min}	
41-50	$\partial CL/\partial \alpha$	
51-60	CL_0	
61-70	M2	Value of second Mach number

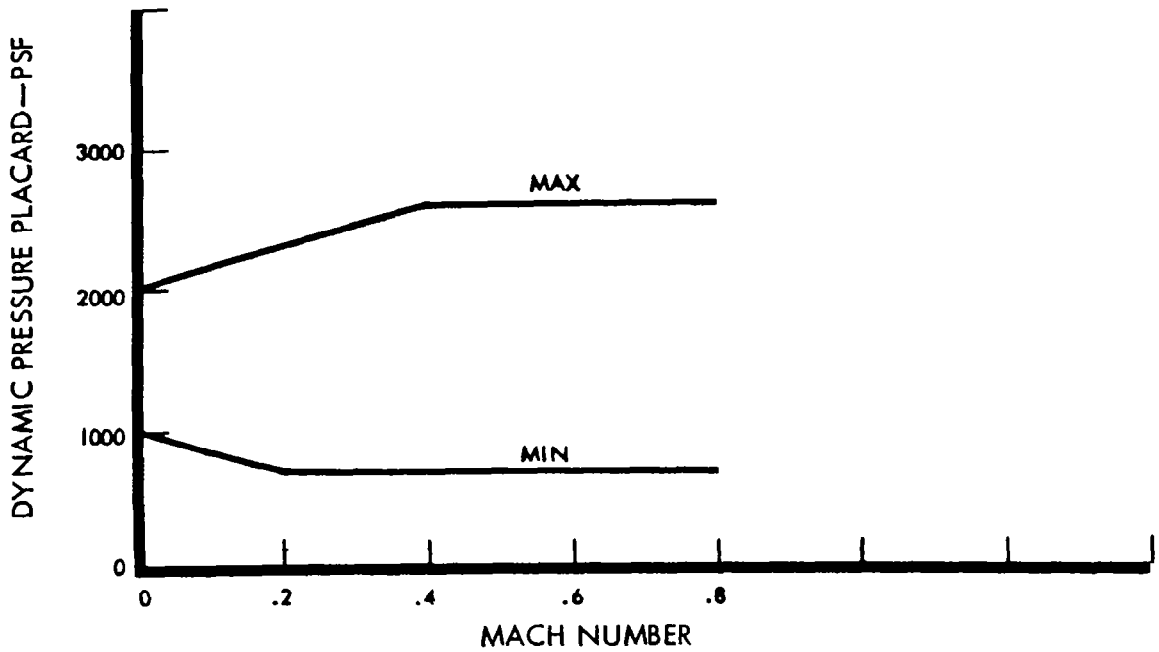
Card 2

Column	Name	Description
1-10	CDO	
11-20	$\partial CD/\partial CL^2$	

(7 values per card until all data for first sweep or input)

Repeat cardset 2 for each sweep angle.

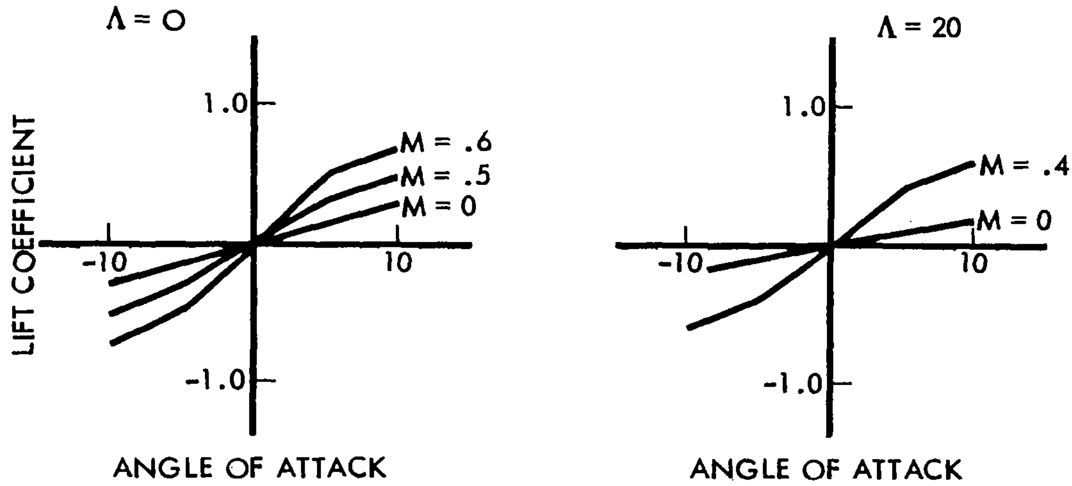
Figure 16 shows an example for Table Format 3.



1 2	3 4	5 6	7 8	9 10	11 12	13 14	15 16	17 18	19 20	21 22	23 24	25 26	27 28	29 30	31 32	33 34	35 36	37 38	39 40	41 42	43 44	45 46	47 48	49 50	51 52	53 54	55 56	57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72	73 74	75 76	77 78	79 80
		3		2		Q	PLACARD TABLE (EXAMPLE)																																
0.					10	00	.			20	00	.			.2					70	0	.			23	00	.			.4								
700.					26	00	.			50.					70	0	.			26	00	.																

NOTE: ADDITIONAL POINTS INCLUDED TO PREVENT TABLE LOCKUP EXTRAPOLATION

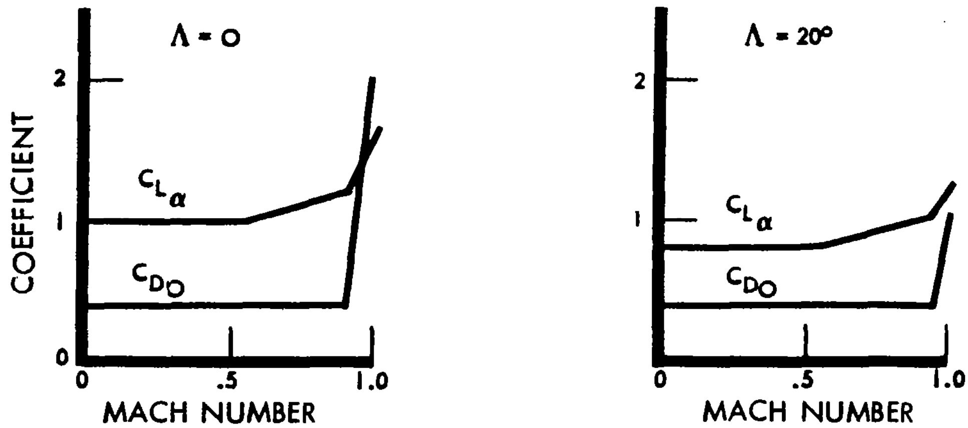
Figure 14 EXAMPLE OF TABLE FORMAT 1



1–10	11–20	21–30	31–40	41–50	51–60	61–70	71–80
34 18	CL TABLE (EXAMPLE)						
3.							
0.	4.						
0.	4.	-50.	-.3	-10.	-.3	10.	
.3	50.	.3					
.5	6.	-50.	-.5	-10.	-.5	-5.	
-.3	5.	.3	10.	.5	50.	.5	
.6	6.	-50.	-.7	-10.	-.7	-5.	
-.5	5.	.5	10.	.7	50.	.7	
10.	2.	-50.	0.	50.	0.		
20.	3.						
0.	4.	-50.	-.2	-10.	-.2	10.	
1–10	11–20	21–30	31–40	41–50	51–60	61–70	71–80
.2	50.	.2					
.4	6.	-50.	-.6	-10.	-.6	-5.	
-.4	5.	.4	10.	.6	50.	.6	
10.	2.	-50.	0.	50.	0.		
60.	2.						
0.	2.	-50.	0.	50.	0.		
50.	2.	-50.	0.	50.	0.		

NOTE: ADDITIONAL POINTS INCLUDED TO PREVENT TABLE LOOKUP EXTRAPOLATION

Figure 15 EXAMPLE OF TABLE FORMAT 2



1–10	11–20	21–30	31–40	41–50	51–60	61–70	71–80
33 11	AERODYNAMICS TABLE (EXAMPLE)						
2.							
0.	4.						
0.	.4	0.	0.	1.	0.	.5	
.4	0.	0.	1.	0.	.9	.4	
0.	0.	1.2	0.	1.	2.	0.	
0.	1.6	0.					
20.	4.						
0.	.4	0.	0.	.8	0.	.55	
.4	0.	0.	.8	0.	.95	.4	
0.	0.	1.	0.	1.	1.	0.	
0.	1.2	0.					

Figure 16 EXAMPLE OF TABLE FORMAT 3

Nomenclature — FORTRAN COMMON. — The table that follows is a list of the FORTRAN names, mathematical symbols, units, descriptions, and subroutines where computed for all items in COMMON.

FORTRAN Name	Symbol	Units	Description	Subroutine Computed
AHS		sec	Maximum step size that the RKVS integration package (STEP1 and STEP2) will take for its next integration step. (This may be decreased, but never increased for the step to be taken)	STEP1 STEP2
AK(40)			Auxiliary printout variables (see appendix B)	AKSTP STP1
AK11	α	deg	Angle of attack	AKSTP
AMAG(15)			Array used by RKVS for the integration of the forward trajectory	INITAL
AP(40)			A packed array containing only those auxiliary variables (AK's) to be printed at each time point on forward trajectories.	STP1
BINNER			Input backward trajectory integration error limit for Runge-Kutta variable step integration	INITAL
CAK11	Cos α		Cosine of the angle of attack	AKSTP
CD	C_D		Drag coefficient	TLD
CI1	Cos θ		Cosine of the pitch angle (Set to 1. if NC(77) = 0)	INITAL CØNTR
CI2	Cos φ		Cosine of the bank angle (Set to 1. if NC(78) = 0)	INITAL CØNTR
CK3	Cos γ_R		Cosine of the flight path angle	AKSTP
CK5	Cos β		Cosine of the latitude (Set to 1. if NC(125) = 0)	INITAL AKSTP
CK6	Cos ψ_R		Cosine of the heading angle (Set to 1. if NC(126) = 0)	INITAL AKSTP
CL	C_L		Lift coefficient	TLD
CØEFK			Step-size coefficient	KCALC
CØF(2, 5)			An array containing aerody-coefficients (see Appendix B)	TLD INITCØ
CØN(28)			Array of constraint(s) and performance name(s)	INITAL

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
CØNST(30)			Array of program constants (see appendix B)	INITAL BLØCK
CPSI(13)			Nondimensional constrain change(s) asked for	INITAL VALID
DEG			Radians to degrees conversion factor	BLØCK
DELC(14)	δ_c	deg	Input nozzle cant angle for each stage measured from body centerline. Assumes zero thrust component normal to body centerline.	INITAL
DELU(4)	$\delta\bar{u}$		Increment added to $\bar{u}$ vector for next trajectory	UCALC
DELX(7)	$\delta\bar{x}_o$		Array of incremental changes in free initial conditions	INITAL VARIC
DELXI(7)			Input maximum allowable increment by which a free initial condition can change per iteration.	INITAL
DELXS(7)			Array of incremental changes in variable initial conditions in order of selected free initial conditions	INITAL VARIC
DENØM			Denominator of first Lagrange multiplier	MATRIX
DESXUK(170)			An array of alphanumeric characters containing all of the state, auxiliary (AK), and control variable names that can be selected for printout via the NC array. Each name is defined by two sequential words in this array.	BLØCK
DESXUP(120)			A packed array of alphanumeric characters containing the printout headings for a given computer run.	TITLES
DPHI			Change in performance (with respect to last accepted iteration)	MATRIX
DPHID			Predicted change in performance due to DUSQ	MATRIX
DPHIP			Total predicted change in performance	MATRIX

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
DPHI3			Predicted changed in performance due to variable initial condition(s).	MATRIX
DPSIM(13)			Input desired final tolerance(s) for terminal constraints.	INITAL
DP2BAR			Value of DUSQ at the start of the first trial of any iteration.	MATRIX
DP2HI			Smallest value of DUSQ attained on any trial of an iteration that is greater than the current DUSQ.	MATRIX
DP2LØ			Largest value of DUSQ attained on any trial of an iteration that is less than the current DUSQ.	MATRIX
DP2NM1			Value of DUSQ used on previous trial or valid step.	MATRIX
DP2TMY			Equal to $\sqrt{DP2BAR*DUSQ}$ and set only when too many trials for an iteration have been run.	MATRIX
DRHØ			Constant used by RKVS integration package	BLØCK
DTMIN(14)		sec	Input minimum allowable integration step for each stage	INITAL
DUSQ			Defined by equation 100	INITAL VALID MATRIX
DXNUM			DUSQ required for DBETA	MATRIX
EPSLN			Input maximum allowable tolerance in stopping conditions	INITAL
ERR(15)			Array of predictor-corrector errors	STEP1
F(15)			Array of derivatives for the equations of motion (see appendix B)	FPRØG PLAC BØØM
FEFEI	$I_{\varphi\varphi}$		Value of $I_{\varphi\varphi}$	MATRIX
FINNER			Input forward trajectory integration error limit for Runge-Kutta variable step integration	INITAL

FORTRAN Name	Symbol	Units	Description	Subroutines Computed	
FSAVE(15)			Values of F array from previous integration step	EXEC	
FSIGN			Stopping condition indicator	EXEC	
FXTRA1			Constraint change factor. That part of the constraint error requested for the next iteration.	INITAL	VALID
FXTRA2	DU^2min		Minimum allowable DU^2	MATRIX	
FXTRA3			Not used		
FXTRA4			Saved value of time when DVAL2 was last called	LAMBDA	DVAL2
GØ	G_o	ft/sec^2	Input mass to weight conversion factor	INITAL	
GR	μ		Input earth gravitational constant times 10^{-15}	INITAL	
GUIDN(10,3)			Array of guidance phase names plus words used in printout of stopping condition and optimization parameter	BLØCK	
HITIM			The last, or highest, time stored in the TIMEU array	INITAL EXEC UCALC	
HMIN			Current minimum step size for a given stage	EXEC	
IDIAG			RKVS diagnostic print indicator	BLØCK	
IDØNE			RKVS step completed indicator	STEP1 STEP 2	
IERRØR			RKVS error message indicator	BLØCK	
IN			RKVS control parameter (see ref. 8)	STEP1 STEP2 GUIDE	EXEC LAMBDA PARTIAL

FORTRAN Name	Symbol	Units	Description	Subroutines Computed	
INCØR			Number of meaningful data words in the TIMEU array at a given time	INITAL UCALC	EXEC
INDBNL			DUSQ "bounce test" indicator	MATRIX	
INDSIC(13)			Number of consecutive accepted iterations since a constraint has been inside its temporary tolerance band.	VALID	
INDTMY			Too many trials indicator	INITAL MATRIX	
INDZER			End point search mode indicator	MATRIX	
INPØ			Printout counter for forward and backward trajectory	EXEC STP1	LAMBDA STP2
INST			Storage counter for forward and backward trajectory	EXEC STP1	LAMBDA STP2
INTØT			Counter for the total of words read into the TIMEU array	EXEC	UCALC
IPF(10)			Packed array containing state variable indices of the placards	TITLES	
IPRNT1			Input variable used to control printout of forward trajectory	INITAL	
IPRNT2			Input variable used to control printout of influence coefficients occurs during backward integration if NC(7) = 1	INITAL	
ISTØR1			Input variable used to control storage of time and partials along forward trajectory	INITAL	
ISTØR2			Input variable used to control storage of adjoints along backward trajectory	INITAL	
ITC			Current iteration counter	INITAL MATIX	
ITR8			Number of iterations that have been plotted	INITAL EXEC	

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
ITWØ27			Constant used by RKVS integration package. Computer word size dependent	BLØCK
IX(7)			Input array of indices for the free initial condition desired	INITAL
IXKU(60)			Packed array of the plot indices of the state variables, AK's, and control variables selected	TITLES
IXTRA1			Thrust option indicator	INITCØ, TLD
IXTRA2			Fuel flow option indicator	INITCØ, TLD
IXTRA3			Aerodynamic option indicator	INITCØ, TLD
IXTRA4			Not used	
KDAT			Logical tape number assigned to the data tape. KDAT contains all input tables except table 0. KDAT is written in INITAL and LAMBDA and read in EXEC and CARDS	BLØCK
KINP			Logical tape number assigned to the input tape, KINP, is read in INITAL	BLØCK
KLAM			Logical tape number assigned to the adjoint tape. KLAM is written in STP2 and contains T, UULAM array, and U array of each point stored during adjoint integration. KLAM is read in UCALC to build new control history.	BLØCK
KØUT			Logical tape number assigned to the output tape. KØUT is written in INITAL, CARDS, STP1, STP2, STEP1, STEP2, MATØUT, VALID, MATRIX, KCALC, VARIC, PRTIAL, AKSTP	BLØCK
KPAR			Logical tape number assigned to the partial tape. KPAR contains NETA records of T, PFX, PFU, and U. KPAR is written in STP1 and read in DVAL2. Also, subroutine UCALC uses KPAR for scratch storage.	BLØCK

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
KPLT			Logical tape number assigned to the plot tape. KPLT is written in STP1 and contains the data to be plotted. KPLT is read by PLOTZ	BLØCK
KPUN			Logical tape number assigned to the punch tape. KPUN is written in CARDS	BLØCK
KSCR			Logical tape number assigned to the scratch tape. KSCR is written in STP1 and contains the control variable history of every accepted valid trajectory.	BLØCK
KTAN			Logical tape number assigned to the control history overlay tape. KTAN is written in INITAL and UCALC and contains the part of the control history that would not fit in the TIMEU array. KTAN is read by EXEC.	BLØCK
KTC			Index of weight in the IX array only if weight is a free initial condition.	INITAL
L(14)			Array of program control indicators (see appendix B)	
LETA			Number of stored integration points generated during a backward trajectory	STP2 LAMBDA
LINES			Printout line counter during the forward and backward trajectory. Subroutines STP1 and STP2 will page when LINES exceeds 57.	EXEC LAMBDA STP1 STP2
LINET1			Number of lines (including blank lines) printed at each printed time during the forward trajectory	INITAL
LINET2			Number of lines (including blank lines) printed at each printed time during the backward trajectory	INITAL
LV			Total number of equations being integrated during backward trajectory	INITAL

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
MAL			Current closed-loop mode number	INITAL GUIDE
MITC			Largest value of J in the PFX(I,J) array for a given problem	INITAL
MNDIAG			Desired maximum number of diagnostics printed from RKVS per (computer) stage	BLØCK
MØDZ			RKVS parameter that determines the method of integration	EXEC LAMBDA
MSTAGE			Input total number of (computer) stages	INITAL
NAM(2)			Two alphanumeric words (STG and END) used as indicators during reading of data. First word determines when stage-dependent data is to be read. Second word determines when all the data for a case has been read.	BLØCK
NARBY			Input degree of problem solution indicator	INITAL
NC(160)			See input description	INITAL
ND(112)			Start and end location of all input stage dependent tables in Z array	INITAL
NDS(34)			Starting locations of all input tables in Z array for a given stage	INITAL EXEC
NEØM			Number of equations of motion being integrated, excluding enroute constraints	TITLES
NERR(15)			Array used by RKVS for error control during integration of forward trajectory	INITAL
NERRS1			RKVS error message control	BLØCK
NERRS2			RKVS error message control	BLØCK
NERRS3			RKVS error message control	BLØCK
NETA			Number of stored integration points generated during a valid forward trajectory	EXEC STP1

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
NETAP(6)			Number of dated points saved for each trajectory to be plotted	EXEC
NETLST			Number of integration steps taken for the previous valid forward trajectory	LAMBDA
NEZ			Location in Z array that contains last data point of last nonstage dependent table	INITAL
NITC			Input maximum number of iterations desired	INITAL
NK			Number of selected auxiliary print-out variables (AK's)	TITLES
NØUP			Total number of points in the control table at any time	INITAL UCALC
NP(40)			Packed array of auxiliary printout variable indices	TITLES
NPF			Number of penalty functions to be integrated	TITLES
NPHASE			Current closed-loop phase number	INITAL
NPLTZ			Number of plots to be made for a given data case	TITLES
NPP			Index of the state variable to be used for the payoff variable	INITAL
NRIC			Number of sets of partial derivatives that will fit in allowable storage during backward integration at any one time.	LAMBDA
NSE			Abort error indicator	MATRIX INITAL AKSTP EXEC STP1 STP2
NST			Index of state variable to be used as stopping condition	INITAL
NSTAGE			Current (computer) stage	INITAL
NTB			Closed-loop guidance table counter. (Initialized to 23 and incremented each time subroutine GUIDE requires a new guidance table)	INITAL GUIDE

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
NTRY			Current trial number (set to 1 at the start of every iteration)	INITAL MATRIX VALID
NU			Number of control variables selected	TITLES
NUMNC			Total number of input nonzero plot indicating NC's (NC(61) - NC(74))	TITLES
NUP1			Number of control variables selected plus 1	INITAL
NV			Subset of the equations being integrated in the predictor-corrector mode during the forward trajectory	INITAL
NV4M			Location in the FB array preceding the first adjoint derivative	INITAL
NWPR			Number of words per record written on KPAR tape on subroutine STP1	INITAL
NX			Total number of equations of motion being integrated during the forward trajectory	TITLES
NXUK			The sum of the selected number of state variables, auxiliary printout variables, and control variables plus 1	TITLES
N2UK			Two times NXUK and indicates the total number of words needed to print the trajectory heading	TITLES
N4			Number of constraints plus the performance and stopping condition	TITLES
N4M1			N4-1	TITLES
N4M2			N4-2	TITLES
ØMEGA	ω	rad/sec	Earth rotational rate	INITAL
PFU(15,4)			Array of $\partial \dot{X}_I / \partial U_J$ I = 1, NX, J = 1, NU	ANPRTL

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
PFX(15,7)			Array of $\partial F_I/\partial X_J$ I = 1, NX, J = 1, MITC	ANPRTL
PHIGRN			Largest value in magnitude that performance parameter has attained on any previous accepted iteration	MATRIX VALID
PHIK			Step-size coefficient with respect to performance	KCALC
PHIM1			Performance value for the last accepted iteration	VALID
PHINL			Nonlinearity of performance variable	KCALC
PHINLD			Desired nonlinearity of performance	BLØCK
PLTNAM(28)			Array containing hollerith data used for plot titles Each plot title is defined by four words of this array	TITLES
PMPH			∂ mach/∂altitude	AKSTP
PMPV			∂mach/∂velocity	AKSTP
PQPH			∂ dynamic pressure/∂altitude	AKSTP
PQPM			∂dynamic pressure/∂mach	AKSTP
PQPV			∂dynamic pressure/∂velocity	AKSTP
PSI(13)			Input desired value(s) for terminal constraint(s)	INITAL
PSIM1(13)			Terminal constraint value(s) for previous accepted iteration	VALID
RAD			Degrees to radians conversion factor	BLØCK
RADIC			Absolute value of first Lagrange multiplier	MATRIX
RØ	R_O	ft	Input earth radius	INITAL
S(10,5)			Array used to transmit placard value and derivatives of the placard input table (see subroutine PLAC)	PLAC

FORTRAN Name	Symbol	Units	Description	Subroutines Computed	
SAK11	Sin α		Sine of the angle of attack	AKSTP	
SI1	Sin θ		Sine of the pitch angle (set to zero if NC(77) = 0)	CØNTR	
SI2	Sin φ		Sine of the bank angle (set to zero if NC(78) = 0)	CØNTR	
SK3	Sin γ_R		Sine of the flight path angle	AKSTP	
SK5	Sin β		Sine of the latitude (set to zero if NC(125) = 0)	AKSTP	
SK6	Sin ψ_R		Sine of the heading angle (set to zero if NC(126) = 0)	AKSTP	
SREF(14)	S	ft^2	Input aerodynamic reference area for each stage	INITAL	
T	t	sec	Trajectory time measure from zero at launch	EXEC STEP1	LAMBDA STEP2
TI		sec	Total time increment to be integrated for a stage	EXEC	LAMBDA
TIMEU(1000)			The first 1000 words of the input control table. The remainder of the control table (if any) is written on KTAN in record sizes of NUPI words	INITAL UCALC	EXEC
TITL(12)			Alphanumeric case title	INITAL	
TØMAX		sec	Terminal trajectory time — updated every accepted iteration	MATRIX	
TSTP(14)		sec	Input stopping time for each stage measured from time zero at launch	INITAL	
TTØL(13)			Temporary tolerances (one for each constraint)	VALID	
U(4)	u		Array of control variables	CØNTR	GUIDE
UI1	θ	deg	Pitch angle (set to zero if NC(77) = 0)	CØNTR	
UI2	φ	deg	Bank angle (set to zero if NC(78) = 0)	CØNTR	
UI3	η		Throttling parameter (set to 1 if NC(79) = 0)	CØNTR	GUIDE
UI4	Λ	deg	Sweep angle (set to zero if NC(80) = 0)	CØNTR	

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
VAR(50)			Array of variables (see appendix B)	AKSTP TLD INITAL
VIN(4)	w_{ii}		Automatic weighting matrix elements	LAMBDA
VINP(4)			Input weighting matrix elements	DVAL2
WDØT(14)	$\dot{W}_{inert}$	lb/sec	Input inert weight flow for each stage	INITAL
WØ(14)		lb	Input initial weight of each stage	INITAL
X(15)			Array of state variables being integrated	INITCØ STEP1 EXEC INITAL
XKM(14)			Input weight flow correction factor for each stage	INITAL
XKT(14)			Input total impulse correction factor for each stage	INITAL
XK1	W	lb	Weight (set to initial weight if NC(121) = 0)	INITCØ
XNAME(19)			Alphanumeric adjoint headings	BLØCK
XNC9			Floating point value equivalent to NC(9)	INITAL
XNUM			Numerator of first Lagrange multiplier	MATRIX
XØ(15)			Input initial values of state variables	INITAL
XØIMAX(7)			Input maximum allowable value of free initial condition	INITAL
XØIMIN(7)			Input minimum allowable value of free initial condition	INITAL
XRHØ			Constant used by RKVS integration package	BLØCK
XSAVES(15)			Terminal value of state variables from previous valid trajectory	VALID
XSTP			Input desired value of stopping condition	INITAL STP1
XXLAM(15,14)	Λ		Influence coefficients corrected for stopping time variation	DVAL2

FORTRAN Name	Symbol	Units	Description	Subroutines Computed
YNAME(14)			Alphanumeric adjoint headings	BLØCK
Z(7000)			General data storage array for nonstage-dependent tables (per stage) during forward trajectory. Used for storage of partial sets during backward integration (See program equivalencing)	INITAL EXEC CARDS UCALC

Output Description

The program produces three types of output: the printout, punched cards, and magnetic tape.

Printed output. — The printout gives listings of the basic input data, controls for the program, trajectory data for both forward and backward integrations, convergence information, a control variable printout, and a trajectory summary. A partial derivative check is printed out during the nominal forward trajectory if selected.

Preliminary trajectory printout:

Page 1

Program title

Data case title

Initial condition data

Stage dependent parameters

Constraint dependent parameters

Optimization parameter

Stopping parameter

Free initial condition parameters

Guidance phase dependent parameters

Page 2

Control card (see card set 2)

NC array

Page 3

(Number of pages are dependent on number and size of tables)

The input tables are printed in the order they are input

The nonstage dependent tables are printed first, followed by the tables for each stage.

Forward trajectory: The forward trajectory printout follows the tabular output. At the top of each page, the names of each variable selected for printout appear. These headings are printed eight per line, beginning with TIME. The state, auxiliary, and control variable headings are printed next, beginning with the first state variable and ending with the last control variable. Any heading followed by an asterisk (*) indicates metric units for that variable. The printout of the forward trajectory data occurs every IPRNT1 integration steps, with

two blank lines between each printed step. A line counter, assuming 57 lines per page, ensures that each page will begin with the headings described above. For data cases with more than one stage, the beginning of each stage starts a new page.

Partial derivative check printout: The forward trajectory printout format is altered if the numerical partial check option is being used. The partial derivative printout will occur every NC (12) stored integration steps immediately following the forward trajectory printout of the same time point. The format for the partial check follows: The word TIME and the current time are printed on the first line. The partials are printed eight per line with 1PE14.7 format.

The analytical partial derivatives of the first equation of motion with respect to the first MITC state variables are printed first, followed by the equivalent numerical partials. This is repeated for each of NX equations of motion in following rows. The printout of the partial derivatives with respect to the state variables has the form of the array PFX (I, J) where I (the row) represents the equation of motion and J (the column) represents the derivative state variable.

The analytical partial derivatives of the NX equations of motion with respect to the first control variable are printed next, followed by the equivalent numerical partials. This is repeated for NU control variables. The printout of the partial derivatives of the equations of motion with respect to the control variables has the form of the array PFU (I, J) where I (the column) represents the equation of motion and J (the row) represents the derivative control variable.

Backward trajectory printout: The backward trajectory is printed only if NC(7) is nonzero. The backward trajectory begins at the final time of the valid step and integrates to the trajectory starting time, printing the influence coefficients and instantaneous impulse responses every IPRNT2 integration steps. Both the influence coefficients and impulse responses are corrected for variations in the stopping condition. At the top of each page is printed a block of headings identifying each variable. The work TIME and the current time are printed on the first line. The second line consists of the influence coefficients for MITC state variables on performance. Next are printed the influence coefficients for MITC state variables on the first constraint, followed by the influence coefficients for MITC state variables on the second constraint, etc., through all the constraints. Following the influence coefficient printout, the instantaneous impulse responses for the first control variable on performance, on the first constraint, the second constraint, etc., are printed. The next row of print is the impulse response for the second control variable on performance, on the first constraint, the second constraint, etc., through all constraints. This is repeated for all the control variables selected. The backward trajectory printout uses the same line count and paging control as the forward trajectory printout.

Matrix printout (convergence information): The convergence printout gives information used by the stepsize controller in making decisions concerning trial trajectories, valid steps, etc. The printout is given as notes and as arrays of

values used by the program. The information aids the user in determining the nature of convergence. The printout information and notes are given below. The underlined statements and statements preceding a colon are for clarification in describing regions of the printout and are not part of the printout.

Printout from convergence logic after a valid step or trial.

Iteration number

Trial number

Number of integration points stored during forward trajectory integration (NETA)

Number of integration points stored during backward trajectory integration (LETA)

Printout from convergence logic after a valid step only.

Majority vote counter

Messages printed under special conditions are:

"Number of tries plus rejected valid steps for current iteration exceeds 10 — abort"

"Too many functions with adverse travel"

"Adverse phi too great"

"DUSQ modified because of too many tries:

"Reject valid step"

Printout from convergence logic after a reverse integration only.

I matrix

Minimum allowable DUSQ

$I_{\varphi\psi} \times I_{\psi\psi}^{-1} \times I_{\psi\varphi}$

Denominator of first Lagrange multiplier

Diagonal elements of automatic weighting matrix

*Derivatives of performance with respect to variable initial condition (DPDX)

*Elements of DPDX (SUM, XXLAM)

Note: (1) The items with * are printed only when the variable initial condition option is selected, NC(5) $\neq$ 0.

(2) XXLAM is the performance change resulting from a unit step in X_o (first term in the bracket in equation 127) and SUM is the performance change that would result if the control function were modified to bring the constraint vector back to the state prior to the application of X_o (second term in the bracket in equation 127).

Messages printed under special conditions are:

"Convergence failed"

"Converging but gradient too large for extremal"

"The last trajectory appears to be an extremal"

"Gradient of phi too negative to continue"

"Overflow occurred in matrix inversion-abort"

"Singular matrix-abort"

Printout from convergence logic after a trial only.

Majority vote counter

*Maximum permissible changes in constraints

*Controlling function and the step-size coefficient

*Function or functions reducing step-size coefficient and the new value of the step-size coefficient

*Final step-size coefficient

Note: The notes above with * are written only if the majority vote is $\geq$ zero

Messages printed under special conditions are:

"Controlling with constraint within tolerance band which reduces the step-size coefficient — repeat-step-size-coefficient calculation"

"Too many functions with adverse travel"

"Adverse phi too great"

"End point search and step-size coefficient greater than 2 — force valid step"

"Penalty function violation — force valid step"

"Performance improved and step-size coefficient equals 0.5, accept current trial and force valid step"

"All constraints outside tolerance band improved but DUSQ reduced too much, accept current trial and force valid step"

"DUSQ was increasing and wants to decrease accept previous trial and force valid step"
"DUSQ was decreasing and wants to increase accept current trial and force valid step"

Printout from convergence logic after a reverse integration, trial, or a rejected valid step.

Step-size (DUSQ) required for $\Delta\beta$

Numerator of first Lagrange multiplier

Lagrange multipliers

Accuracy check for step-size (DUSQ) (this value should agree with the value of DUSQ for next trajectory)

Heading block of constraint and performance names

Current end point values of constraints and performance

Previous valid step end point values of constraints and performance

Change in constraint and performance end point values

Constraint travel indicator (0 if inside tolerance band, 1 if constraint change is toward desired constraint value, and -1 if constraint change is away from desired constraint value)

Constraint tolerance indicators (number of consecutive iterations a constraint has been outside its tolerance band)

Allowable nondimensional forward constraint change (in the direction of the desired constraint value)

Allowable nondimensional backward constraint change (in the direction away from the desired constraint value)

Nondimensional constraint change asked for (percentage of constraint error to be eliminated by this iteration)

Constraint tolerance bands

Nonlinearities of constraints and performance

Step-size coefficients of constraints and performance

Maximum step-size coefficients of constraints (based on maximum permissible constraint travel)

Predicted changes in constraint and performance end points (total)

Predicted changes in constraint and performance end points due to step-size (DUSQ)

Predicted changes in constraint and performance end points due to changes in variable initial conditions

Changes in variable initial conditions to be made on next trial or valid step (printed only if variable initial condition option is selected, NC(5) $\neq$ 0)

Final step-size coefficient

DUSQ for next trajectory

Messages printed under special conditions:

"DUSQ too small to meet DBETA"

"DUSQ set equal to DUSQ required for DBETA"

"End point search"

"Scale DBETA to match DUSQ"

"DUSQ set to minimum allowable value"

"DUSQ too small for optimization — DUSQ reset"

"6 tries are too many — force valid step"

"8 tries are too many — force valid step"

"Next trajectory will be a trial"

"Next trajectory will be a valid step"

Control variable table and restart table: The control variable table and the restart table are printed following the last iteration. The restart table is not printed following the running of a nominal trajectory only.

Trajectory summary: The trajectory summary follows the printout of the control variable table and restart table if one or more iterations have been run. The heading block that appears at the beginning of the trajectory summary is identical to the one which appears during the printout of the forward trajectory. The format of the trajectory summary is also identical to the forward trajectory format. The summary contains the end points for each successful iteration.

Punched card output. — The program produces punched cards in the form of control variable and restart tables. These tables are used to restart a problem from the point where it was stopped.

Punched output is produced when either or both of the following options are selected:

1) NC (8) > 0 and NARBY = 2

2) NC (8) > 0 and NC (13) > 0

The control table is punched six words per card in octal format (6Ø12) and the restart table is punched in 24I3/6E12.5. Both tables are preceded by title cards in the format required for input to the program. A restart table is not punched following a nominal trajectory.

The restart table consists of previous valid step convergence information, used by the MATRIX subroutine. Included in this table are DUSQ, performance and constraint end points, constraint temporary tolerance bands, and the stopping time of the previous valid step.

Note: If punched cards are not produced due to machine error or program abort, the control variable table can be recovered from the magnetic tape (KSCR).

Restart table output description:

Card 1

Column	Name	Description
1-5		Restart table number (0)
6-10		Number of cards in restart table (title card not included in this count)
12-23		Alphanumeric word RESTART
24-47		Case title as input on title card of data case

Card 2 Format 24I3

Column	Name	Description
1-3	INDTMY	Forced valid step indicator = 0 valid step not forced = 1 valid step forced because too many trials were run = 2 valid step forced because control logic is using the minimum allowable value of DUSQ
4-6	INDSIC(1)	Number of consecutive iterations a constraint has been outside its respective temporary tolerance band. A value is present for each constraint (None of these values appear for a no-constraint problem.)
7-9	INDSIC(2)	
10-12	INDSIC(3)	
⋮	⋮	
40-42	INDSIC(13)	

Card 3 — Format 6E12.5

Column	Name	Description
1-12	DUSQ	Value of DUSQ used for last successful valid step
13-24	FXTRA1	Amount of nondimensional constraint error asked for on the last valid step. (This value will be zero for a no-constraint problem.) This is the maximum amount of constraint error that can be asked for by any one constraint.
25-36	DP2TMY	$(DP2BAR \times DUSQ)^{1/2}$ where DP2BAR is the value of DUSQ used on the second-to-last successful valid step and DUSQ is the current value of DUSQ. Note: DUSQ is set equal to DP2TMY only when the indicator INDTMY has been set equal to 1.
37-48	TØMAX	Trajectory time for last successful valid step.
49-60	PHIM1	Value of the performance function on the second-to-last successful valid step.

(The following values occur only if the problem has one or more constraints.)

61-72	PSIM1(1)	Terminal value of the first constraint on the second-to-last successful valid step.

Card 4 (If needed) — Format 6E12.5

Column	Name	Description
1-12	TTØL(1)	The temporary tolerance band corresponding to the first constraint (expressed as half of the total band width).
13-24	CPSI(1)	The amount of nondimensional constraint error asked for on the last valid step. Note: This value is normally equal to FXTRA1 but under the conditions mentioned in appendix C, the value may be 0.1.
25-36	PSIM1(2)	Values for second constraint
37-48	TTØL(2)	
49-60	CPSI(2)	

Special notes:

1) The value of DUSQ input through the use of a restart table will override the value of DUSQ input in the initial condition data section of the input;

2) The values of the temporary tolerances input through the restart table will override those normally computed by the control logic using the final input constraint tolerances;

3) If it is desired to change the amount of constraint error asked for all that is necessary is to change the value of FXTRA1.

   (FXTRA1 must never be greater than 1.)

   The individual values of CPSI are set equal to FXTRA1 or .1 (under special conditions mentioned above).

   The values of CPSI never need to be reset.

Magnetic tape output. — The program produces a magnetic tape when plotting (KPLT) and another magnetic tape when iterating (KSCR).

Plot tape (KPLT): The tape KPLT is produced when NC(11) $\neq$ 0. This tape contains the data for the variables selected for plotting. For each valid trajectory that is plotted, NETA records are written on KPLT in binary format. The number of points stored when plotting (NETA) must be less than or equal to 1000. (The absolute value of NC(11) defines which iterations will be plotted.) The number of words written per record is two times the number of plots to be made, or the number of nonzero elements in the NC array from NC(61) through NC(74). The plot data is written on tape every ISTOR1 integration steps in the order selected by input.

Control table tape (KSCR): The tape KSCR contains the control variable history for the nominal and each accepted valid trajectory of a data case in binary format. This tape is generated when NC(8) > 0; and NARBY = 2 and/or NC (13) > 0. For each valid trajectory there are NETA records written on tape, where each record consists of TIME and NU control variables. This tape (KSCR) is used to recover the control table for any iteration, in the event a data case is interrupted and doesn't punch cards.

The procedure used to recover a control table is described below. A short FORTRAN IV program called "THETA" is included in the source decks for the STOP program. This program reads the binary tape (KSCR) on logical unit 2 and produces a listing and punched cards (OCTAL FORMAT 6Ø12) of the control variable table selected by input. Input to this program is described below.

CARD NUMBER 1 (315, 4A6)

COLS 1-5, ITC, Number of control tables written on KSCR up to, but not including, the desired table. (The nominal and every accepted valid step will have a control table written on tape.) Equal to ITC printed immediately following desired valid step.

COLS 6-10, NU, Number of control variables used.

COLS 11-15, NTABLE, Number of copies of control table desired.

COLS 21-44, TITLE

Any information desired on the title card of the punched control table.

CARD NUMBER 2 (14I5)

COLS 1-5, NETAS(1)

Number of points stored during nominal

6-10, NETAS(2)

Number of points stored during first valid trajectory.

11-15, NETAS(3)

Number of points stored during second valid trajectory.

16-20, NETAS(4)

Number of points stored during third valid trajectory.

• •
• •
• •
• •

NETAS(ITC)

Number of points stored during valid trajectory preceding the one from which restart is desired.

NETA

Number of points stored during valid trajectory from which restart is desired.

Both the printed and punched tables produced contain one additional point at the end of the data to prevent extrapolation off the end of the table.

Sample Problem

A sample problem is given which demonstrates the data setup and output for a simple, single-stage, air-launched rocket. One iteration is given showing the nominal trajectory, the control logic output, one trial, a valid step, an output control table and restart table, and the trajectory summary.

Statement of problem. — The flight path and control variable history are required to maximize the final weight of the rocket with the following initial conditions.

$$W_o = 1500 \text{ pounds}$$

$$h_o = 500 \text{ feet}$$

$$\gamma_o = 60 \text{ degrees}$$

$$V_o = 600 \text{ fps}$$

The terminal constraints imposed are:

$$V_f = 2000 \text{ fps}$$

$$h_f = 1200 \text{ feet}$$

The final velocity is selected as the stopping condition. The initial weight and flight path angles are free to change to improve performance. The initial conditions W_o and γ_o above are the starting values.

The aerodynamics and propulsion data are given in tables II and III. The reference area is 2 square feet.

TABLE II Aerodynamic Data

M	C_{D_o}	$\frac{\partial C_D}{\partial C_L^2}$	C_{L_α}
0	.2	.5	.075
1.5	.4	.5	.075
10	.4	.5	.075

TABLE III Propulsion Data

Thrust	7000 pounds
Specific impulse	280 seconds
Nozzle exit area	144 square inches

Sample input. — The statement of problem is transformed into data input cards through the use of the STOP input instructions given under "Input Data Preparation." The data input cards are listed below.

```
EXAMPLE ROCKET CHECK CASE
    1    5    1    1    2    1    1
    4    3    5    2    8    1    9    1   13    2   16    0   17
   18    4   19    1   20    2   33   10   47    3   77    1    7
   81    1   82    1   83    1   84    1   85    1   87    1  121    1
  122    1  123    1  124    1   91    1
2000.     14.081718 20902992. 0.          32.174
100.      18.       18.       .001
1500.     500.      60.       600.
1500.
400.
2.
0.
1.
1.
0.
.1
12000.    100.
          1 2000.   1000.     100.
          3 70.     50.       2.
     0    4 1000.
     1   85 400.
     0    1 THETA TABLE
0.        60.       100.      60.
     1    1  WEIGHTING  TABLE
0.        1.        100.      1.
STG
    31    2  ENGINE TABLE
0.        7000.     144.      280.      100.      7000.     144.
280.
    33    2  AERO TABLE
0.        .2        .5        .075      1.5       .4        .5
.075      10.       .4        .5        .075
END
```

Sample output. — The results of the problem for one iteration are given below. A complete description of the printed output is given in the section on output description.

SUPERSONIC TRANSPORT OPTIMIZATION PROGRAM (STOP) THETA CONTROL 7094 IBSYS 13 VERSION
EXAMPLE ROCKET CHECK CASE

INITIAL CONDITION DATA

XSTP = 2.0000000E 03 GR = 1.4081718E 16 RO = 2.0902992E 07 OMEGA = 0.0000000E-39 GO = 3.2174000E 01
DUSQ = 1.0000000E 02 FINNER= 1.8000000E 01 BINNER= 1.8000000E 01 EPSLN = 1.0000000E-03 MLSP =-0.0000000E-39
TSTART=-0.0000000E-39 L =-0.0000000E-39 XO(1)= 1.5000000E 03 XO(2)= 5.0000000E 02 XO(3)= 6.0000000E 01
XO(4)= 6.0000000E 02 XO(

STAGE DEPENDENT PARAMETERS

STAGE	WEIGHT	TSTP	SREF	DELC	XKT	XKM	WDOT	DTMIN
1	1500.00000	400.00000	2.00000	0.00000	1.00000	1.00000	0.00000	0.10000

CONSTRAINT DEPENDENT PARAMETERS

VARIABLE	PSI	DPSIMAX
ALTITUDE	12000.00000	100.00000

MAXIMIZE WEIGHT

STOP ON VELOCITY R

FREE INITIAL CONDITION PARAMETERS

VARIABLE	XMAX	XMIN	DELX
WEIGHT	2000.00000	1000.00000	100.00000
GAMMA R	70.00000	50.00000	2.00000

GUIDANCE PHASE DEPENDENT PARAMETERS

PHASE	MODE	SWITCH VARIABLE	SWITCH VALUE	ALPHA MIN	ALPHA MAX	THROTTLE MIN	THROTTLE MAX
1	ALPHA	INCREASE TO VELOCITY R	1000.00000	-0.00000	-0.00000	-0.00000	-0.00000
2	ALPHA	INCREASE TO TIME	400.00000	-0.00000	-0.00000	-0.00000	-0.00000

INTEGER TABLES

IPRNT1 = 1 IPRNT2 = 5 ISTOR1 = 1 ISTOR2 = 1 NARBY = 2 NITC = 1 MSTAGE = 1

NC CONTROLS

NC(1)= 0	NC(2)= 0	NC(3)= 0	NC(4)= 3	NC(5)= 2	NC(6)= 0	NC(7)=-0	NC(8)= 1	NC(9)= 1	NC(10)= 0
NC(11)= 0	NC(12)= 0	NC(13)= 2	NC(14)= 0	NC(15)= 0	NC(16)= 0	NC(17)=-0	NC(18)= 4	NC(19)= 1	NC(20)= 2
NC(21)= 0	NC(22)= 0	NC(23)= 0	NC(24)= 0	NC(25)= 0	NC(26)= 0	NC(27)= 0	NC(28)= 0	NC(29)= 0	NC(30)= 0
NC(31)= 0	NC(32)= 0	NC(33)=10	NC(34)= 0	NC(35)= 0	NC(36)= 0	NC(37)= 0	NC(38)= 0	NC(39)= 0	NC(40)= 0
NC(41)= 0	NC(42)= 0	NC(43)= 0	NC(44)= 0	NC(45)= 0	NC(46)= 0	NC(47)= 3	NC(48)= 0	NC(49)= 0	NC(50)= 0
NC(51)= 0	NC(52)= 0	NC(53)= 0	NC(54)= 0	NC(55)= 0	NC(56)= 0	NC(57)= 0	NC(58)= 0	NC(59)= 0	NC(60)= 0
NC(61)= 0	NC(62)= 0	NC(63)= 0	NC(64)= 0	NC(65)= 0	NC(66)= 0	NC(67)= 0	NC(68)= 0	NC(69)= 0	NC(70)= 0
NC(71)= 0	NC(72)= 0	NC(73)= 0	NC(74)= 0	NC(75)= 0	NC(76)= 0	NC(77)= 1	NC(78)= 0	NC(79)= 0	NC(80)= 0
NC(81)= 1	NC(82)= 1	NC(83)= 1	NC(84)= 1	NC(85)= 1	NC(86)= 0	NC(87)= 1	NC(88)= 0	NC(89)= 0	NC(90)= 0
NC(91)= 1	NC(92)= 0	NC(93)= 0	NC(94)= 0	NC(95)= 0	NC(96)= 0	NC(97)= 0	NC(98)= 0	NC(99)= 0	NC(100)= 0
NC(101)= 0	NC(102)= 0	NC(103)= 0	NC(104)= 0	NC(105)= 0	NC(106)= 0	NC(107)= 0	NC(108)= 0	NC(109)= 0	NC(110)= 0
NC(111)= 0	NC(112)= 0	NC(113)= 0	NC(114)= 0	NC(115)= 0	NC(116)= 0	NC(117)= 0	NC(118)= 0	NC(119)= 0	NC(120)= 0
NC(121)= 1	NC(122)= 1	NC(123)= 1	NC(124)= 1	NC(125)= 0	NC(126)= 0	NC(127)= 0	NC(128)= 0	NC(129)= 0	NC(130)= 0
NC(131)= 0	NC(132)= 0	NC(133)= 0	NC(134)= 0	NC(135)= 0	NC(136)= 0	NC(137)= 0	NC(138)= 0	NC(139)= 0	NC(140)= 0
NC(141)= 0	NC(142)= 0	NC(143)= 0	NC(144)= 0	NC(145)= 0	NC(146)= 0	NC(147)= 0	NC(148)= 0	NC(149)= 0	NC(150)= 0
NC(151)= 0	NC(152)= 0	NC(153)= 0	NC(154)= 0	NC(155)= 0	NC(156)= 0	NC(157)= 0	NC(158)= 0	NC(159)= 0	NC(160)= 0

NON-STAGE DEPENDENT TABLES

TABLE 0. 1 CARD(S) THETA TABLE

0.000000	60.000000	100.000000	60.000000	-0.000000	-0.000000	-0.000000

TABLE 1. 1 CARD(S) WEIGHTING TABLE

0.000000	1.000000	100.000000	1.000000	-0.000000	-0.000000	-0.000000

STAGE DEPENDENT TABLES

STAGE 1

TABLE 31. 2 CARD(S) ENGINE TABLE

0.000000	7000.000000	144.000000	280.000000	100.000000	7000.000000	144.000000
280.000000	-0.000000	-0.000000	-0.000000	-0.000000	-0.000000	-0.000000

TABLE 33. 2 CARD(S) AERO TABLE

0.000000	0.200000	0.500000	0.075000	1.500000	0.400000	0.500000
0.075000	10.000000	0.400000	0.500000	0.075000	-0.000000	-0.000000

TIME	WEIGHT	ALTITUDE	GAMMA R	VELOCITY R	THRUST	LIFT	DRAG
Q	MACH NUMBER	WEIGHT FLOW	ALPHA	PITCH ANGLE			
-0.000	1500.000	500.000	60.000	600.000	4921.729	0.000	229.249
421.757	0.538	25.000	0.000	60.000			
0.500	1487.500	766.721	59.301	636.569	4941.750	49.412	261.514
471.027	0.572	25.000	0.699	60.000			
1.250	1468.750	1193.149	58.483	692.075	4973.434	125.145	318.301
549.801	0.622	25.000	1.517	60.000			
2.375	1440.625	1893.829	57.668	776.490	5024.634	237.181	418.467
677.911	0.700	25.000	2.332	60.000			
4.062	1398.438	3087.819	57.098	905.035	5109.458	386.900	591.821
888.786	0.819	25.000	2.902	60.000			
5.750	1356.250	4461.089	57.013	1035.342	5203.335	500.092	782.826
1116.151	0.942	25.000	2.987	60.000			
6.250	1343.750	4902.583	56.574	1074.978	5232.698	0.000	785.042
1187.299	0.979	25.000	0.000	56.574			
6.583	1335.417	5204.788	56.262	1101.560	5252.571	0.000	825.168
1235.379	1.005	25.000	0.000	56.262			
7.083	1322.917	5669.862	55.803	1141.597	5282.800	0.000	887.150
1308.197	1.043	25.000	0.000	55.803			
7.833	1304.167	6393.807	55.135	1201.988	5329.009	0.000	983.961
1418.583	1.101	25.000	0.000	55.135			
8.958	1276.042	7538.446	54.174	1293.208	5400.005	0.000	1137.028
1585.342	1.190	25.000	0.000	54.174			
10.646	1233.854	9385.487	52.817	1431.095	5509.388	0.000	1361.182
1833.245	1.325	25.000	0.000	52.817			
13.177	1170.573	12440.834	50.943	1639.415	5676.950	0.000	1747.498
2184.373	1.536	25.000	0.000	50.943			
15.708	1107.292	15829.469	49.234	1854.644	5844.557	0.000	2004.126
2505.158	1.760	25.000	0.000	49.234			
17.345	1066.387	18198.317	48.207	2000.000	5951.141	0.000	2154.912
2693.640	1.915	25.000	0.000	48.207			

ITERATION 0 TRIAL 0 NETA= 26 LETA= 0

I MATRIX

0.33750698E-01 -0.51741087E 01

-0.51741087E 01 0.29956934E 04

MINIMUM ALLOWABLE DUSQ	0.12824789E 03		
I PHI PSI*I PSI PSI INVERSE*I PSI PHI	0.89366292E-02		
DENOMINATOR OF FIRST LAGRANGE MULTIPLIER	0.24814069E-01		
AUTOMATIC WEIGHTING MATRIX ELEMENTS	0.10000000E 01		
VARIC CHECKOUT PRINTOUT - DPDX,SUM,XXLAM	0.72270353E 00	-0.16818235E-01	0.7058853CE 00
VARIC CHECKOUT PRINTOUT - DPDX,SUM,XXLAM	0.26056315E 00	-0.33655244E-01	0.22689791E 00
DUSQ REQUIRED FOR DBETA	0.583961136 03		
NUMERATOR OF FIRST LAGRANGE MULTIPLIER	-0.383961136 03		
DUSQ TOO SMALL TO MEET DBETA			
SCALE DBETA TO MATCH DUSQ			
DUSQ TOO SMALL FOR OPTIMIZATION-DUSQ RESET			
DUSC REQUIRED FOR DBETA	0.200C0C00E 03		
NUMERATOR OF FIRST LAGRANGE MULTIPLIER	0.21000000E 03		
LAGRANGE MULTIPLIERS	C.91994244E 02	-0.99493583E-01	
ACCURACY CHECK FOR DUSQ	0.40999999E 03		

	ALTITUDE	WEIGHT
CURRENT END POINT VALUES	18198.317139	1066.386520
PREVIOUS VALID STEP END POINT VALUES	0.000000	0.000000
CHANGE IN END POINTS	0.000000	1066.386520
CONSTRAINT TRAVEL INDICATOR(S)	23159	
CONSTRAINT CROSS-OVER INDICATOR(S)	1	
ALLOWABLE FORWARD CONSTRAINT CHANGE(N-D)	0.000000	
ALLOWABLE BACKWARD CONSTRAINT CHANGE(N-D)	0.000000	
CONSTRAINT CHANGE ASKED FOR(N-D)	0.05CC00	
CONSTRAINT TOLERANCE(S)	1000.000000	
NON-LINEARITIES	0.000000	0.000000
STEP-SIZE COEFFICIENTS	C.000C00	0.000000
MAXIMUM STEP-SIZE COEFFICIENT(S)	0.000000	
PREDICTED END POINT CHANGE(S) (TOTAL)	-181.370571	45.195420
PREDICTED END POINT CHANGE(S) (DUSQ)	-774.040482	3.619660
PREDICTED END POINT CHANGE(S) (VIC)	592.669907	41.575759
CHANGE IN WEIGHT VIC	58.522521	
CHANGE IN GAMMA R VIC	1.170450	
FINAL STEP SIZE COEFFICIENT	0.000000	
DUSQ FOR NEXT TRAJECTORY	409.999996	
NEXT TRAJECTORY WILL BE A TRIAL		

TIME	WEIGHT	ALTITUDE	GAMMA R	VELOCITY R	THRUST	LIFT	DRAG
Q	MACH NUMBER	WEIGHT FLOW	ALPHA	PITCH ANGLE			
-0.000	1558.523	500.000	61.170	600.000	4921.729	291.923	279.764
421.757	0.538	25.000	4.614	65.785			
18.161	1104.493	18152.601	44.945	2000.000	5949.163	1319.862	2319.637
2697.753	1.915	25.000	3.262	48.207			

ITERATION 1 TRIAL 1 NETA= 0 LETA= 73

MAJORITY VOTE TEST	1	
*** ENTERING KCALC ***		
MAXIMUM PERMISSABLE CHANGE IN ALTITUDE	-1503.091888	
CONTROLLING WITH WEIGHT		
STEP-SIZE COEFFICIENT	1.912763	
FINAL STEP-SIZE COEFFICIENT	1.912763	
*** LEAVING KCALC ***		
DUSQ REQUIRED FOR DBETA	0.99495189E 03	
NUMERATOR OF FIRST LAGRANGE MULTIPLIER	0.50510020E 03	
DUSQ TOO SMALL FOR OPTIMIZATION-DUSQ RESET		
DUSQ REQUIRED FOR DBETA	0.99495189E 03	
NUMERATOR OF FIRST LAGRANGE MULTIPLIER	0.10446995E 04	
LAGRANGE MULTIPLIERS	0.20518551E 03	-0.22191217E 00
ACCURACY CHECK FOR DUSQ	0.20396513E 04	

	ALTITUDE	WEIGHT
CURRENT END POINT VALUES	18152.601318	1104.493439
PREVIOUS VALID STEP END POINT VALUES	18198.317139	1066.386520
CHANGE IN END POINTS	-45.715820	38.106918
CONSTRAINT TRAVEL INDICATOR(S)	1	
CONSTRAINT CROSS-OVER INDICATOR(S)	1	
ALLOWABLE FORWARD CONSTRAINT CHANGE(N-D)	4.850000	
ALLOWABLE BACKWARD CONSTRAINT CHANGE(N-C)	1.000000	
CONSTRAINT CHANGE ASKED FOR(N-C)	0.050000	
CONSTRAINT TOLERANCE(S)	1000.000000	
NON-LINEARITIES	-0.747942	-0.156841
STEP-SIZE COEFFICIENTS	0.401100	1.912763
MAXIMUM STEP-SIZE COEFFICIENT(S)	10.000000	
PREDICTED END POINT CHANGE(S) (TOTAL)	-592.795692	87.597941
PREDICTED END POINT CHANGE(S) (DUSQ)	-1726.432968	8.073352
PREDICTED END POINT CHANGE(S) (VIC)	1133.637283	79.524590
CHANGE IN WEIGHT VIC	111.939734	
CHANGE IN GAMMA R VIC	2.238795	
FINAL STEP SIZE COEFFICIENT	1.912763	
DUSQ FOR NEXT TRAJECTORY	2039.651367	
NEXT TRAJECTORY WILL BE A VALID STEP		

TIME Q	WEIGHT MACH NUMBER	ALTITUDE WEIGHT FLOW	GAMMA R ALPHA	VELOCITY R PITCH ANGLE	THRUST	LIFT	DRAG
-0.000	1611.940	500.000	62.239	600.000	4921.729	674.631	499.030
421.757	0.538	25.000	10.664	72.903			
0.500	1599.440	772.873	62.871	629.861	4942.210	553.834	420.293
461.069	0.566	25.000	8.008	70.879			
1.250	1580.690	1209.892	63.210	677.365	4974.670	328.963	347.486
526.418	0.609	25.000	4.166	67.376			
2.000	1561.940	1679.385	62.807	727.192	5009.077	35.740	344.409
598.354	0.655	25.000	0.398	63.205			
2.750	1543.190	2179.021	61.634	777.954	5045.171	-338.138	438.584
674.735	0.702	25.000	-3.341	58.293			
3.500	1524.440	2704.175	59.632	827.658	5082.532	-817.511	673.005
751.867	0.748	25.000	-7.249	52.384			
4.625	1496.315	3520.764	54.843	893.304	5139.474	-1776.120	1449.194
854.745	0.810	25.000	-13.853	40.990			
5.750	1468.190	4324.671	47.413	932.396	5194.183	-2830.618	2772.825
908.956	0.848	25.000	-20.761	26.652			
5.972	1462.634	4476.731	46.567	946.030	5204.382	829.263	770.940
931.452	0.861	25.000	5.935	52.502			
6.194	1457.079	4630.588	46.877	961.043	5214.681	1277.296	1032.216
956.783	0.875	25.000	8.900	55.777			
6.417	1451.523	4788.485	47.177	976.188	5225.147	1212.689	1000.150
982.488	0.889	25.000	8.229	55.406			
6.750	1443.190	5030.755	47.560	999.446	5241.149	1115.798	961.770
1022.336	0.911	25.000	7.276	54.836			
7.250	1430.690	5407.516	47.989	1035.413	5265.801	967.247	922.927
1084.772	0.945	25.000	5.944	53.934			
8.000	1411.940	6001.856	48.325	1091.357	5304.118	732.860	901.957
1183.553	0.998	25.000	4.128	52.453			
9.125	1383.815	6955.196	48.189	1178.465	5364.142	359.697	946.902
1340.342	1.082	25.000	1.789	49.978			
10.250	1355.690	7974.733	47.350	1267.674	5426.412	-34.541	1069.472
1502.966	1.168	25.000	-0.153	47.197			
11.375	1327.565	9048.618	45.892	1357.352	5489.906	-421.016	1251.190
1666.614	1.255	25.000	-1.684	44.208			

TIME Q	WEIGHT MACH NUMBER	ALTITUDE WEIGHT FLOW	GAMMA R ALPHA	VELOCITY R PITCH ANGLE	THRUST	LIFT	DRAG
13.062	1285.377	10730.072	42.836	1490.702	5585.130	-851.944	1563.386
1906.894	1.387	25.000	-2.978	39.857			
14.750	1243.190	12458.354	39.553	1624.168	5677.865	-699.756	1771.316
2142.732	1.521	25.000	-2.177	37.376			
16.437	1201.002	14236.733	38.003	1762.885	5768.072	1001.489	2012.363
2383.980	1.662	25.000	2.801	40.803			
17.187	1182.252	15070.146	38.913	1817.742	5808.590	2812.498	2775.167
2466.939	1.720	25.000	7.601	46.514			
17.937	1163.502	15954.675	40.828	1860.691	5850.402	2779.515	2778.078
2511.187	1.766	25.000	7.379	48.207			
18.437	1151.002	16574.791	41.860	1893.591	5878.998	2425.986	2616.005
2548.264	1.802	25.000	6.347	48.207			
19.187	1132.252	17547.898	43.093	1947.788	5922.707	2002.734	2472.697
2610.777	1.860	25.000	5.114	48.207			
19.860	1115.440	18462.139	43.944	2000.000	5962.500	1707.145	2408.875
2669.995	1.917	25.000	4.263	48.207			

ITERATION 1 TRIAL 2 NETA= 46 LETA= 73

MAJORITY VOTE TEST -1

TOO MANY FUNCTIONS WITH ADVERSE TRAVEL

REJECT VALID STEP		
DUSC REQUIRED FOR DBETA	0.17390345E 03	
NUMERATOR OF FIRST LAGRANGE MULTIPLIER	0.33600939E 03	
LAGRANGE MULTIPLIERS	0.11636616E 03	-0.39952482E-01
ACCURACY CHECK FOR DUSQ	0.50991284E 03	

	ALTITUDE	WEIGHT
CURRENT END POINT VALUES	18462.139404	1115.440140
PREVIOUS VALID STEP END POINT VALUES	18198.317139	1066.386520
CHANGE IN END POINTS	263.822266	49.053619
CONSTRAINT TRAVEL INDICATOR(S)	-1	
CONSTRAINT CROSS-OVER INDICATOR(S)	1	
ALLOWABLE FORWARD CONSTRAINT CHANGE(N-D)	0.000000	
ALLOWABLE BACKWARD CONSTRAINT CHANGE(N-D)	0.000000	
CONSTRAINT CHANGE ASKED FOR(N-D)	0.050000	
CONSTRAINT TOLERANCE(S)	1000.000000	
NON-LINEARITIES	0.000000	-0.156841
STEP-SIZE COEFFICIENTS	0.000000	1.912763
MAXIMUM STEP-SIZE COEFFICIENT(S)	0.000000	
PREDICTED END POINT CHANGE(S) (TOTAL)	-154.957928	43.896452
PREDICTED END POINT CHANGE(S) (DUSQ)	-721.776566	4.134158
PREDICTED END POINT CHANGE(S) (VIC)	566.818642	39.762295
CHANGE IN WEIGHT VIC	55.969867	
CHANGE IN GAMMA R VIC	1.119397	
FINAL STEP SIZE COEFFICIENT	0.500000	
DUSQ FOR NEXT TRAJECTORY	509.912842	
NEXT TRAJECTORY WILL BE A VALID STEP		

TIME	WEIGHT	ALTITUDE	GAMMA R	VELOCITY R	THRUST	LIFT	DRAG
Q	MACH NUMBER	WEIGHT FLOW	ALPHA	PITCH ANGLE			
-0.000	1555.970	500.000	61.119	600.000	4921.729	424.471	336.050
421.757	0.538	25.000	6.710	67.829			
0.500	1543.470	770.210	61.290	633.312	4942.011	378.055	333.821
466.171	0.569	25.000	5.407	66.696			
1.250	1524.720	1203.642	61.264	684.577	4974.209	280.574	340.011
537.786	0.616	25.000	3.478	64.742			
2.000	1505.970	1670.359	60.882	737.246	5008.420	142.118	363.235
615.178	0.664	25.000	1.540	62.422			
3.125	1477.845	2430.178	59.581	818.194	5063.112	-161.037	451.064
740.791	0.739	25.000	-1.449	58.132			
4.250	1449.720	3254.056	57.252	899.669	5121.030	-630.281	653.142
873.922	0.815	25.000	-4.808	52.444			
5.375	1421.595	4124.881	53.592	977.405	5180.710	-1347.066	1091.385
1004.861	0.888	25.000	-8.937	44.655			
5.708	1413.262	4387.560	52.182	998.503	5198.407	-1631.839	1307.993
1040.442	0.908	25.000	-10.456	41.726			
6.042	1404.928	4651.805	51.555	1022.655	5216.068	934.577	903.562
1082.712	0.931	25.000	5.755	57.310			
6.264	1399.373	4831.407	51.710	1038.944	5227.991	887.038	902.060
1111.425	0.946	25.000	5.321	57.030			
6.597	1391.039	5106.821	51.897	1063.664	5246.149	816.657	905.229
1155.270	0.970	25.000	4.713	56.610			
7.097	1378.539	5533.309	52.084	1101.318	5273.969	709.754	919.939
1222.578	1.006	25.000	3.870	55.954			
7.847	1359.789	6202.468	52.169	1158.862	5316.895	547.583	962.273
1326.350	1.061	25.000	2.752	54.921			
8.972	1331.664	7269.720	51.896	1246.938	5383.562	299.734	1063.712
1486.182	1.146	25.000	1.345	53.241			
10.660	1289.477	9001.749	50.712	1381.293	5487.179	-72.060	1280.892
1728.448	1.277	25.000	-0.278	50.435			
12.347	1247.289	10868.727	48.785	1516.506	5592.759	-392.434	1545.421
1964.861	1.412	25.000	-1.332	47.453			
14.035	1205.102	12843.162	46.455	1652.085	5697.825	-511.069	1781.726
2189.885	1.550	25.000	-1.556	44.900			

TIME	WEIGHT	ALTITUDE	GAMMA R	VELOCITY R	THRUST	LIFT	DRAG
Q	MACH NUMBER	WEIGHT FLOW	ALPHA	PITCH ANGLE			
15.722	1162.914	14910.199	44.442	1791.874	5800.899	-113.223	1929.129
2409.748	1.694	25.000	-0.313	44.129			
17.410	1120.727	17099.265	44.322	1933.814	5902.731	1521.893	2311.300
2612.022	1.844	25.000	3.884	48.207			
18.218	1100.523	13216.741	44.994	2000.000	5951.938	1297.230	2309.866
2691.984	1.915	25.000	3.213	48.207			

ITERATION 1 TRIAL 3 NETA= 36 LETA= 73

MAJORITY VOTE TEST 0

THETA TABLE 11 LINES

-0.000000	67.828964	0.250000	67.281850	0.500000	66.696447	0.875000
65.760860	1.250000	64.741703	1.625000	63.633300	2.000000	62.422361
2.562500	60.429440	3.125000	58.131699	3.687500	55.520065	4.250000
52.444272	4.812500	48.851157	5.375000	44.655149	5.541667	43.236309
5.708333	41.726336	5.875000	49.142176	6.041667	57.309796	6.152778
57.170612	6.263889	57.030365	6.430555	56.819995	6.597222	56.609626
6.847222	56.283332	7.097222	55.954317	7.472222	55.443798	7.847222
54.920916	8.409722	54.100831	8.972222	53.240920	9.815972	51.875994
10.659722	50.434552	11.503471	48.943418	12.347222	47.453438	13.190971
46.046348	14.034721	44.899603	14.878471	44.153043	15.722221	44.129181
18.217864	48.206633	182.178642	48.206633			

RESTART TABLE

0 1						
509.912842	0.050000	-0.000000	17.344538	1066.386520	18198.317139	1000.000000
0.050000						

TRAJECTORY SUMMARY

****

TIME	WEIGHT	ALTITUDE	GAMMA R	VELOCITY R	THRUST	LIFT	DRAG
Q	MACH NUMBER	WEIGHT FLOW	ALPHA	PITCH ANGLE			
17.345	1066.387	18198.317	48.207	2000.000	5951.141	0.000	2154.912
2693.640	1.915	25.000	0.000	48.207			
18.218	1100.523	13216.741	44.994	2000.000	5951.938	1297.230	2309.866
2691.984	1.915	25.000	3.213	48.207			

Program setup. — The program is divided into nine logical blocks or links, where each link performs a particular job. These links are required to ensure that the program will fit into a 32K core computer. Additional information is given in the section on program and data overlay.

Link 0: Controls the flow of the program from link to link and remains in core at all times. The link consists of the following subroutines:

LOAD	LOOK1D
ATMOS	UNITZ
BLOCK DATA	

Link 1: All operations related to the forward trajectory are performed. The subroutines included in the link are:

AKSTP	EXEC	PLAC
ANPARP	FPROG	STEP1
ANPRTL	INITCO	STP1
BOOM	LOOK3D	TLD
CONTR		

Link 2: The numerical partial derivative check is made using the subroutine:

PRTIAL

Link 3: The control variable history for the nominal trajectory generation is made using the subroutine:

GUIDE

Link 4: The operations related to the automatic convergence logic are performed. The subroutines included in the link are:

KCALC	MATRX2	VALID
MATOUT	UCALC	VARIC
MATRIX		

Link 5: All operations and controls related to the backward trajectory are performed. The link consists of the following subroutines:

DVAL2	STEP2
LAMBDA	STP2

Link 6: This link performs the initialization that is required once per data case. The subroutines included are:

INITAL

TITLES

Link 7: The final output of trajectory data that is required only once per data case is performed. The subroutine called is

CARDS

Link 8: The last link controls the plotting of data and is called once per run. The subroutines used are

PLOTZ

SKALZ

A set of auxiliary subroutines are required for plotting similar to the Boeing numerical plotting system (NPS).

Organization of links: The organization of the computer links and the approximate core storage required are shown in figure 17. This chart shows that all links of origin one are called from the basic link 0. Links 2 and 3 (origin two) are called only by the forward trajectory, link 1.

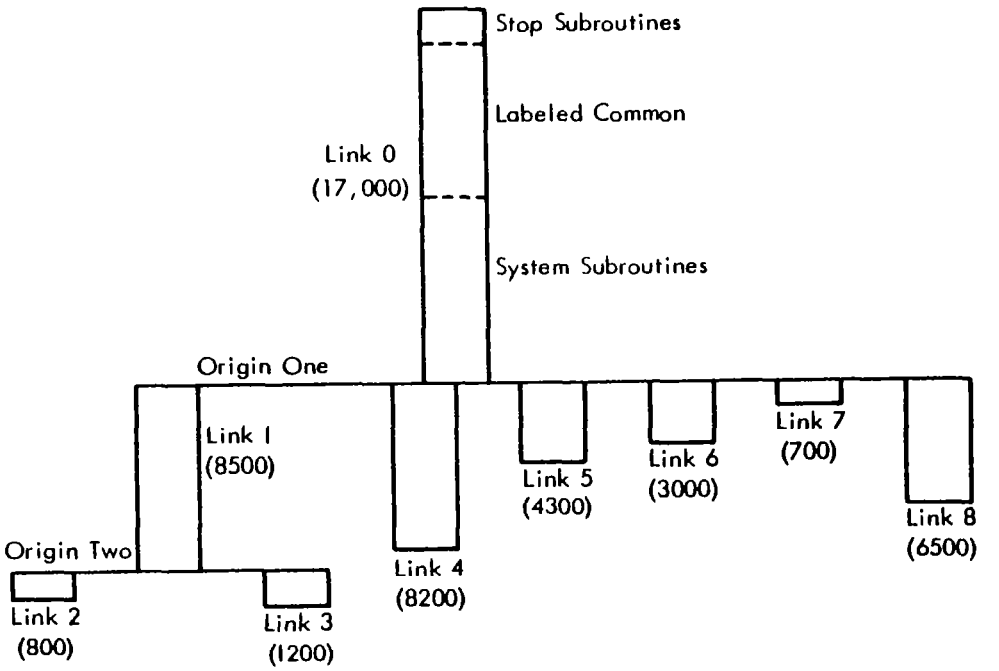


Figure 17. PROGRAM ORGANIZATION

Physically in the computer, links 1, 4, 5, 6, 7, and 8 occupy the space beginning with the same storage location (ORIGIN ONE) but at different times as governed by the main program (LOAD). Similarly links 2 and 3 occupy the storage beginning with the same location (ORIGIN TWO).

Program deck: The recommended program deck setup is shown in figure 18. The control cards required for machine operation precede link 0. The data deck for a given problem is placed following link 8.

To speed up the overlaying of links, two additional tape (or disk) units are used to store the program. The forward integration routines (links 1, 2, and 3) are stored on SYSLB3. The convergence logic routines (link 4) are stored on SYSCK1. All other links are stored on SYSUT2.

Data setup. — The setup of the data is performed as detailed in "Input Data Preparation." The physical setup of the data deck is shown in figure 19. Note that each set of stage-dependent data must be preceded by an STG card. A STG card must be present for a stage even if no stage-dependent tables are input. The last card in the data deck is an END card.

General Machine Operation

Tape or disk requirements. — STOP may use a maximum of 10 I/O units during execution, excluding the system units reserved for input, output, punch, and normal link loading (i.e., SYNIN1, SYSOU1, SYSPP1, SYSUT2).

Listed in figure 20 is a brief description of all I/O units used — their program symbol, system name, function, mode, and buffer size. (Buffer sizes are set in subroutine UNITZ.)

Figure 21 shows the I/O units used during execution of a typical data case. This data case is set to complete 1 iteration (NARBY = 2, NITC = 1), punch the control table used for the last iteration (NC(8) = 1), and plot the nominal and last iteration (NC(11) = -1).

End of run indication. — The program will normally exit successfully by reading an end of file on the input tape (SYSIN1). The monitor will print out the comment "END OF FILE READING." The program has the capability to run multiple data cases, and always terminates one case by trying to read the next.

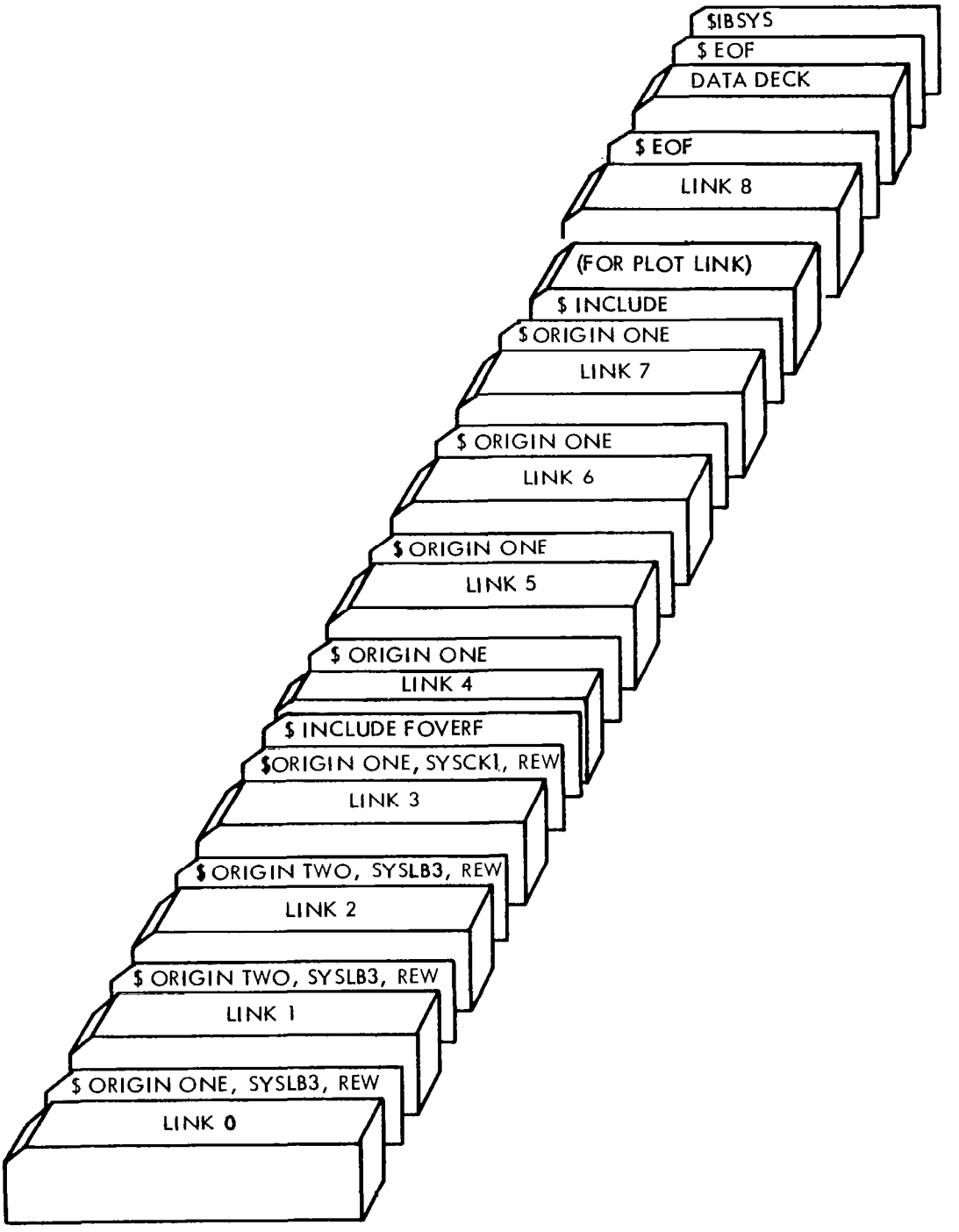


Figure 18. PROGRAM SETUP

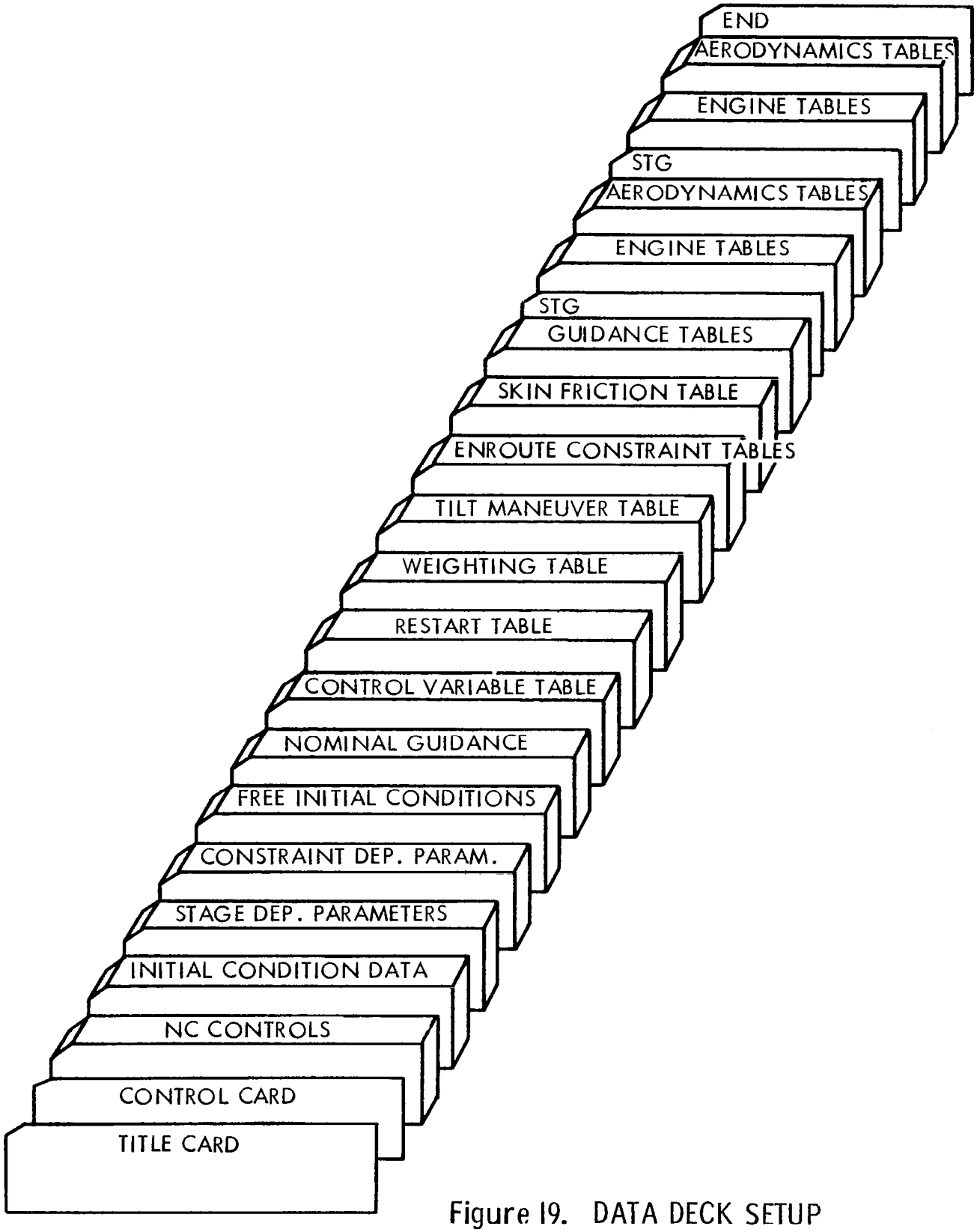


Figure 19. DATA DECK SETUP

Program Symbol	System Name	Function	Mode	Buffer Size (decimal)	Remarks
KINP	SYSIN1	Input	BCD	14	System Units
KOUT	SYSOU1	Printed output	BCD	120	
KPUN	SYSPP1	Punch output	BCD	22	
	SYSUT2	Link storage	BIN		
	SYSLB3	Forward integration link	BIN		Link storage
	SYSCK1	Matrix link	BIN		
KDAT	SYSUT6	Input data storage	BIN	256	Scratch units required by STOP
KLAM	SYSUT3	Impulse response storage	BIN	256	
KPAR	SYSUT4	Partial storage	BIN	256	
KPLT	SYSUT1	Plot data storage	BIN	20	
*KSCR	SYSUT9	Control table storage	BIN	20	
KTAN	SYSUT5	Control table overlay	BIN	256	
	SYSUT7	Scratch tape for the NPS generated code	BIN	256	Used only when plotting
	*SYSCK2	Plot output	BCD	22	

* These units must be assigned to TAPE, as they may be saved

Figure 20. INPUT/OUTPUT (I/O) USAGE

Special machine operating information. — The two I/O units, KSCR and SYSCK2, must be assigned as physical tape units and comments inserted asking that these tapes be saved.

Routines Loaded From	Link Number	I/O Unit Read	I/O Unit Written	Purpose
CORE	0			
SYSUT2	6	KINP	KOUT KDAT KTAN	Initialize data
SYSLB3	1	KDAT	KOUT KPAR KSCR KPLT	Forward integration of nominal
SYSCK1	4		KOUT	Majority vote test
SYSUT2	5	KPAR	KLAM KOUT KDAT	Backward integration
SYSCK1	4	KLAM	KTAN KOUT KPAR	1. Steepest ascent logic 2. Build new control table
SYSLB3	1	KDAT KTAN	KOUT	Forward integration of try 1
SYSCK1	4	KLAM KPAR	KTAN KOUT KPAR	Analyze try 1, make another try or run valid step
SYSLB3	1	KDAT KTAN	KOUT KPAR KSCR KPLT	Forward integration of iteration 1
SYSUT2	7	KSCR KDAT	KOUT KPUN	Print and punch final output
SYSUT2	8	KPLT	KOUT SYSUT7 SYSCK2	Prepare plot tape
SYSUT2	6	KINP		Exits trying to read next data case

Note: KTAN will not be used if the control table used does not exceed 1000 points.

Figure 21. EXAMPLE OF I/O USAGE

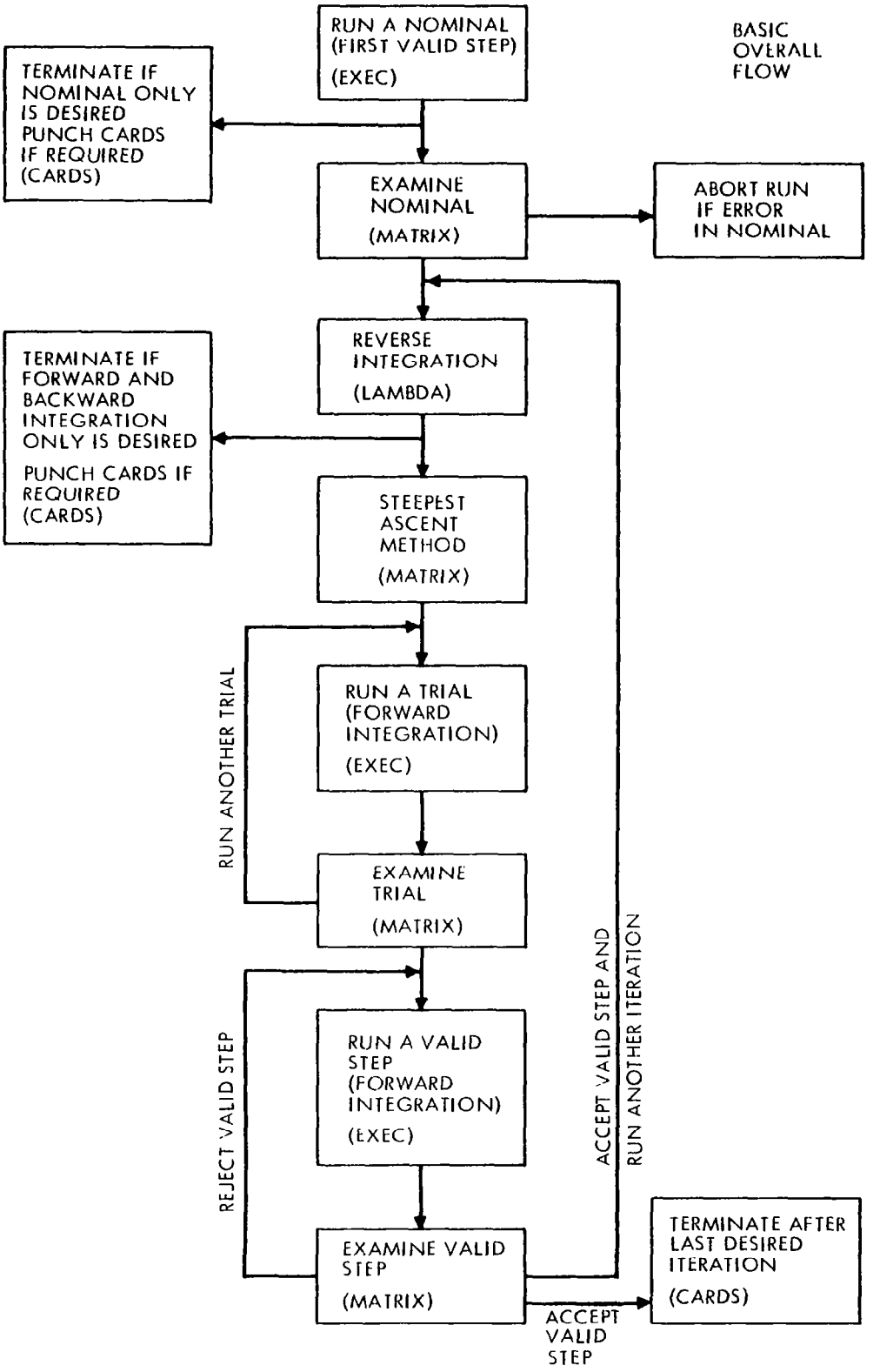
The following control cards are required for each computer run, and must precede the program deck.

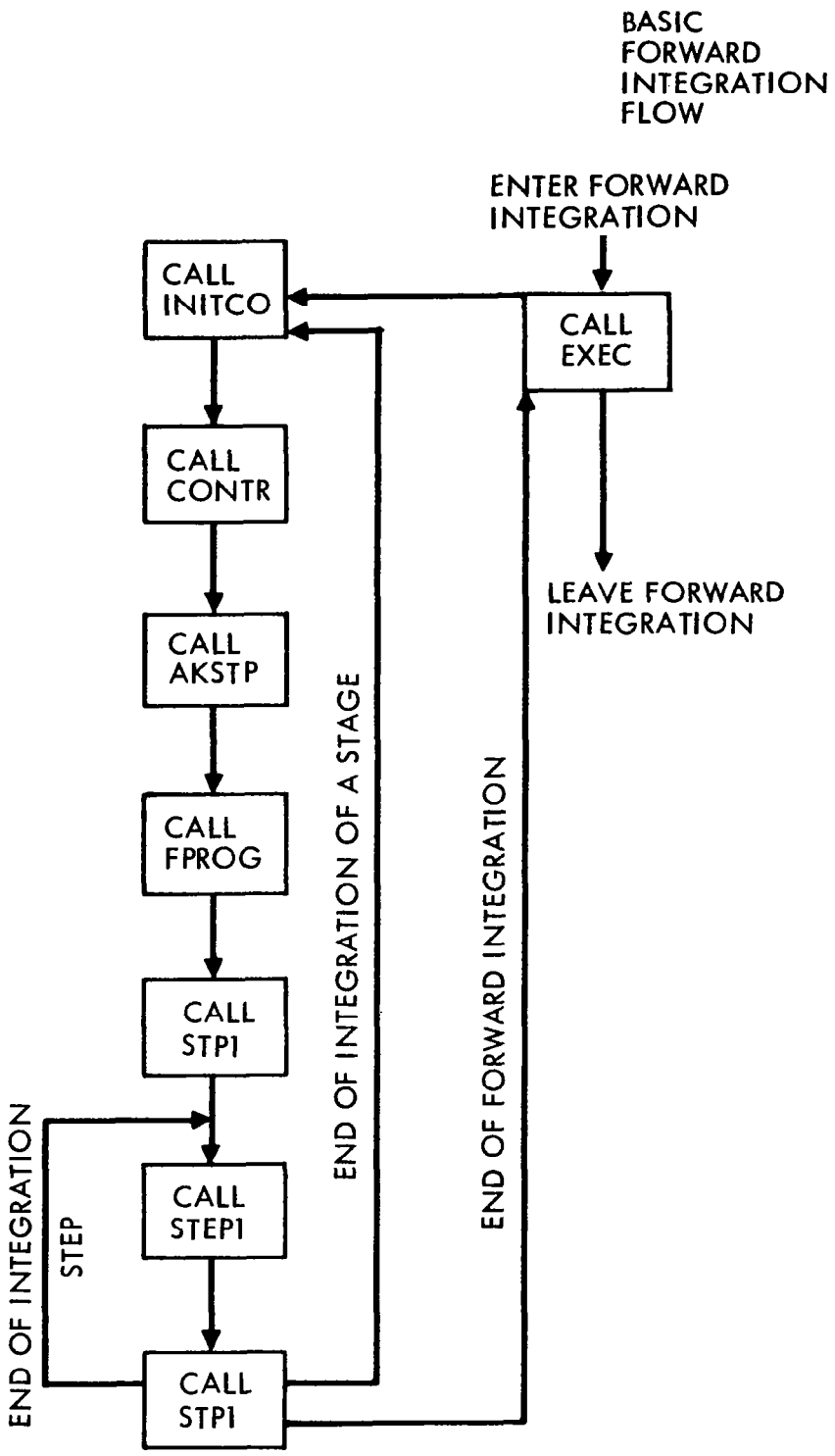
```
$JOB  99F-61C AS2413 WATSON BCC A2 5-0934
$*    ACCTTIME 5
$*    PRINT ESTIMATE (2000)
$*
$*    READY A4,A5,B3,B5,C2,C3,C4
$*
$ATTACH        A4
$AS            SYSUT7
$ATTACH        A5
$AS            SYSLB2
$ATTACH        B3
$AS            SYSUT9
$ATTACH        B5
$AS            SYSLB3
$ATTACH        C2
$AS            SYSUT5
$ATTACH        C3
$AS            SYSCK1
$ATTACH        C4
$AS            SYSUT6
$ALT           NPS
$EXECUTE       IBJOB                77777
$IBJOB         GO,LOGIC,MAP,FILES,FIOCS,EXTLIB
$POOL          -UNIT03-,-UNIT04-,-UNIT09-,-UNIT10-,-UNIT11-,BLK=256
$ETC           BUFCT=4
$GROUP         -UNIT03-,-UNIT04-,-UNIT09-,-UNIT10-,-UNIT11-,OPNCT=4
$ETC           BUFCT=4
$POOL          -UNIT08-,-UNIT14-,BLK=22,BUFCT=2
$GROUP         -UNIT08-,-UNIT14-,OPNCT=2,BUFCT=2
```

NOTE: (1) When NC(8) $\neq$ 0, a comment card must be included with the control cards, asking that a save tape be mounted on B3.

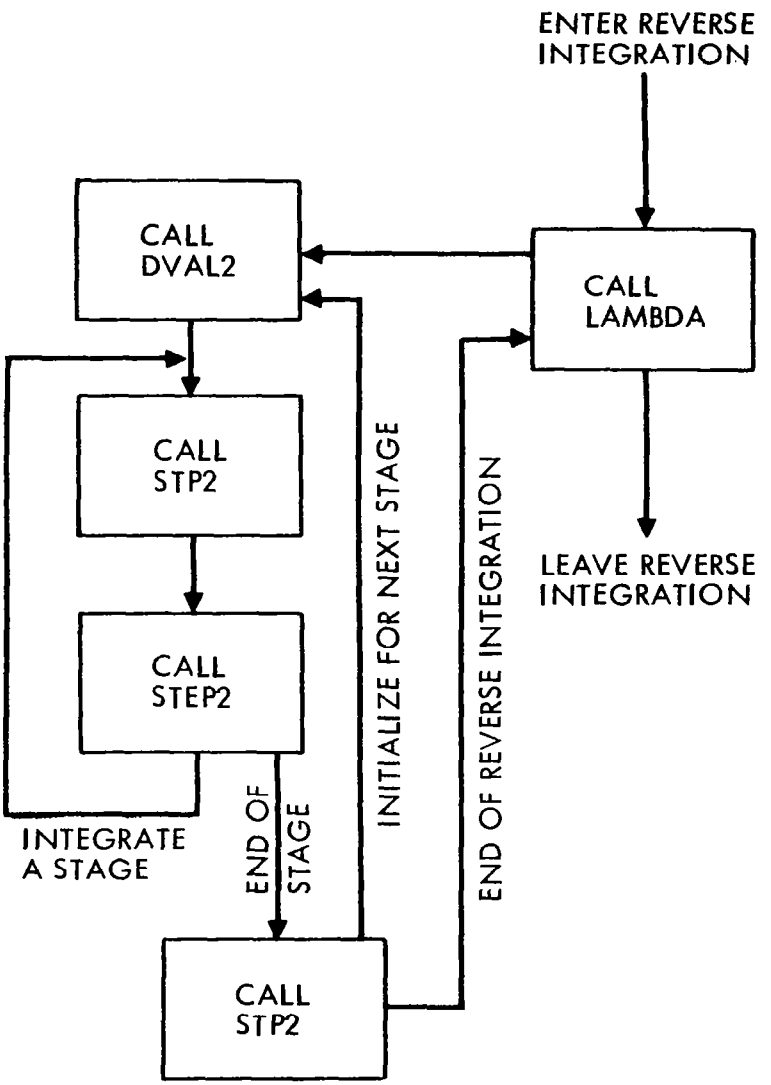
(2) When plotting (NC(11) $\neq$ 0), a comment card must be included with the control cards stating that unit C1 will contain the plot output.

Basic program flow. —The following flow charts are intended to give the user a general picture of the basic program organization and the subroutines called by each major area of the program. The flow charts are broken down into the logical areas of forward integration, reverse integration, and control logic.

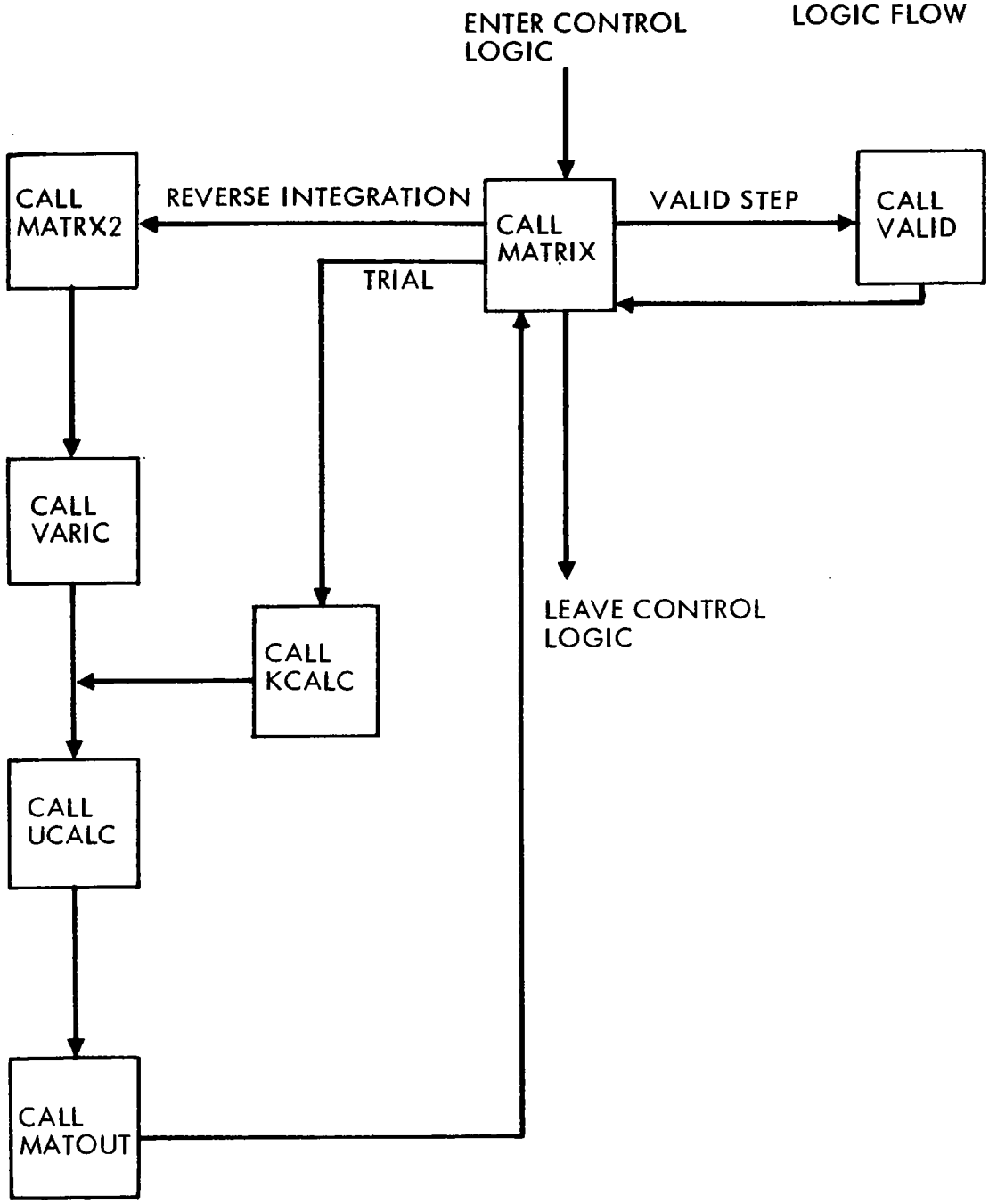

BASIC OVERALL FLOW
RUN A NOMINAL (FIRST VALID STEP) (EXEC)
TERMINATE IF NOMINAL ONLY IS DESIRED PUNCH CARDS IF REQUIRED (CARDS)
EXAMINE NOMINAL (MATRIX)
ABORT RUN IF ERROR IN NOMINAL
REVERSE INTEGRATION (LAMBDA)
TERMINATE IF FORWARD AND BACKWARD INTEGRATION ONLY IS DESIRED PUNCH CARDS IF REQUIRED (CARDS)
STEEPEST ASCENT METHOD (MATRIX)
RUN A TRIAL (FORWARD INTEGRATION) (EXEC)
RUN ANOTHER TRIAL
EXAMINE TRIAL (MATRIX)
RUN A VALID STEP (FORWARD INTEGRATION) (EXEC)
REJECT VALID STEP
ACCEPT VALID STEP AND RUN ANOTHER ITERATION
EXAMINE VALID STEP (MATRIX)
TERMINATE AFTER LAST DESIRED ITERATION (CARDS)
ACCEPT VALID STEP


BASIC
FORWARD
INTEGRATION
FLOW
ENTER FORWARD
INTEGRATION
CALL
INITCO
CALL
EXEC
CALL
CONTR
LEAVE FORWARD
INTEGRATION
CALL
AKSTP
CALL
FPROG
CALL
STP1
END OF INTEGRATION OF A STAGE
END OF FORWARD INTEGRATION
END OF INTEGRATION
STEP
CALL
STEP1
CALL
STP1

BASIC REVERSE
INTEGRATION
FLOW



BASIC CONTROL LOGIC FLOW



Subroutine descriptions and flow diagrams.— Each of the subroutines in STOP is described giving the purpose and the numerical method if applicable. Other subroutines called by the one being described are listed. The approximate core storage used is indicated. The arguments of subroutines using a calling sequence are defined in detail. Flow charts showing the organization and detail of each routine are given.

AKSTP — AK Store

Purpose

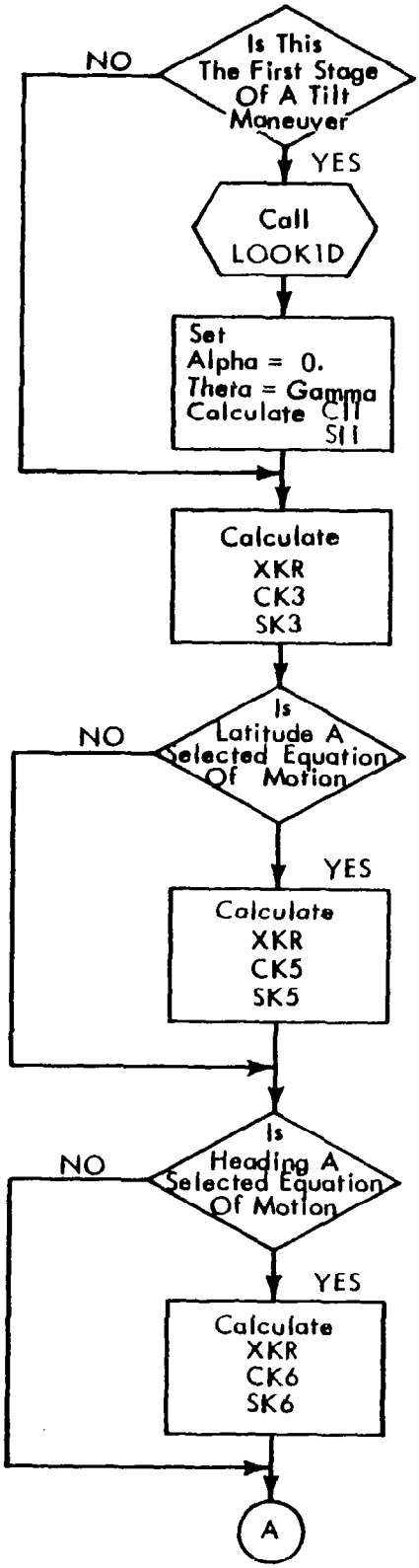
AKSTP defines and calculates basic and frequently used variables in the program. These variables include some AK's, VAR's, partials, and trigonometric functions of angle of attack, latitude, flight path angle, and heading.

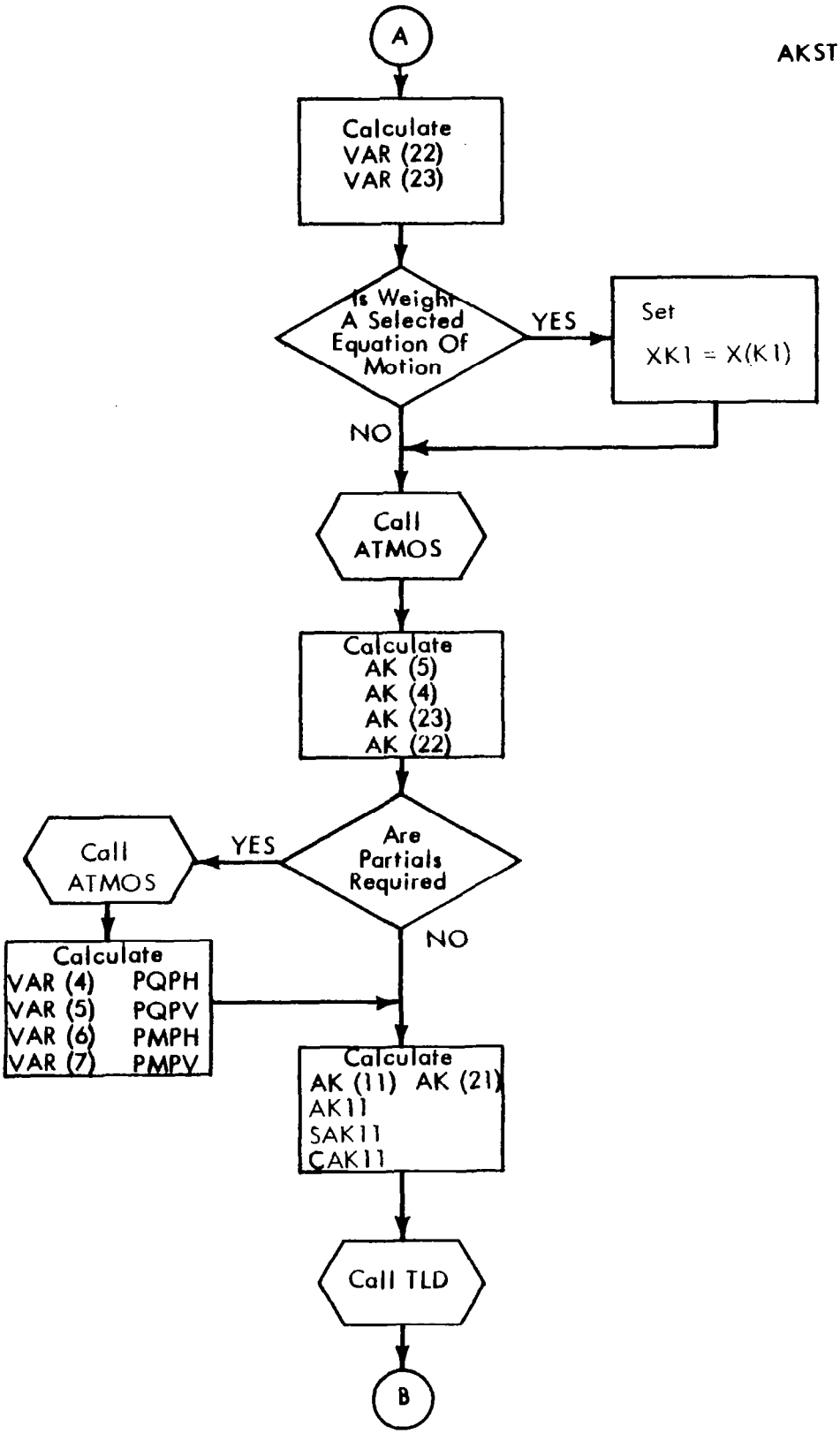
Subroutines Called

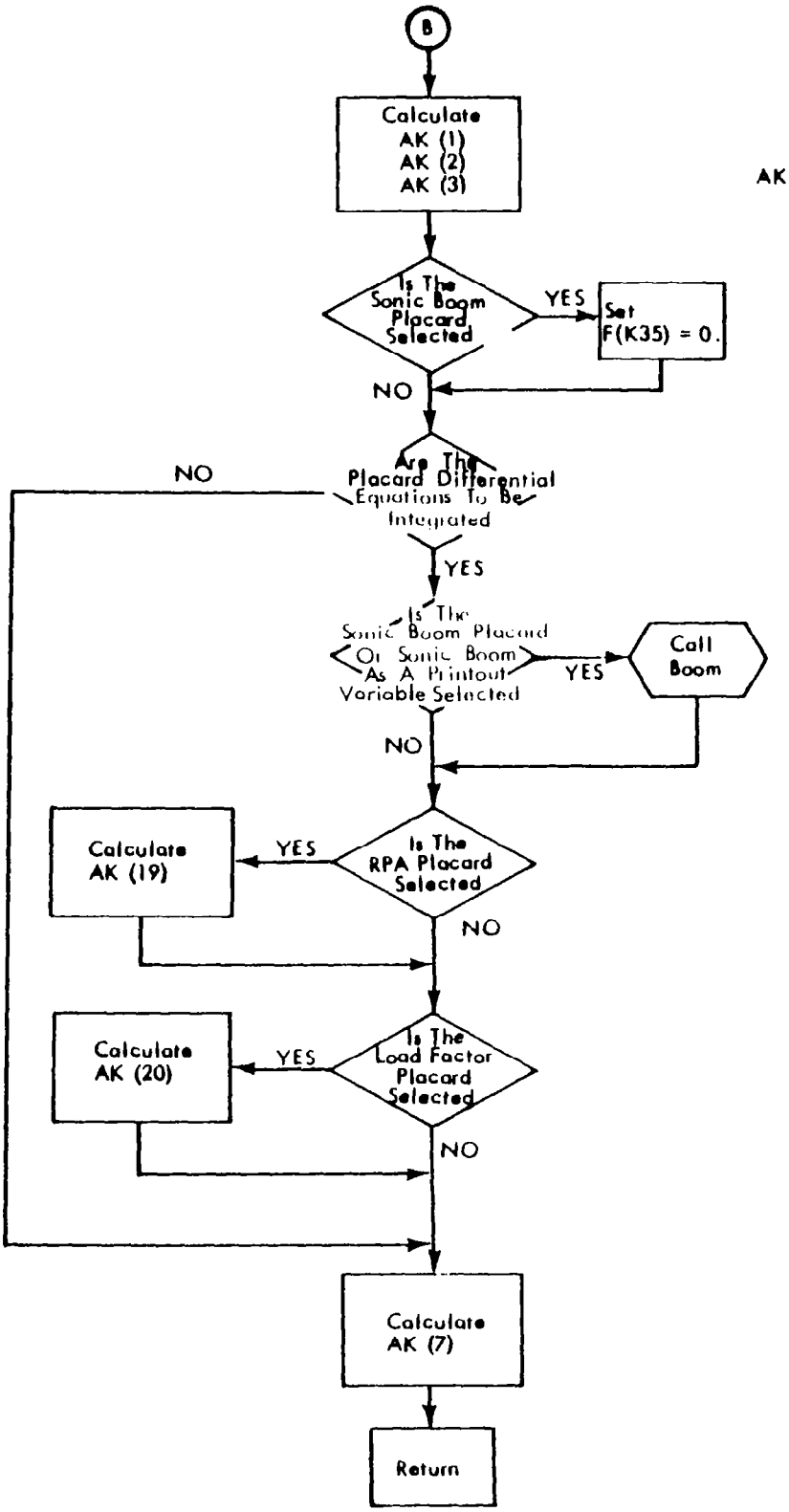
ATMOS BOOM LOOK1D TLD

Storage Used

952 cells


Is This The First Stage Of A Tilt Maneuver
NO
YES
Call LOOK1D
Set Alpha = 0. Theta = Gamma Calculate CII SII
Calculate XKR CK3 SK3
Is Latitude A Selected Equation Of Motion
NO
YES
Calculate XKR CK5 SK5
Is Heading A Selected Equation Of Motion
NO
YES
Calculate XKR CK6 SK6
A


A
Calculate
VAR (22)
VAR (23)
Is Weight
A Selected
Equation Of
Motion
YES
Set
XK1 = X(K1)
NO
Call
ATMOS
Calculate
AK (5)
AK (4)
AK (23)
AK (22)
Are
Partials
Required
YES
Call
ATMOS
NO
Calculate
VAR (4) PQPH
VAR (5) PQPV
VAR (6) PMPH
VAR (7) PMPV
Calculate
AK (11) AK (21)
AK11
SAK11
CAK11
Call TLD
B


B
Calculate
AK (1)
AK (2)
AK (3)
AKSTP
Is The
Sonic Boom
Placard
Selected
YES
Set
F(K35) = 0.
NO
Are The
Placard Differential
Equations To Be
Integrated
NO
YES
Is The
Sonic Boom Placard
Or Sonic Boom
As A Printout
Variable Selected
YES
Call
Boom
NO
Calculate
AK (19)
YES
Is The
RPA Placard
Selected
NO
Calculate
AK (20)
YES
Is The
Load Factor
Placard
Selected
NO
Calculate
AK (7)
Return

ANPARP — Analytical Partial Derivatives of Placard Equations of Motion

Purpose

The analytical partial derivatives of the placard equations of motion (part of the F array) are performed in ANPARP. These derivatives of F (placard) are with respect to all the state and control variables. See appendix B for an algebraic description of these partials.

Method

1) Derivatives of the placard equations with respect to the state variables are given as PFX(I, J) where I identifies the placard equations, and J identifies the derivative state variable; i.e., PFX(K25, K4) = ∂F(K25)/∂X(K4).

2) Derivatives of the placard equation with respect to the control variables are given as PFU(I, J) where I identifies the placard equation of the F array, and J the derivative control variable; i.e., PFU(K25, I1) = ∂F(K25)/∂U(I1).

3) All partials not calculated are set to zero.

4) Words not in COMMON computed in ANPRTL and required by ANPARP are transmitted through the calling sequence:

```
CALL ANPARP (PLPH, PLPV, PLPA, PLPS, PDPH,
PDPV, PDPA, PDPS, PTPH, PTPV, PTPA, PTPT)
```

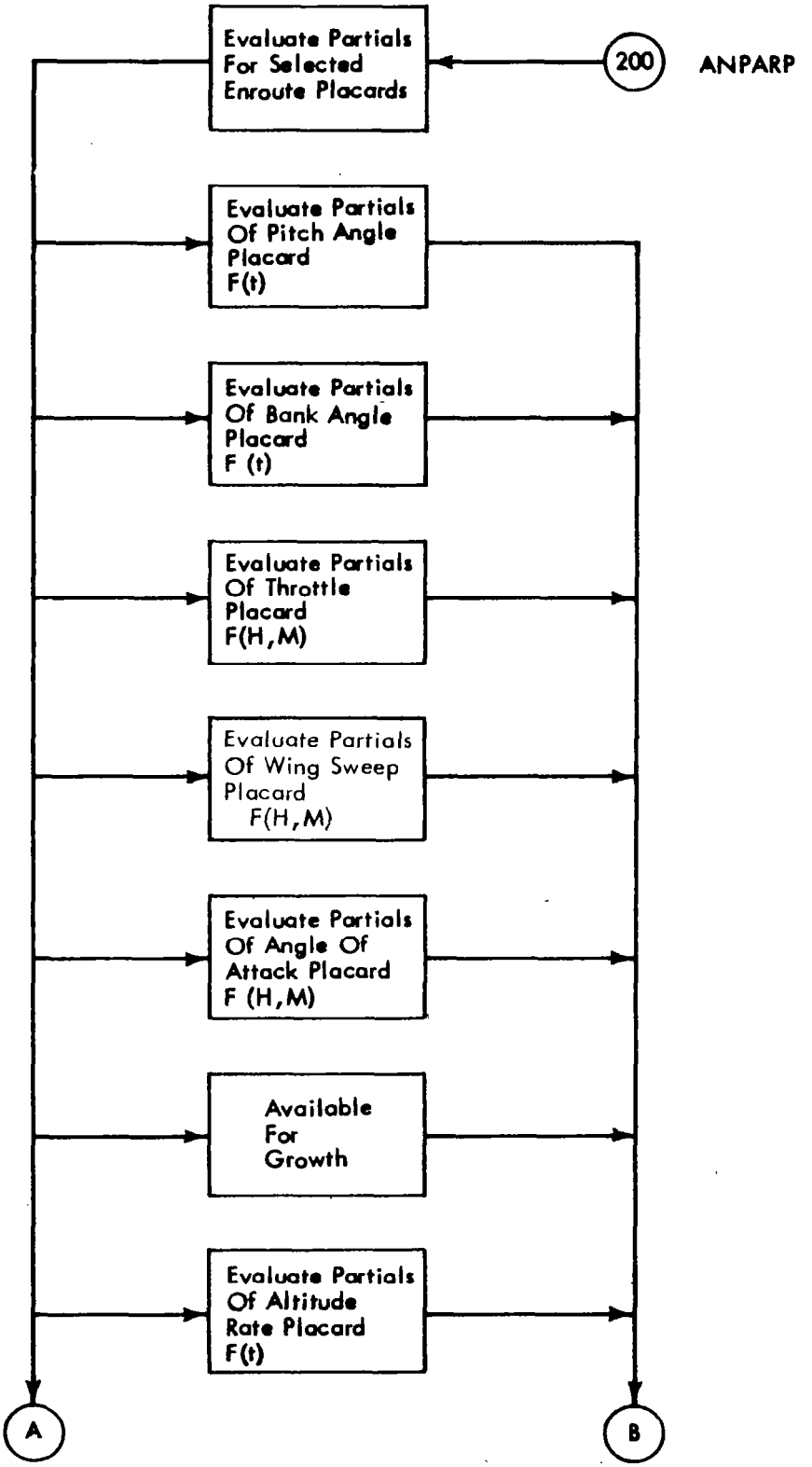
where the elements in the call are defined in appendix B.

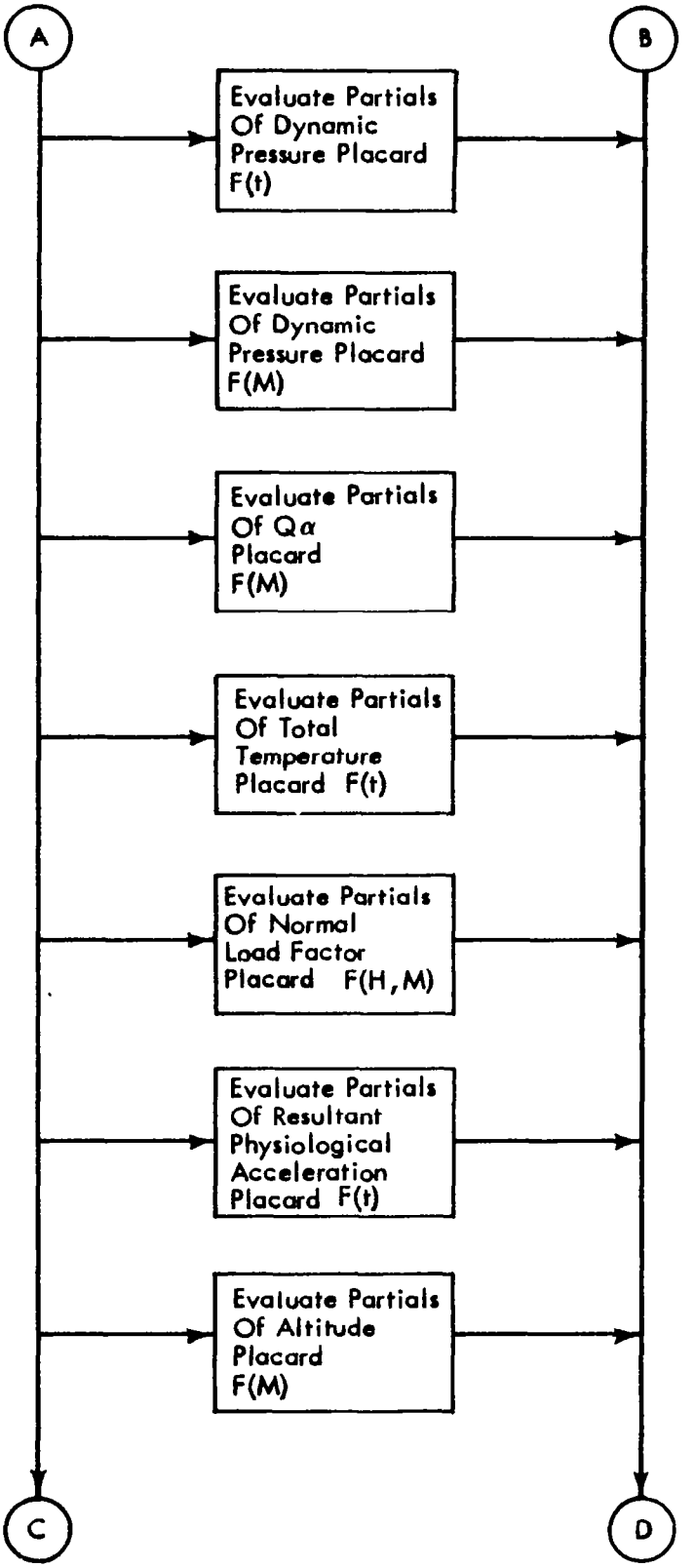
Subroutines Called

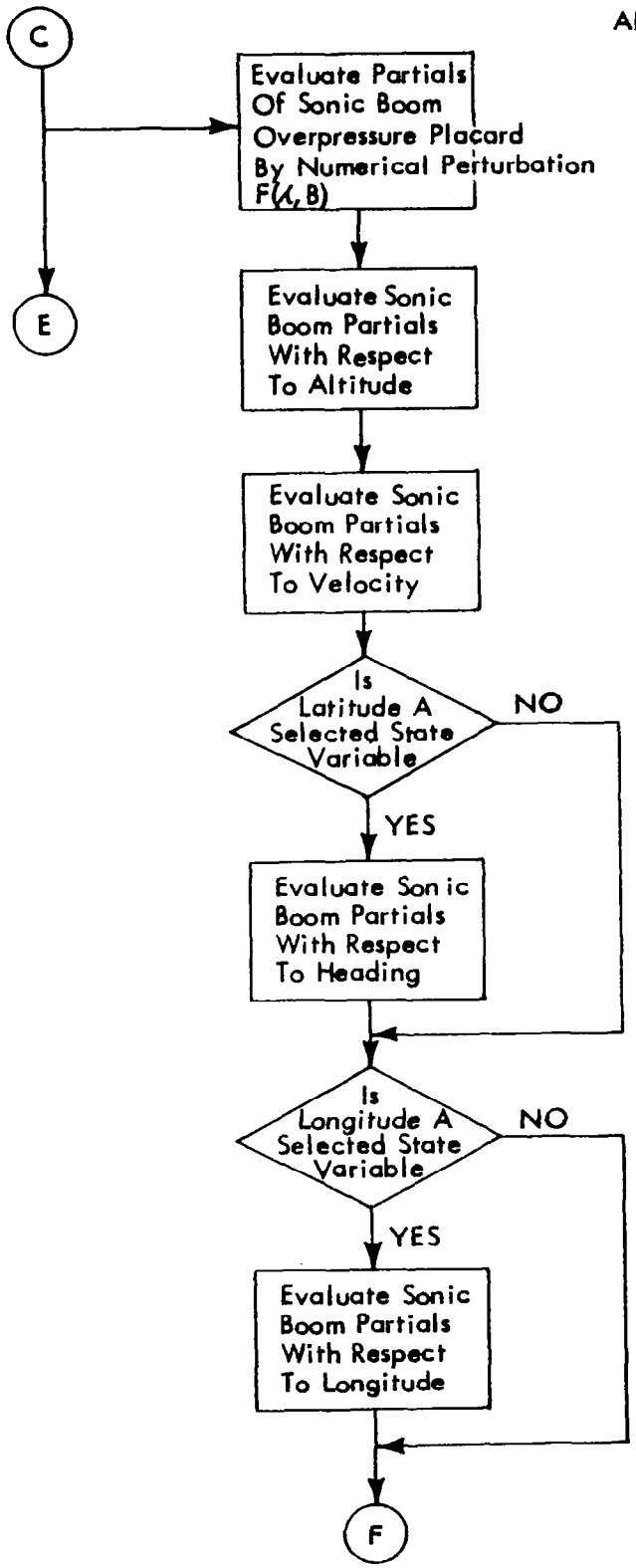
BOOM

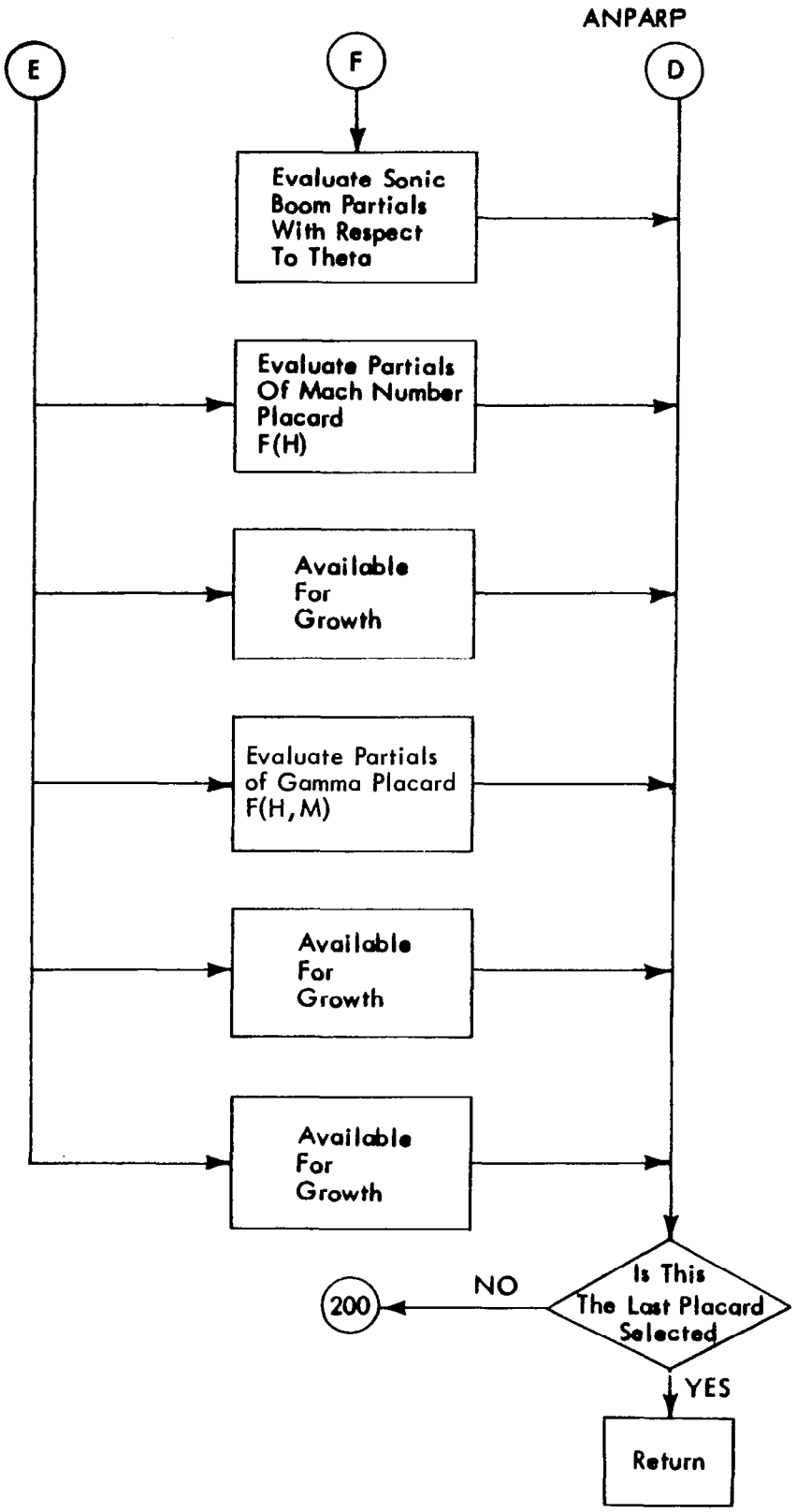
Storage Used

1,371 cells


Evaluate Partials For Selected Enroute Placards
200
ANPARP
Evaluate Partials Of Pitch Angle Placard F(t)
Evaluate Partials Of Bank Angle Placard F (t)
Evaluate Partials Of Throttle Placard F(H,M)
Evaluate Partials Of Wing Sweep Placard F(H,M)
Evaluate Partials Of Angle Of Attack Placard F (H,M)
Available For Growth
Evaluate Partials Of Altitude Rate Placard F(t)
A
B


A
B
Evaluate Partials Of Dynamic Pressure Placard F(t)
Evaluate Partials Of Dynamic Pressure Placard F(M)
Evaluate Partials Of Qα Placard F(M)
Evaluate Partials Of Total Temperature Placard F(t)
Evaluate Partials Of Normal Load Factor Placard F(H,M)
Evaluate Partials Of Resultant Physiological Acceleration Placard F(t)
Evaluate Partials Of Altitude Placard F(M)
C
D


C
Evaluate Partials Of Sonic Boom Overpressure Placard By Numerical Perturbation F(λ,B)
E
Evaluate Sonic Boom Partials With Respect To Altitude
Evaluate Sonic Boom Partials With Respect To Velocity
Is Latitude A Selected State Variable
NO
YES
Evaluate Sonic Boom Partials With Respect To Heading
Is Longitude A Selected State Variable
NO
YES
Evaluate Sonic Boom Partials With Respect To Longitude
F


ANPARP
E
F
D
Evaluate Sonic Boom Partials With Respect To Theta
Evaluate Partials Of Mach Number Placard F(H)
Available For Growth
Evaluate Partials of Gamma Placard F(H, M)
Available For Growth
Available For Growth
Is This The Last Placard Selected
NO
200
YES
Return

ANPRTL — Analytical Partial Derivatives of the Equations of Motion

Purpose

The analytical partial derivatives of the equations of motion (excluding placard equations) are calculated in ANPRTL. These derivatives of the F array are with respect to all the state and control variables. Appendix B gives an algebraic description of these partials. ANPRTL is called from STP1 only during valid trajectories.

Method

1) Derivatives of the equations of motion with respect to the state variables are given as PFX(I, J) where I identifies the equation of motion and J identifies the derivative state variable; i.e., PFX(K2, K4) = ∂F(K2)/∂X(K4).

2) Derivatives of the equations of motion with respect to the control variables are given as PFU(I, J) where I identifies the equation of motion and J identifies the derivative control variable; i.e., PFU(K2, I1) = ∂F(K2)/∂U(I1).

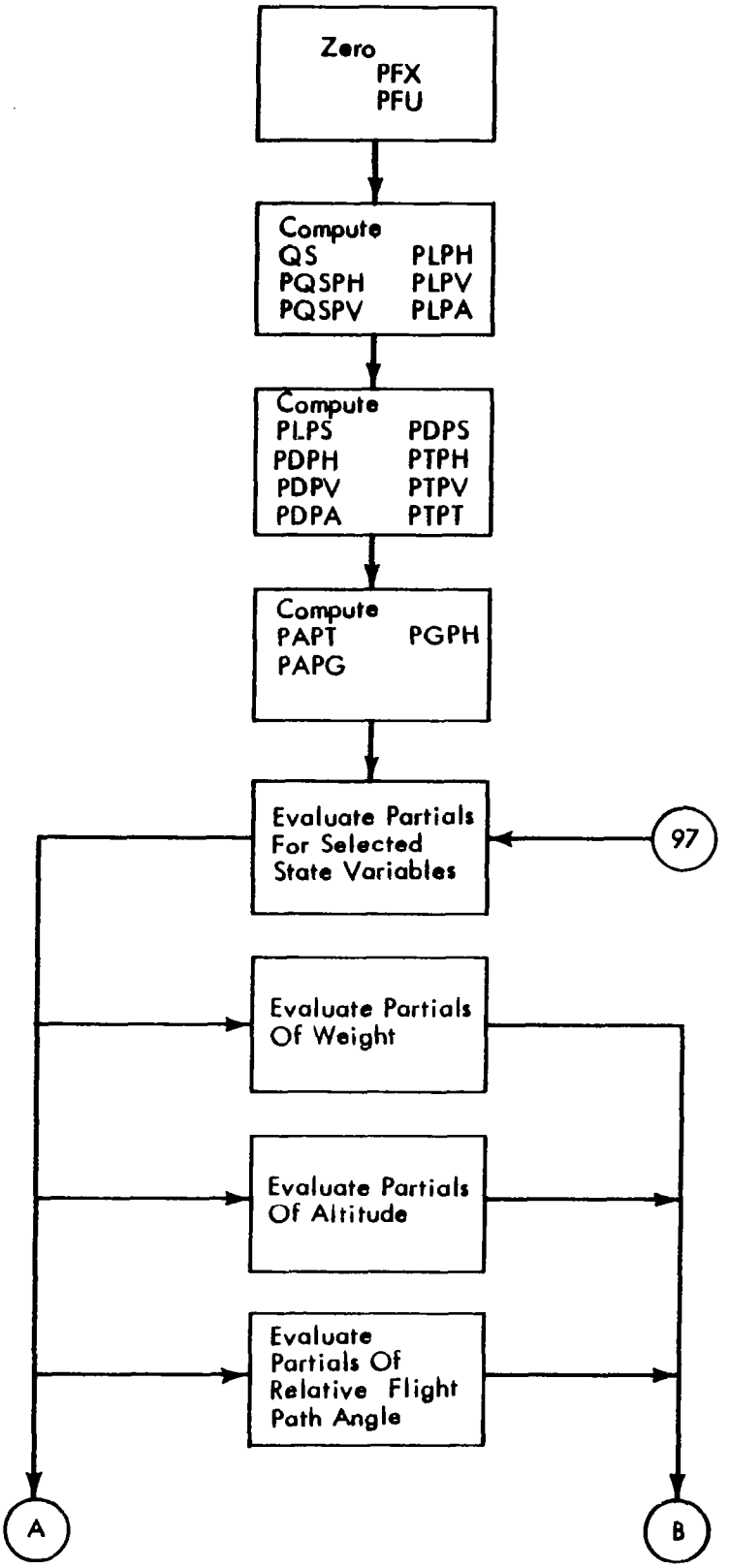
3) All partials not calculated are set to zero.

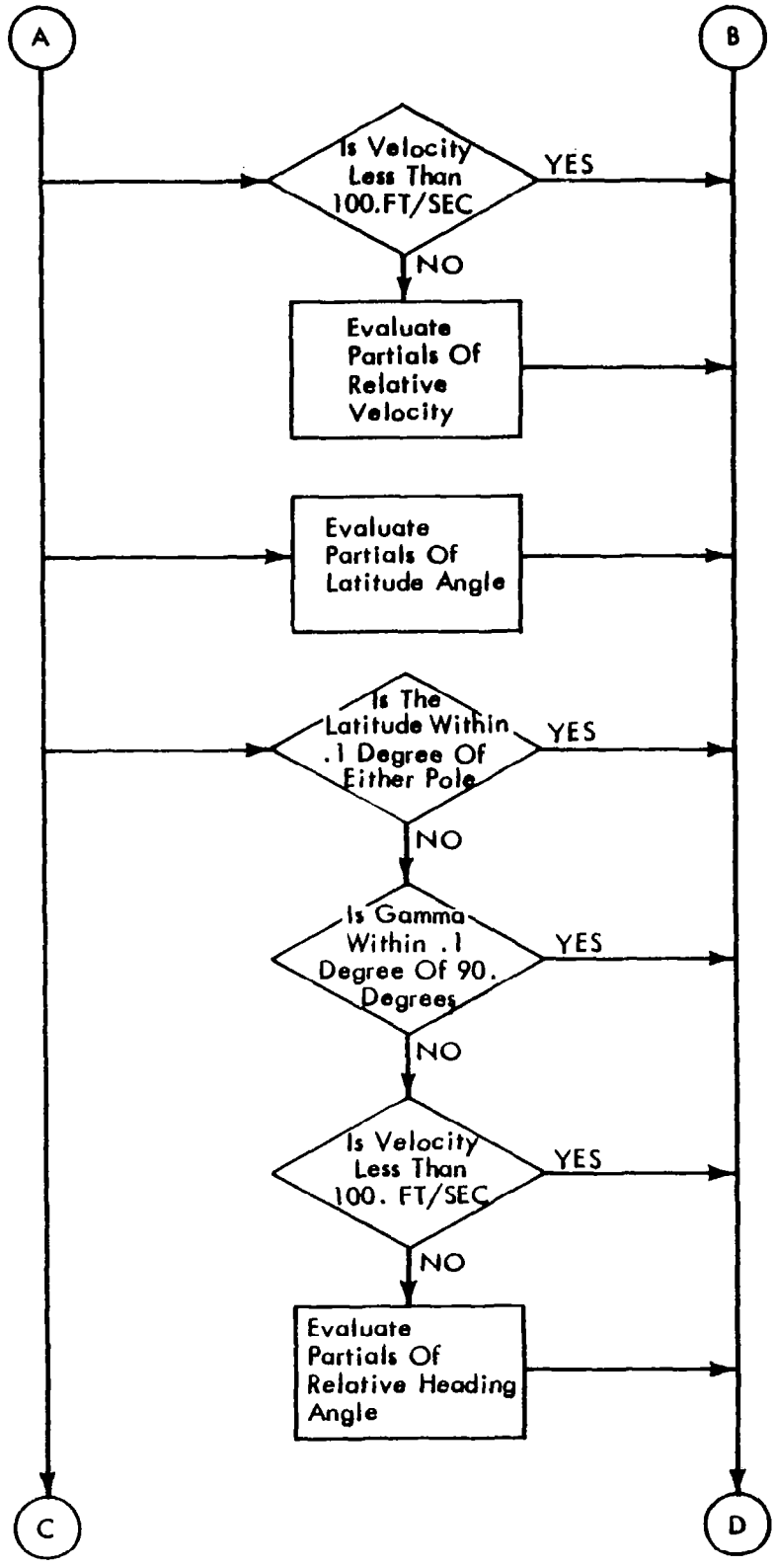
Subroutines Called

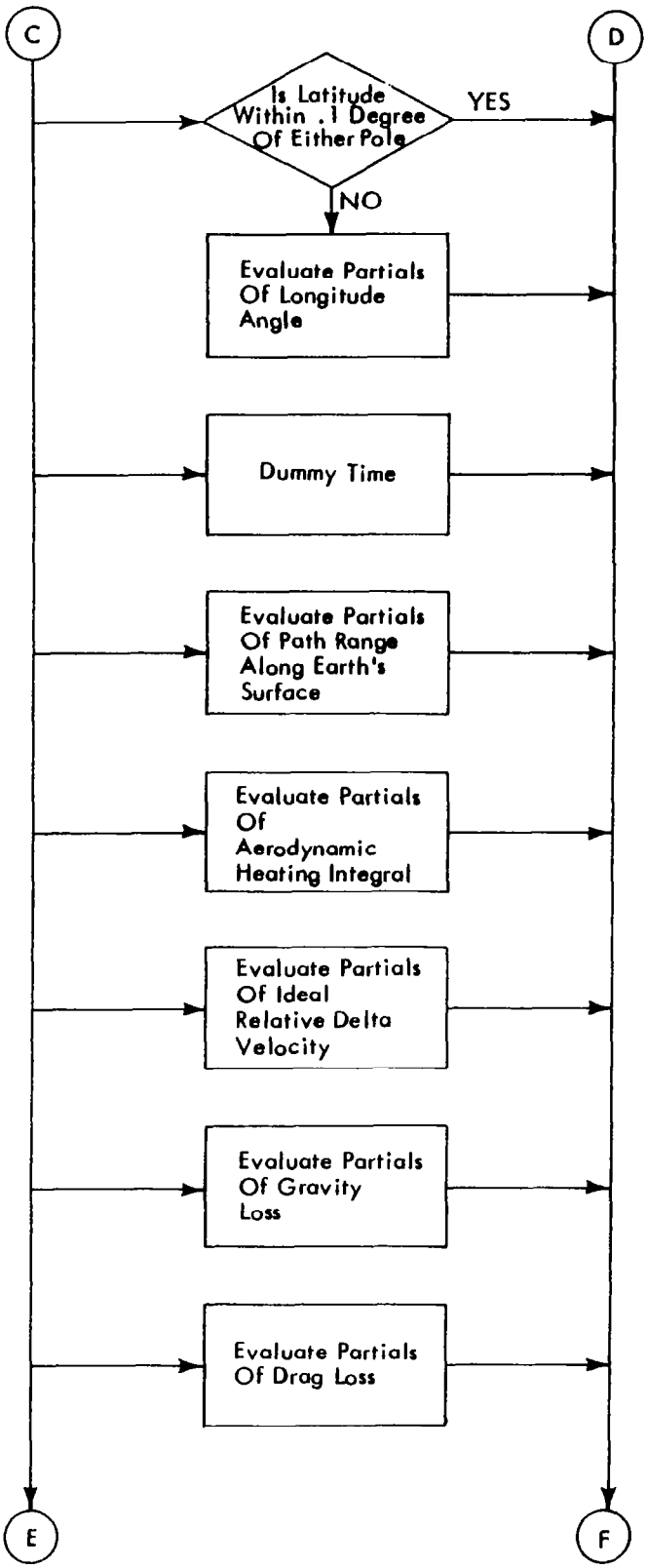
ANPARP

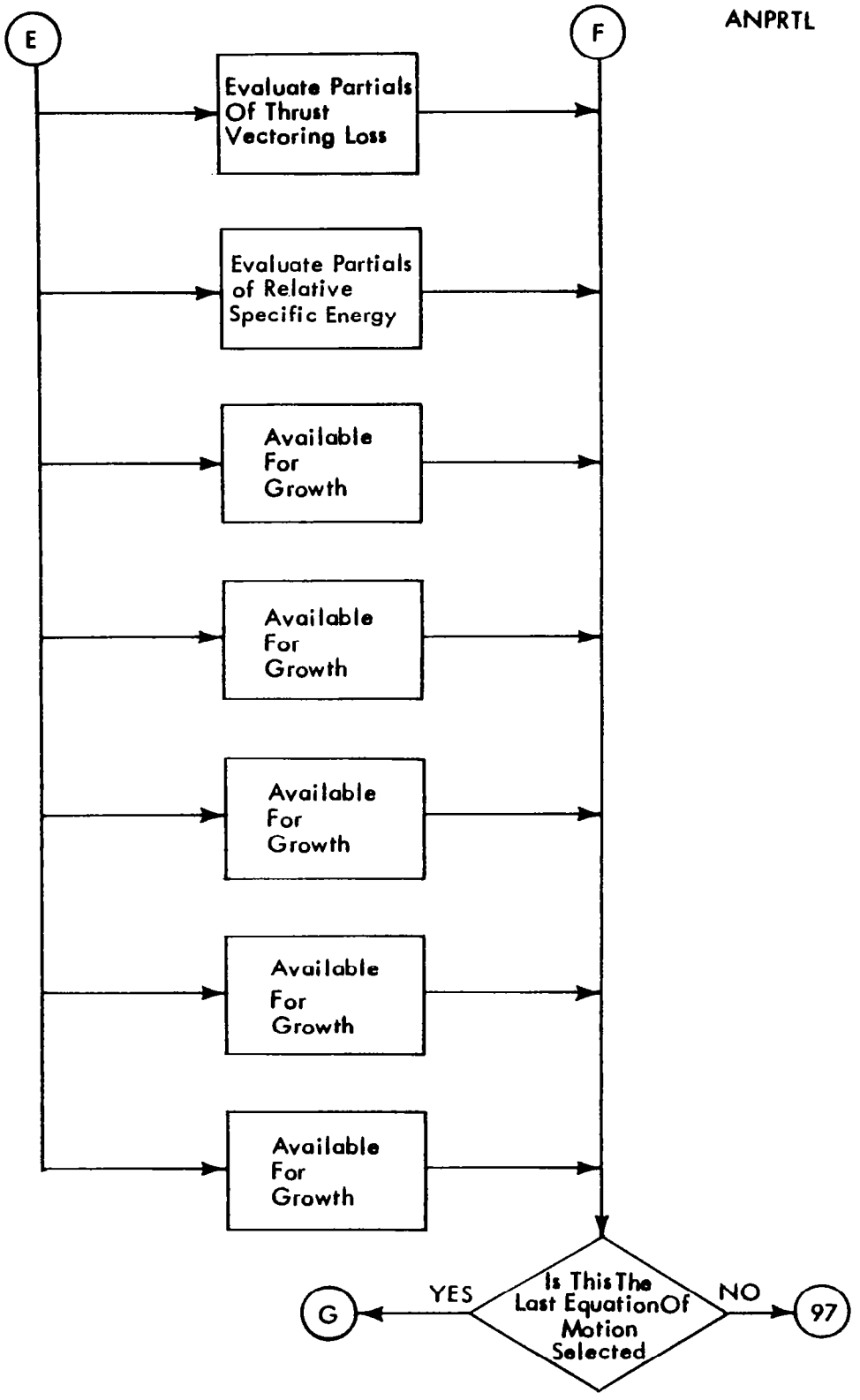
Storage Used

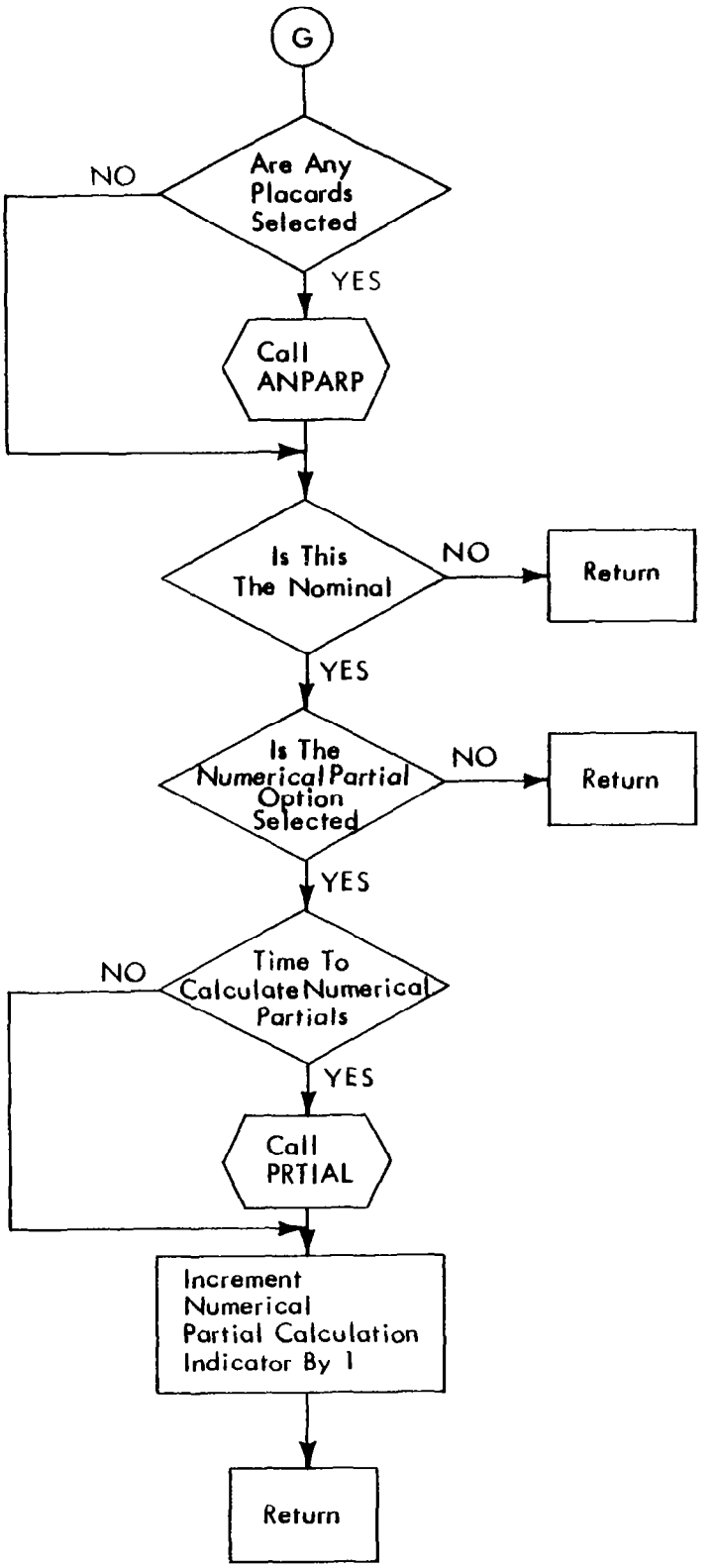
1,649 cells


Zero
PFX
PFU
Compute
QS
PQSPH
PQSPV
PLPH
PLPV
PLPA
Compute
PLPS
PDPH
PDPV
PDPA
PDPS
PTPH
PTPV
PTPT
Compute
PAPT
PAPG
PGPH
Evaluate Partials For Selected State Variables
97
Evaluate Partials Of Weight
Evaluate Partials Of Altitude
Evaluate Partials Of Relative Flight Path Angle
A
B


A
B
ANPRTL
Is Velocity Less Than 100.FT/SEC
YES
NO
Evaluate Partials Of Relative Velocity
Evaluate Partials Of Latitude Angle
Is The Latitude Within .1 Degree Of Either Pole
YES
NO
Is Gamma Within .1 Degree Of 90. Degrees
YES
NO
Is Velocity Less Than 100. FT/SEC
YES
NO
Evaluate Partials Of Relative Heading Angle
C
D


C
D
Is Latitude Within .1 Degree Of Either Pole
YES
NO
Evaluate Partials Of Longitude Angle
Dummy Time
Evaluate Partials Of Path Range Along Earth's Surface
Evaluate Partials Of Aerodynamic Heating Integral
Evaluate Partials Of Ideal Relative Delta Velocity
Evaluate Partials Of Gravity Loss
Evaluate Partials Of Drag Loss
E
F


E
F
Evaluate Partials Of Thrust Vectoring Loss
Evaluate Partials of Relative Specific Energy
Available For Growth
Available For Growth
Available For Growth
Available For Growth
Available For Growth
Is This The Last Equation Of Motion Selected
YES
NO
G
97


G
Are Any Placards Selected
NO
YES
Call ANPARP
Is This The Nominal
NO
Return
YES
Is The Numerical Partial Option Selected
NO
Return
YES
Time To Calculate Numerical Partials
NO
YES
Call PRTIAL
Increment Numerical Partial Calculation Indicator By 1
Return

ATMOS — Atmosphere Calculations

Purpose

ATMOS calculates atmospheric pressure, temperature, density, and speed of sound. Any atmosphere may be used; however, the calling sequence must be the same as presently used.

Method

The ATMOS subroutine normally used in the program is an analytical representation of the 1962 ARDC standard atmosphere.

The call to ATMOS is made with input and output variables transmitted through the calling sequence, i.e.,

CALL ATMOS (H, VA, P, D, T)

where H is the altitude, ft

VA is the speed of sound, fps

P is the pressure, psf

D is the density, slugs/ft^3

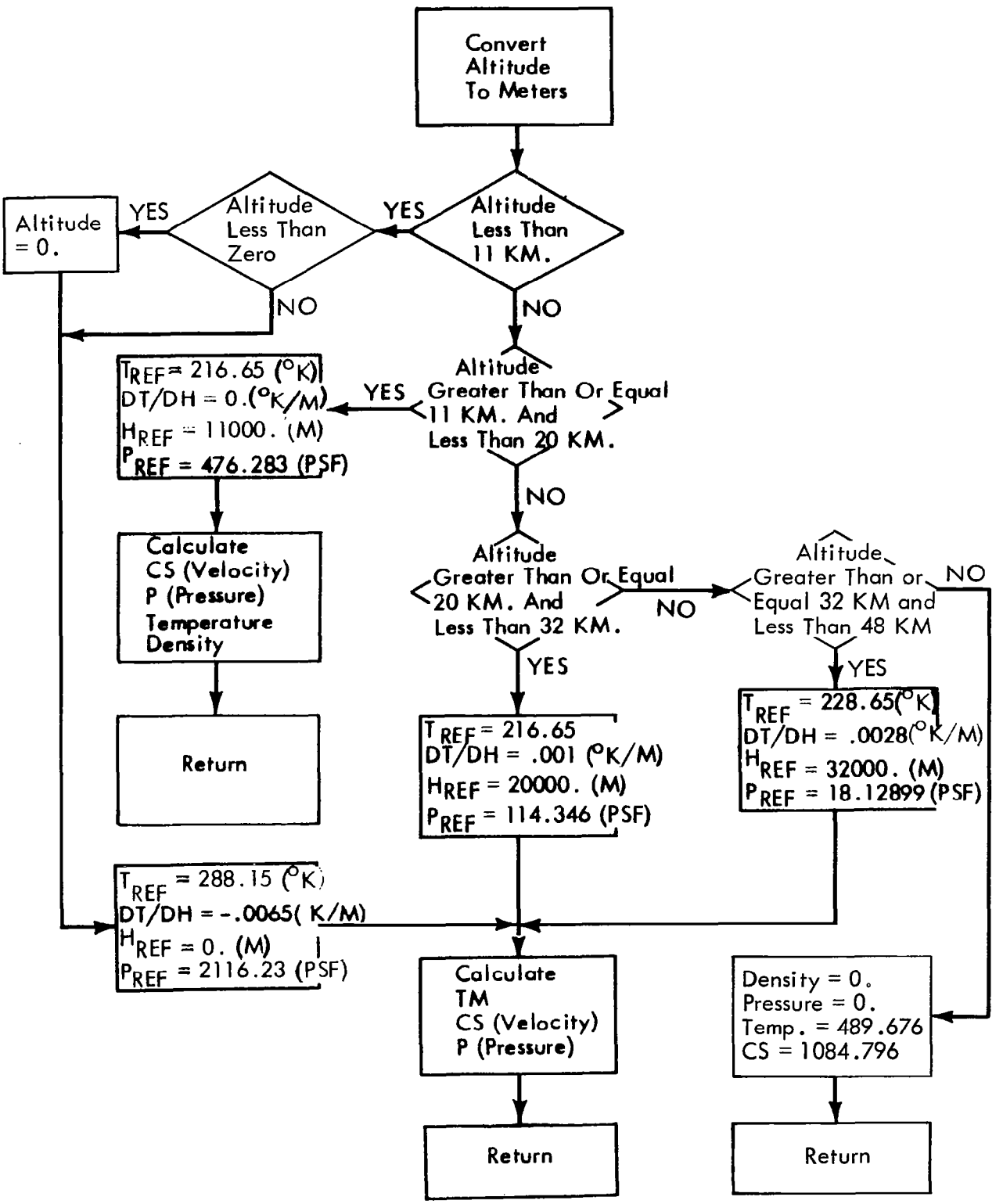
T is the temperature, deg Rankine

Limitation

This ATMOS subroutine is a modified version of the complete 1962 ARDC standard atmosphere. Because of a limiting altitude of 157,000 feet and other simplifications, the modified ATMOS is considerably faster than the complete atmosphere with little or no loss of accuracy within the bounds of common application. Other ATMOS subroutines are presented in reference 9.

Storage Used

250 cells


Convert Altitude To Meters
Altitude Less Than 11 KM.
YES
Altitude Less Than Zero
YES
Altitude = 0.
NO
NO
Altitude Greater Than Or Equal 11 KM. And Less Than 20 KM.
YES
T_{REF} = 216.65 (°K)
DT/DH = 0.(°K/M)
H_{REF} = 11000. (M)
P_{REF} = 476.283 (PSF)
NO
Calculate CS (Velocity) P (Pressure) Temperature Density
Return
Altitude Greater Than Or Equal 20 KM. And Less Than 32 KM.
NO
Altitude Greater Than or Equal 32 KM and Less Than 48 KM
NO
YES
T_{REF} = 228.65(°K)
DT/DH = .0028(°K/M)
H_{REF} = 32000. (M)
P_{REF} = 18.12899 (PSF)
YES
T_{REF} = 216.65
DT/DH = .001 (°K/M)
H_{REF} = 20000. (M)
P_{REF} = 114.346 (PSF)
T_{REF} = 288.15 (°K)
DT/DH = -.0065(K/M)
H_{REF} = 0. (M)
P_{REF} = 2116.23 (PSF)
Calculate TM CS (Velocity) P (Pressure)
Return
Density = 0.
Pressure = 0.
Temp. = 489.676
CS = 1084.796
Return

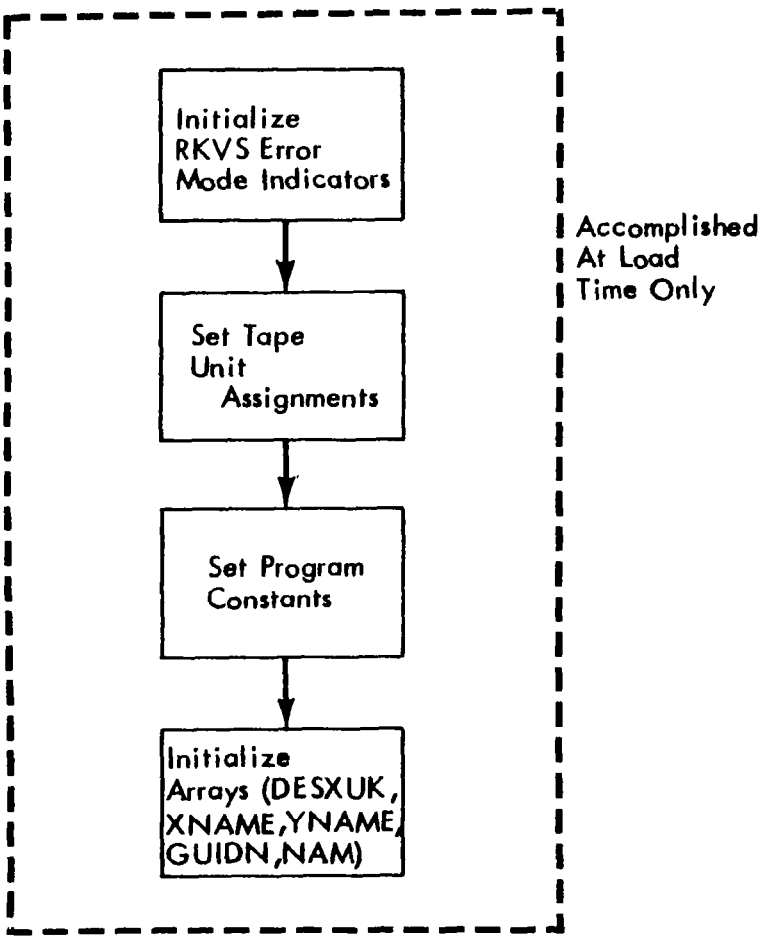
BLOCK — Program Constants and Output Titles

Purpose

This subroutine is provided for input of numerical constants and alphanumeric printout titles, which are required by the program and are not case dependent. These variables are set once for each computer run.

Method

All constants and titles are input by data statements.


Initialize
RKVS Error
Mode Indicators
Set Tape
Unit
Assignments
Set Program
Constants
Initialize
Arrays (DESXUK,
XNAME,YNAME,
GUIDN,NAM)
Accomplished
At Load
Time Only

BOOM — Sonic Boom Overpressure and Placard Determination

Purpose

BOOM calculates the sonic boom overpressure and determines the equation of motion for the overpressure violation in the path of the aircraft.

Method

Sonic boom overpressure on the ground is determined for the conditions along the flight path. The sonic boom penalty function is calculated by the method discussed in the analytical development section on enroute constraints. The terminal value of the penalty function (auxiliary state variable) must be constrained to zero. Location of the shock signature on the ground is determined as a function of the longitude, latitude, and aircraft flight conditions.

Assumptions and Limitations

The overpressure violation is calculated at an odd number of points so that both maximum lateral cutoff points and the midpoint are included. The number of points is selected by setting NC(75) = number of points. If NC(75) is even, the number of points calculated is NC(75) + 1. If too few points are selected, a region where the placard should be violated may slip through the mesh without being detected.

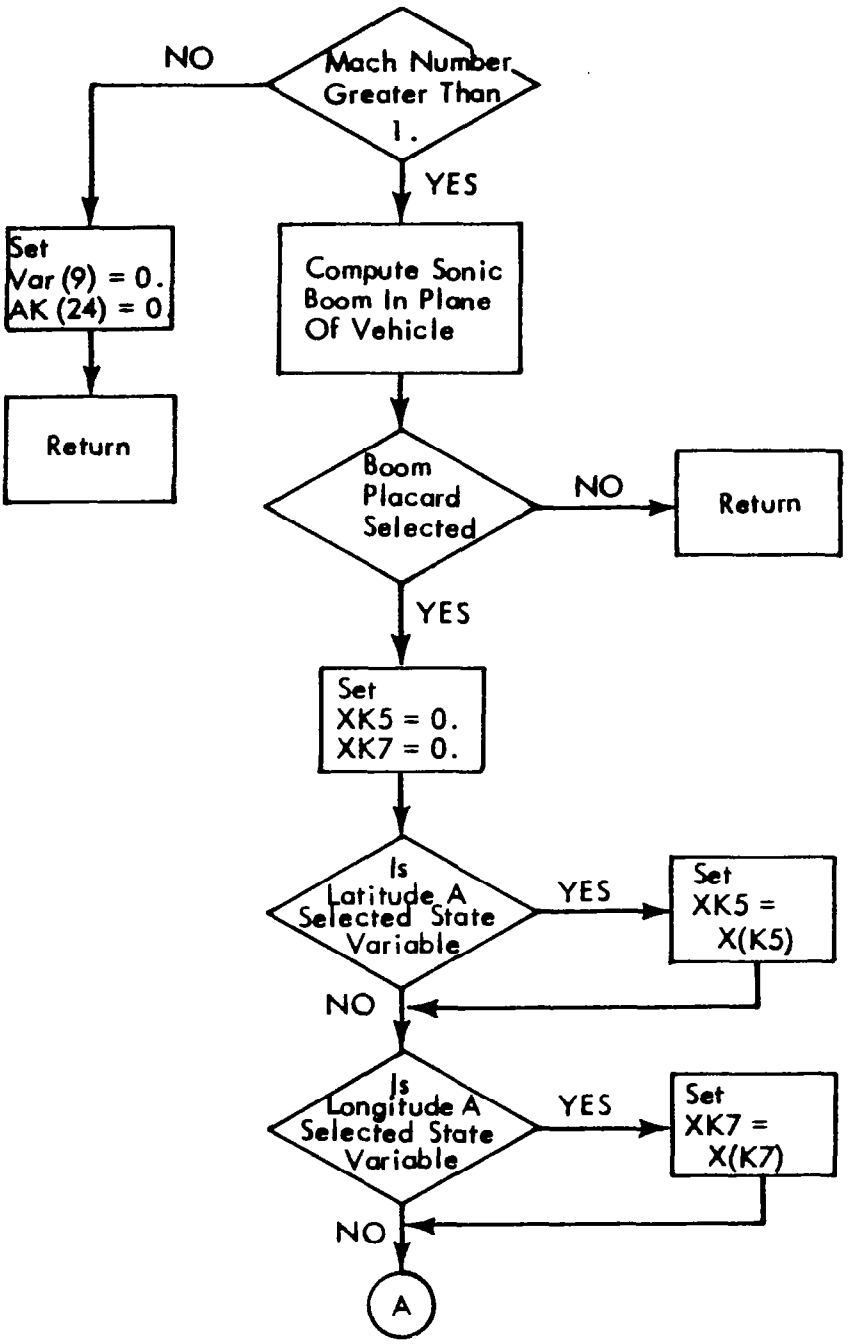
Subroutines Called

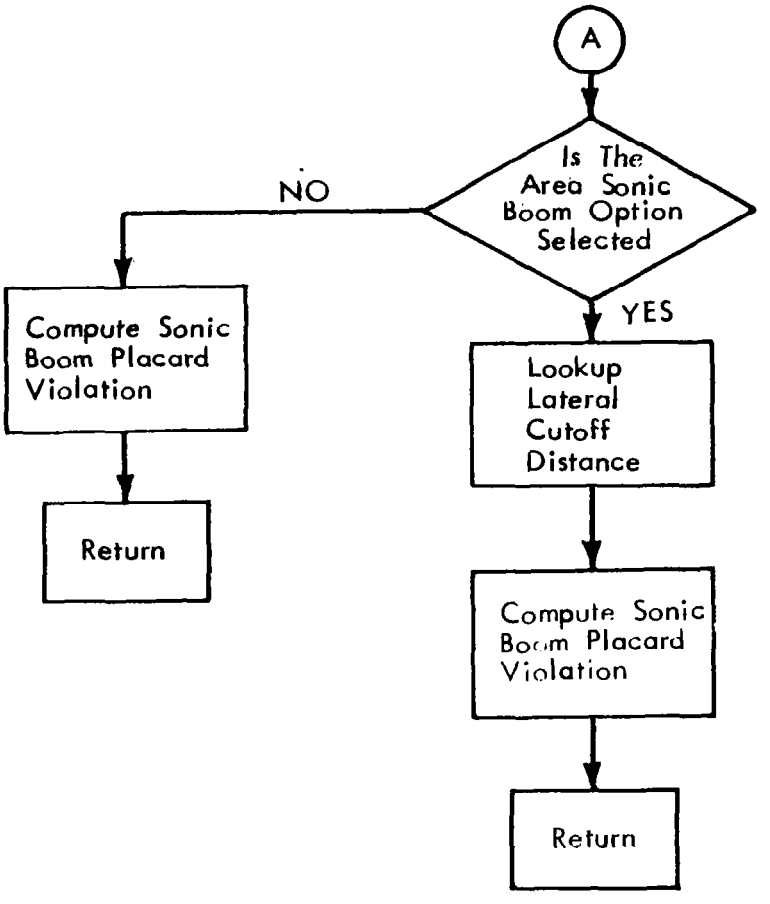
LOOK3D

PLAC

Storage Used

298 cells


Mach Number Greater Than 1.
NO
YES
Set Var (9) = 0. AK (24) = 0
Return
Compute Sonic Boom In Plane Of Vehicle
Boom Placard Selected
NO
Return
YES
Set XK5 = 0. XK7 = 0.
Is Latitude A Selected State Variable
YES
Set XK5 = X(K5)
NO
Is Longitude A Selected State Variable
YES
Set XK7 = X(K7)
NO
A


A
Is The Area Sonic Boom Option Selected
NO
YES
Compute Sonic Boom Placard Violation
Return
Lookup Lateral Cutoff Distance
Compute Sonic Boom Placard Violation
Return

CARDS — Card Output Routine

Purpose

CARDS is called at the end of each data case and is assigned the following tasks:

1) Recover control table from last valid trajectory from KSCR unit and print and punch control table and restart table;

2) Recover trajectory summary from KDAT unit and print trajectory summary table.

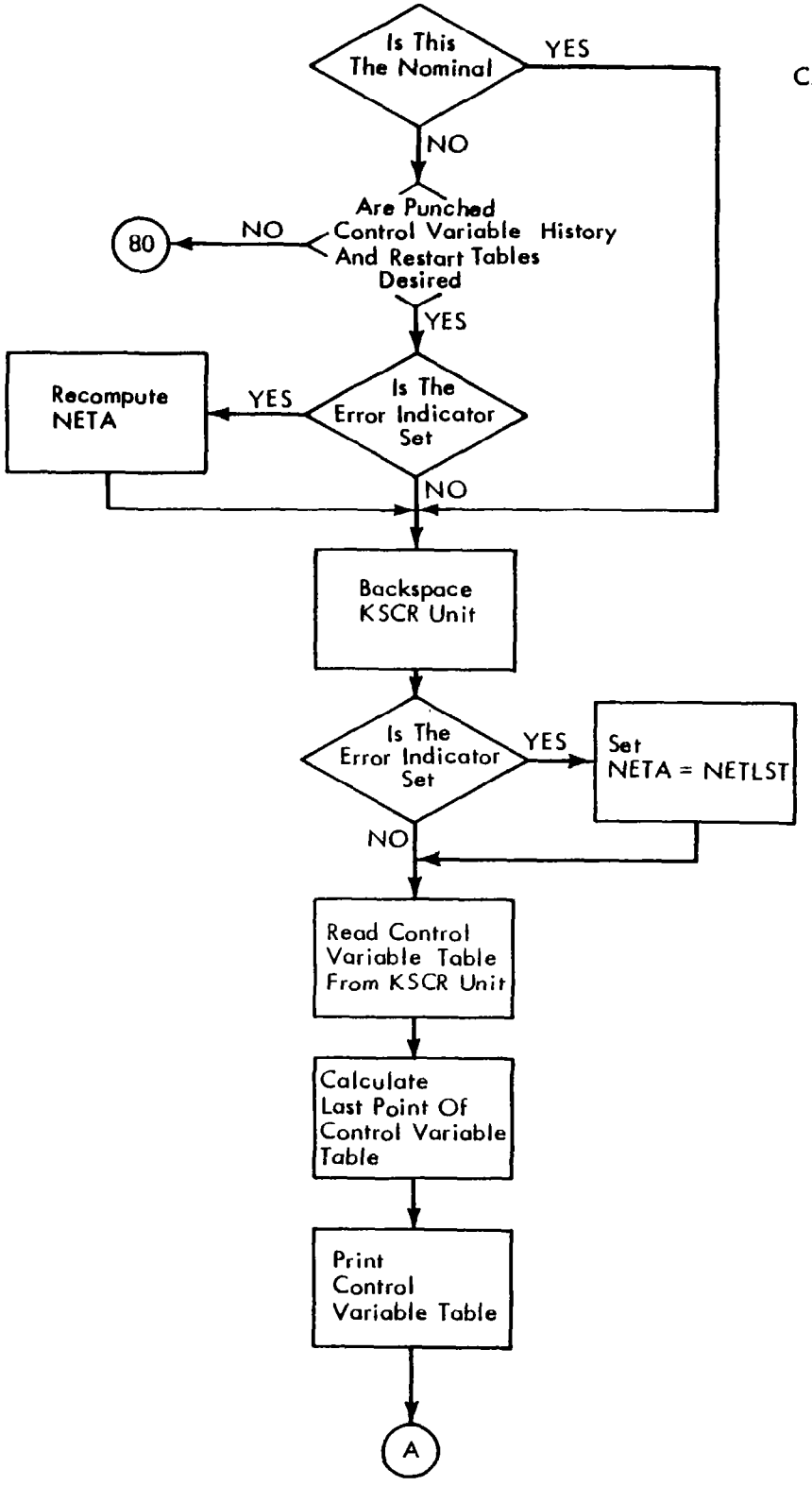
Method

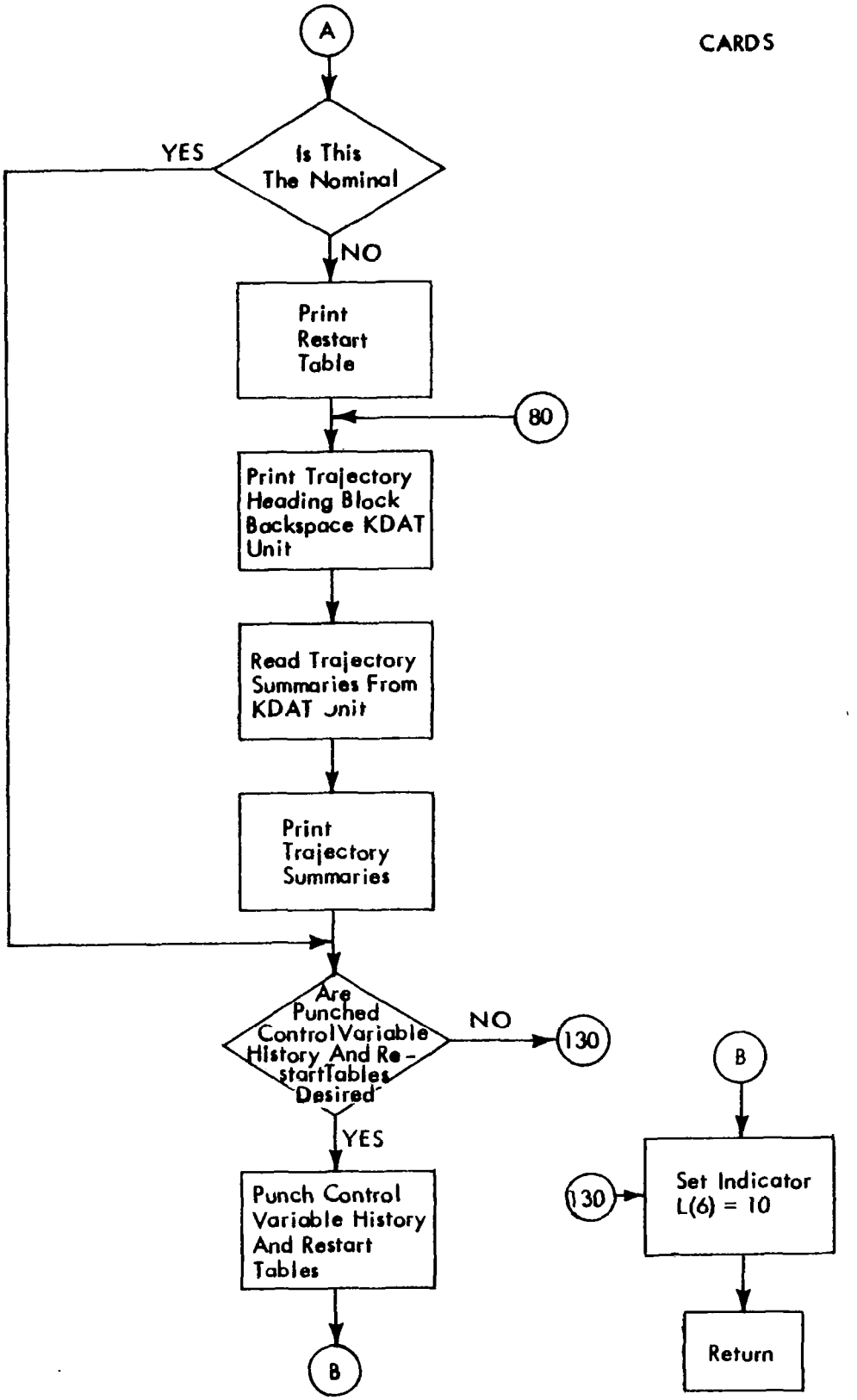
The control variable table is recovered by backspacing the KSCR unit to the start of the last valid trajectory and reading the control history into core. The restart table is built from convergence information in COMMON. The control table is then punched in octal format (in order to maintain sufficient accuracy) and the restart table is punched in decimal format (for ease in modification if required).

The trajectory summary is obtained from the KDAT unit which was written by LAMBDA at the end of each valid iteration.

Storage Used

681 cells


Is This The Nominal
YES
CARDS
NO
Are Punched Control Variable History And Restart Tables Desired
NO
80
YES
Is The Error Indicator Set
YES
Recompute NETA
NO
Backspace KSCR Unit
Is The Error Indicator Set
YES
Set NETA = NETLST
NO
Read Control Variable Table From KSCR Unit
Calculate Last Point Of Control Variable Table
Print Control Variable Table
A


A
CARDS
YES
Is This The Nominal
NO
Print Restart Table
80
Print Trajectory Heading Block Backspace KDAT Unit
Read Trajectory Summaries From KDAT Unit
Print Trajectory Summaries
Are Punched Control Variable History And Re-start Tables Desired
NO
130
YES
Punch Control Variable History And Restart Tables
B
B
130
Set Indicator L(6) = 10
Return

CONTR — Open Loop Control Variable Calculation

Purpose

CONTR performs a table lookup for the control vector ($\bar{u}$) in the TIMEU array at each new time point as required by the integration package.

Method

The control variables obtained are specified by the setting of NC's 77 through 80 for θ, φ, η, and Λ respectively. An NC set to 0 indicates the control variable is not required, set to 1 indicates the control variable is used. Values of the variables as functions of time are obtained by linear interpolation from the control variable table which is either input or generated by the program. For the nominal trajectory, if NC(13) = 0, the control variables are calculated as described above; but if NC(13) $\neq$ 0, the control variables θ or η are calculated in the GUIDE subroutine and override the value looked up by CONTR. CONTR is called prior to GUIDE.

Assumptions and Limitations

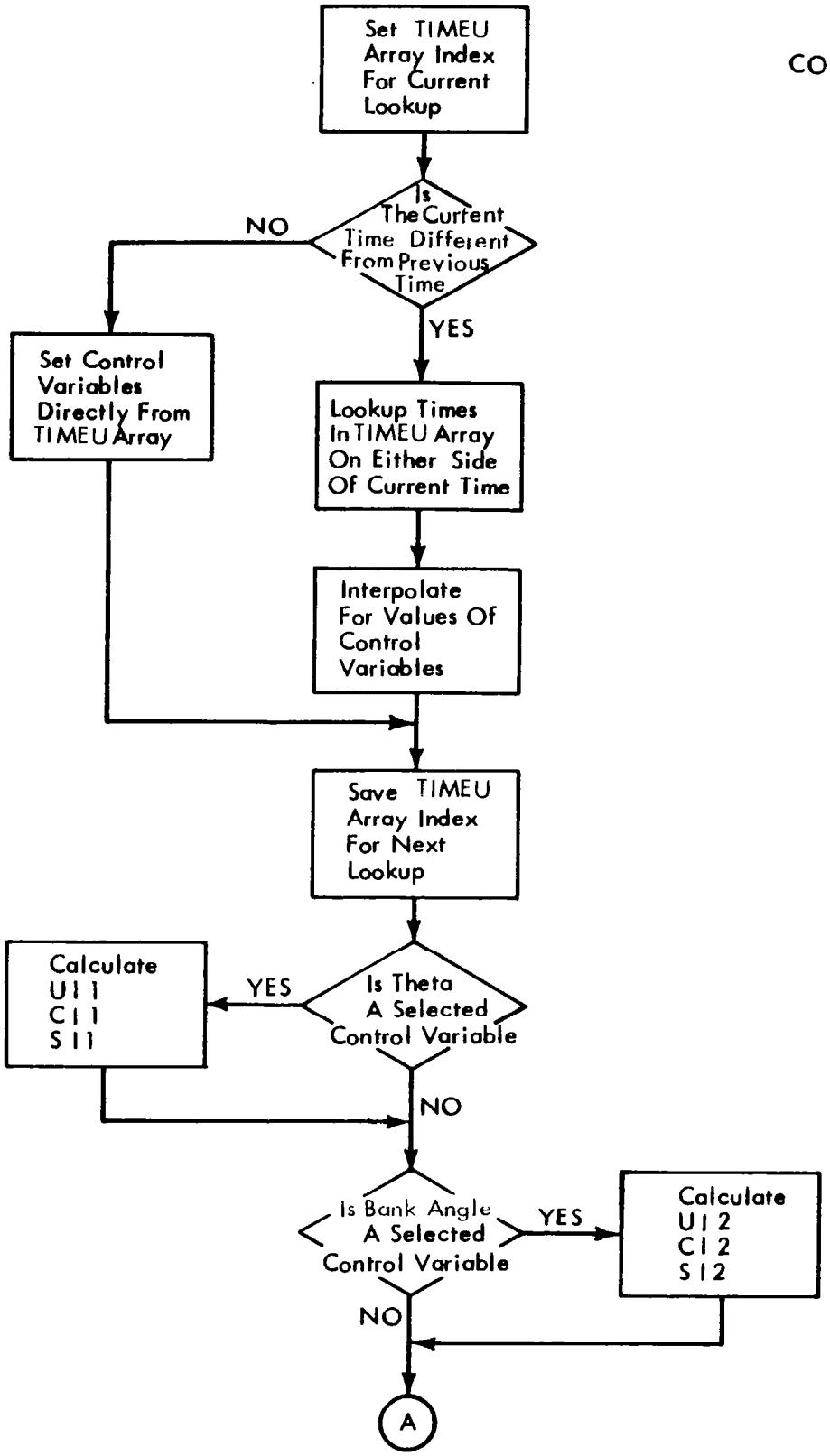
The control variables are obtained by a linear table look-up. The table look-up package does not extrapolate so the user must input data that will not permit a table overrun.

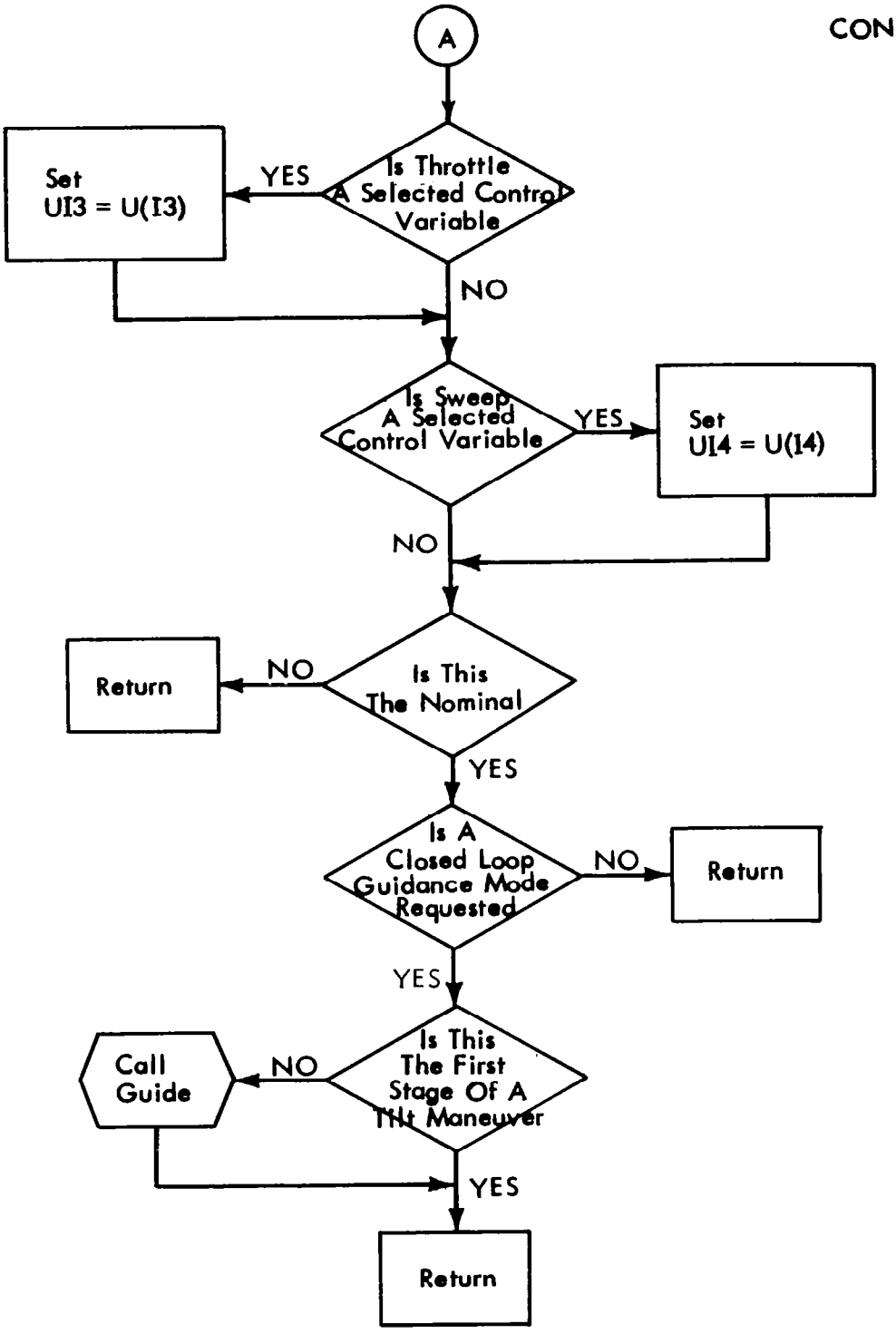
Subroutines Called

GUIDE

Storage Used

197 cells


Set TIMEU Array Index For Current Lookup
Is The Current Time Different From Previous Time
NO
YES
Set Control Variables Directly From TIMEU Array
Lookup Times In TIMEU Array On Either Side Of Current Time
Interpolate For Values Of Control Variables
Save TIMEU Array Index For Next Lookup
Is Theta A Selected Control Variable
YES
Calculate U11 C11 S11
NO
Is Bank Angle A Selected Control Variable
YES
Calculate U12 C12 S12
NO
A


A
Is Throttle A Selected Control Variable
YES
Set UI3 = U(I3)
NO
Is Sweep A Selected Control Variable
YES
Set UI4 = U(I4)
NO
Is This The Nominal
NO
Return
YES
Is A Closed Loop Guidance Mode Requested
NO
Return
YES
Is This The First Stage Of A Tilt Maneuver
NO
Call Guide
YES
Return

DVAL2 — Adjoint Variable Derivative Evaluation

Purpose

The derivatives of the adjoint variables and the integrated payoff sensitivities are evaluated for the reverse integration.

Method

The differential equations for the adjoint variables are evaluated as discussed in the section on the steepest-ascent method. The partial derivative PFX or PFU and the control variables $\bar{u}$ were stored as functions of time during the forward trajectory. Because this subroutine is important for the basic understanding of the computer evaluations of the adjoint variables, weighting matrices, and I matrix, the computer language for important variables will be given.

The array of all derivatives evaluated during the backward integration, the FB array, is constructed with elements for $\dot{\Lambda}$ given by

$$\dot{\Lambda} = - F'\Lambda$$

where

$F \equiv$ PFX matrix

$\Lambda \equiv$ XB array

The derivatives for the elements of the I matrix are

$$\frac{dI}{dt} = \text{XXDOT array}$$

$$= \Lambda' \, G \, W^{-1} \, G' \, \Lambda \text{ (from equation 109)}$$

where

$\Lambda \equiv$ XXLAM matrix (given by equation 110)

$G \equiv$ PFU matrix

$W^{-1} \equiv$ VINP array

The derivatives of the elements for the automatic weighting are evaluated by

$$\frac{dS_u^{\varphi}}{dt} = \left| s_u^{\varphi} \right| \text{ (from equations 98 and 134)}$$

where

$s_u^{\varphi}(t) \equiv G' \lambda_{\varphi\Omega}$ (equation 130)

$\equiv$ PFU (I, J)* XXLAM (I, 1)

The elements of the UULAM matrix are the weighted impulse responses of the constraints and performance to the control variables as given by $W^{-1} G' \Lambda$.

The call to DVAL is made through the calling statement.

CALL DVAL (UULAM)

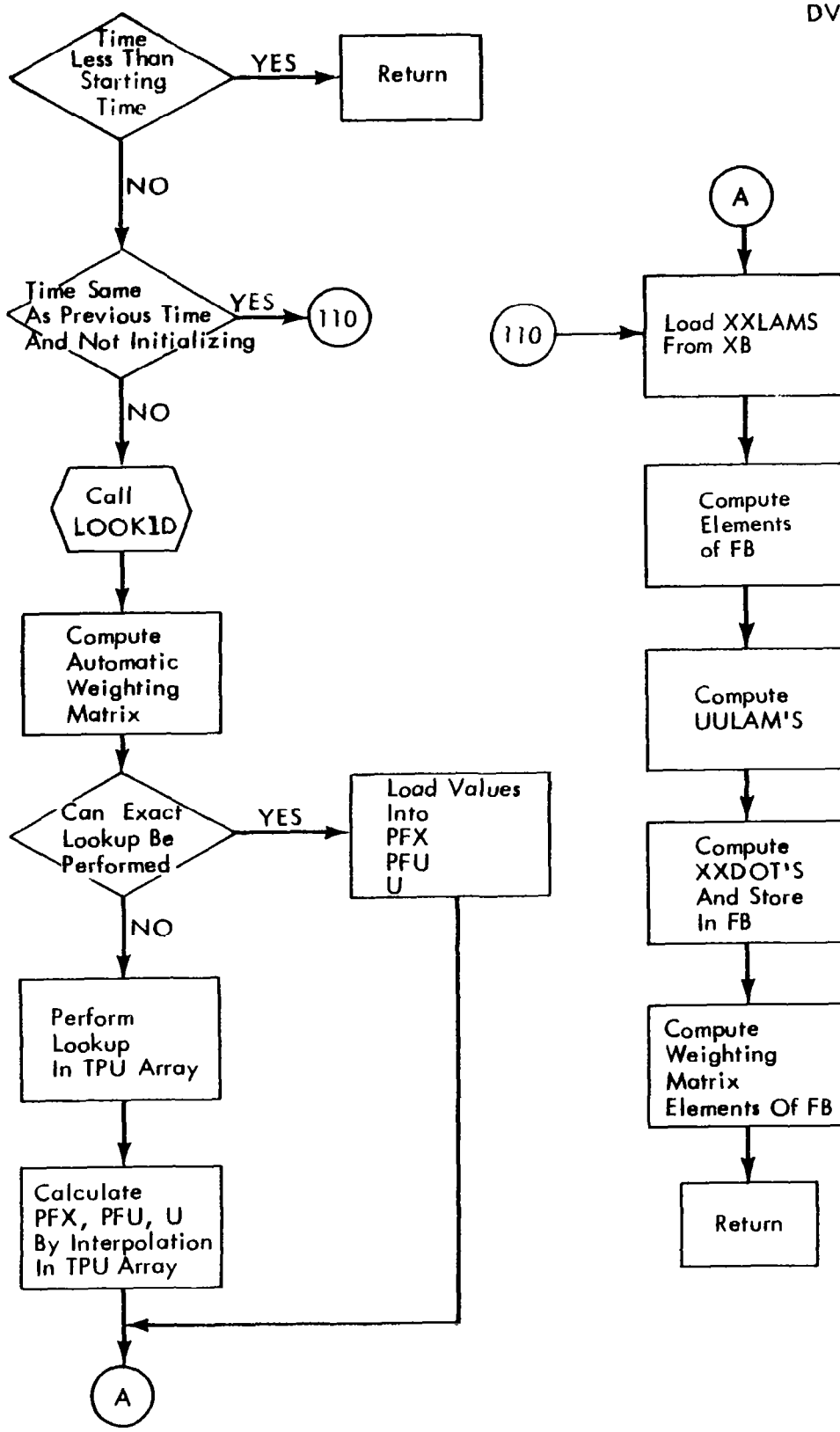
Note: When a single-dimension array is used to store the elements of a matrix, the elements of the matrix are stored by rows.

Subroutines Called

LOOK1D

Storage Used

1229 cells


Time Less Than Starting Time
YES
Return
NO
Time Same As Previous Time And Not Initializing
YES
110
NO
Call LOOK1D
Compute Automatic Weighting Matrix
Can Exact Lookup Be Performed
YES
Load Values Into PFX PFU U
NO
Perform Lookup In TPU Array
Calculate PFX, PFU, U By Interpolation In TPU Array
A
A
110
Load XXLAMS From XB
Compute Elements of FB
Compute UULAM'S
Compute XXDOT'S And Store In FB
Compute Weighting Matrix Elements Of FB
Return

EXEC — Forward Integration Flow

Purpose

EXEC is called at the start of every forward trajectory. The basic purpose is to control the program flow in performing the forward trajectory. To accomplish this, the following tasks are accomplished.

1) Read nonstage-dependent data from KDAT;

2) Initialize state variable array;

3) Read stage-dependent data from KDAT;

4) Position KDAT past last written trajectory summary at start of last stage only;

5) Initialize the state variable derivative array;

6) Integrate to end of stage or XSTP for last stage. Control data overlay is accomplished as required by reading blocks of data from KTAN.

Method

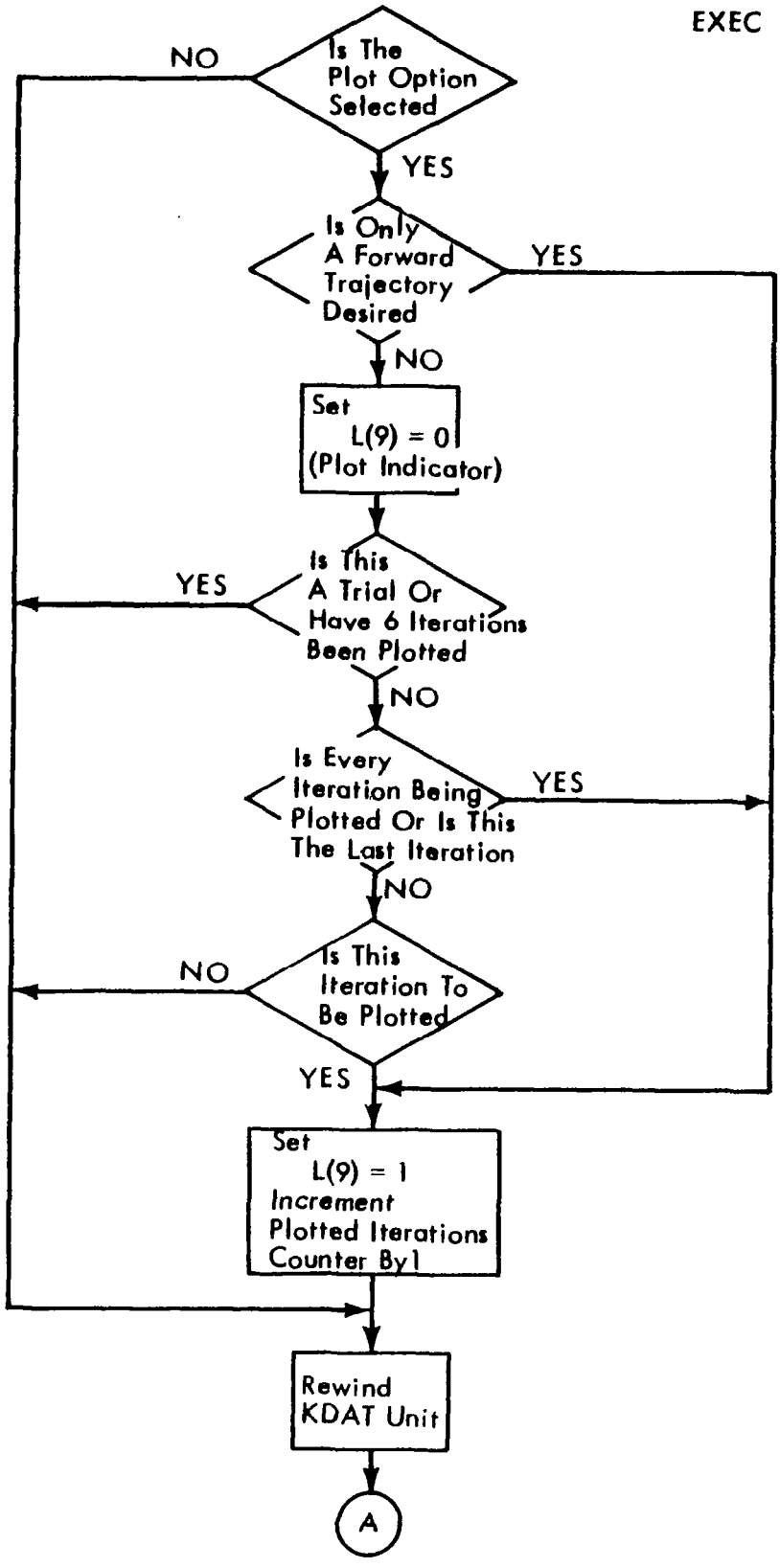
Integration of the forward trajectory utilizes the variable step Runge-Kutta option of STEP1. The final step to XSTP is iterated using the fixed step option until X(NST) is within EPSLN of XSTP.

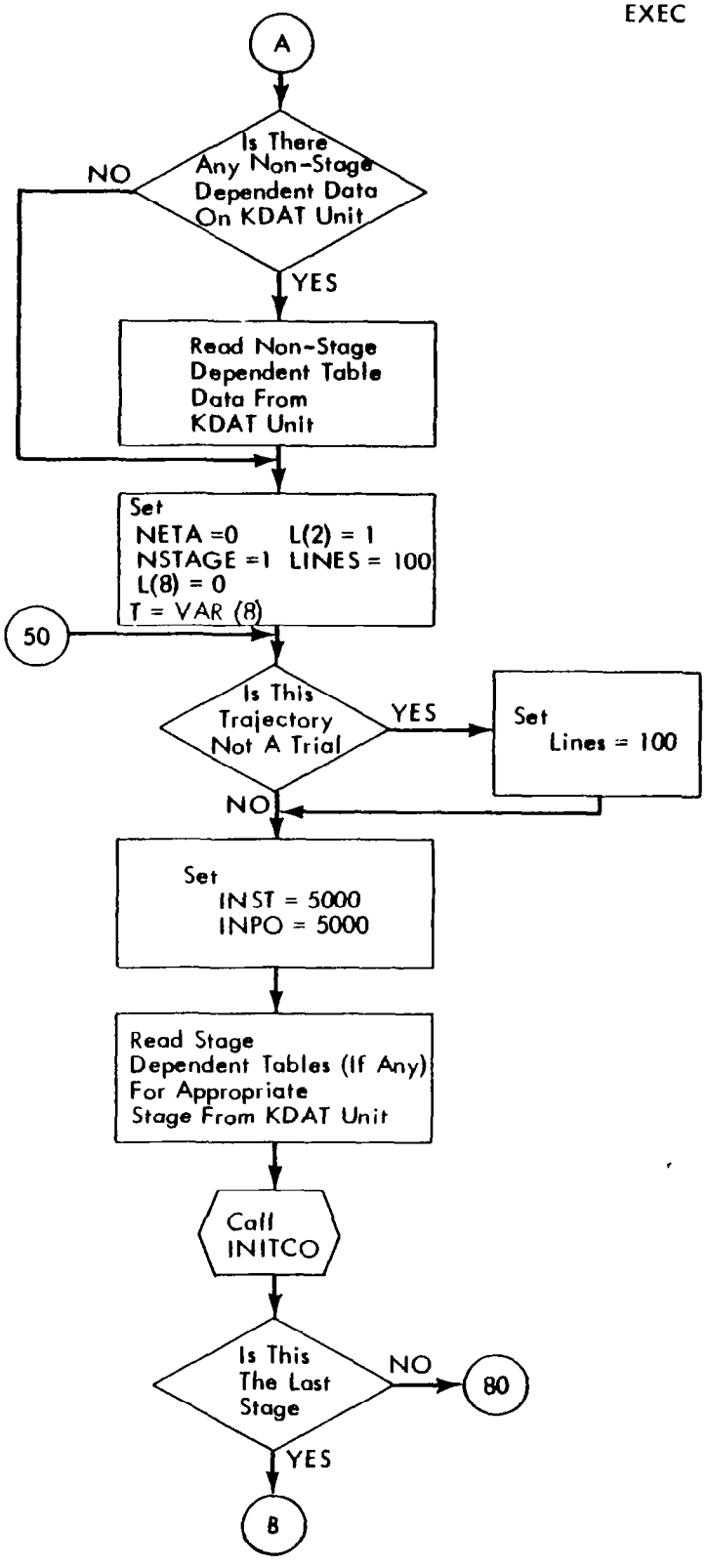
Subroutines Called

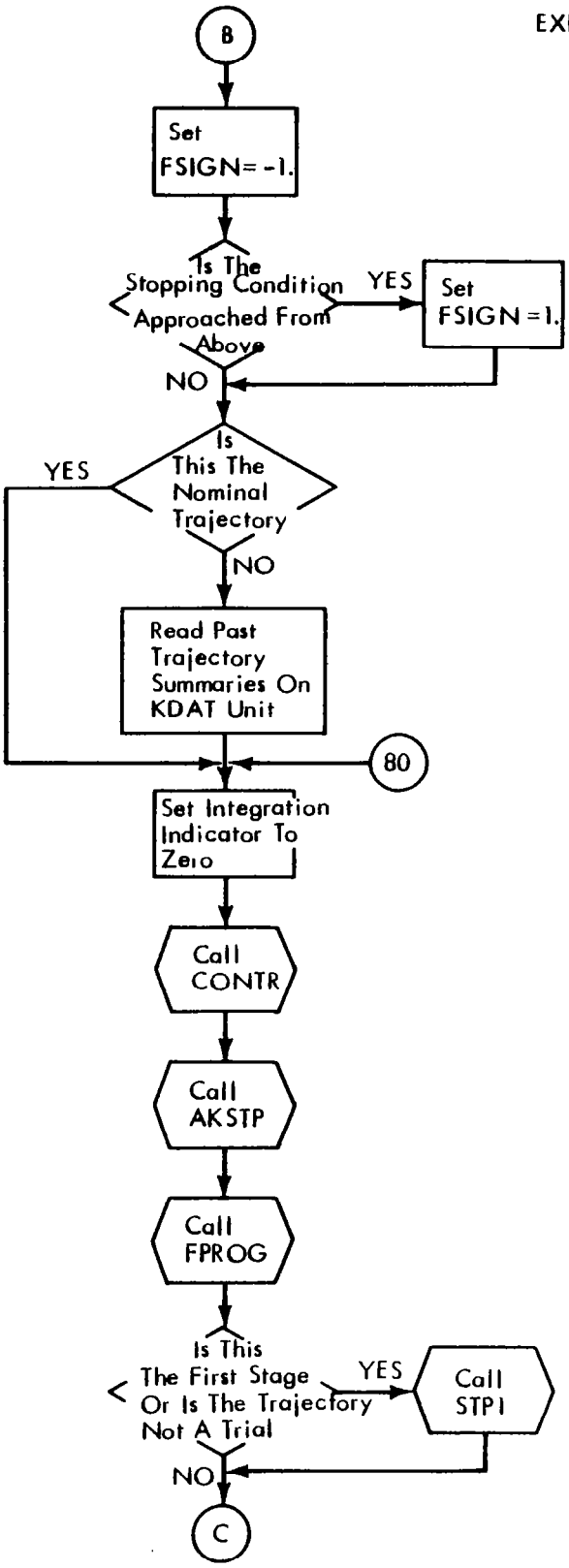
AKSTP	FPROG	STEP1
CONTR	INITCO	STP1

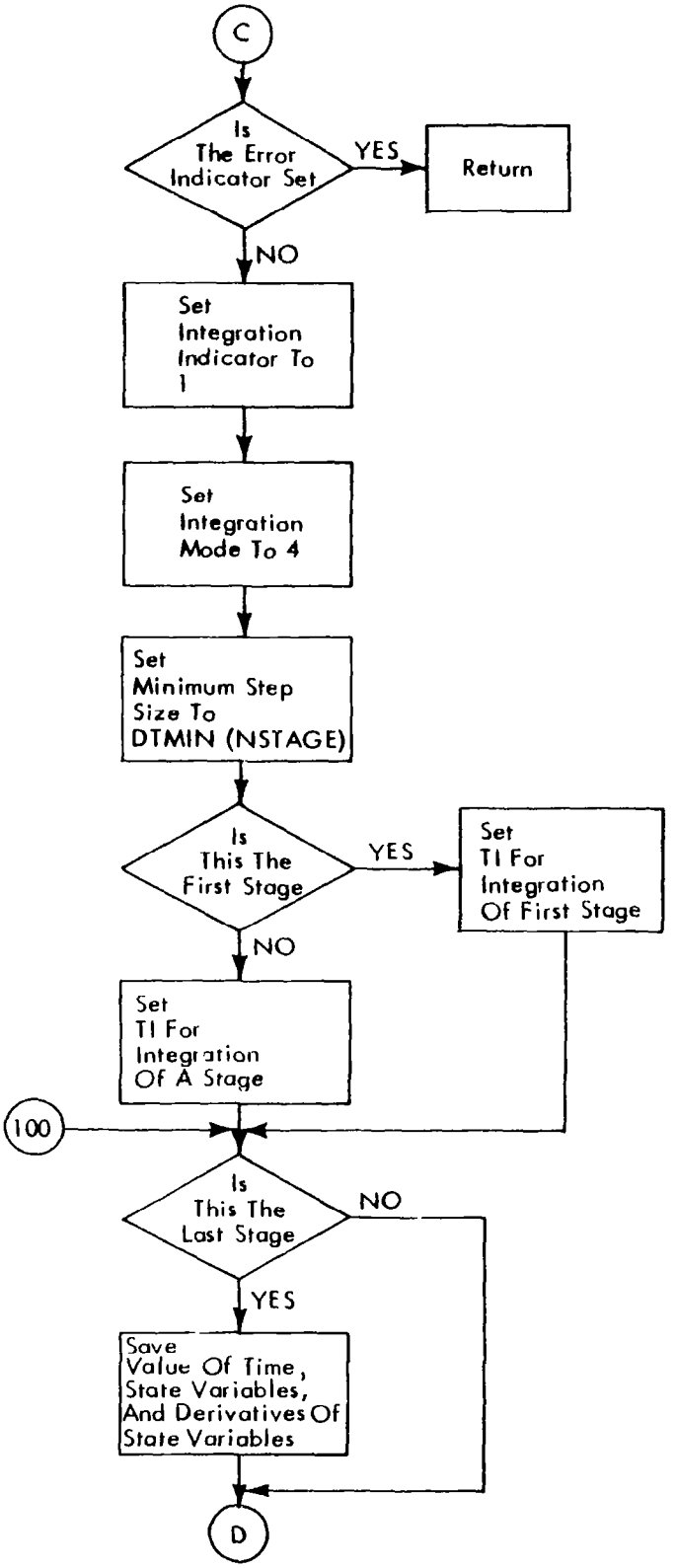
Storage Used

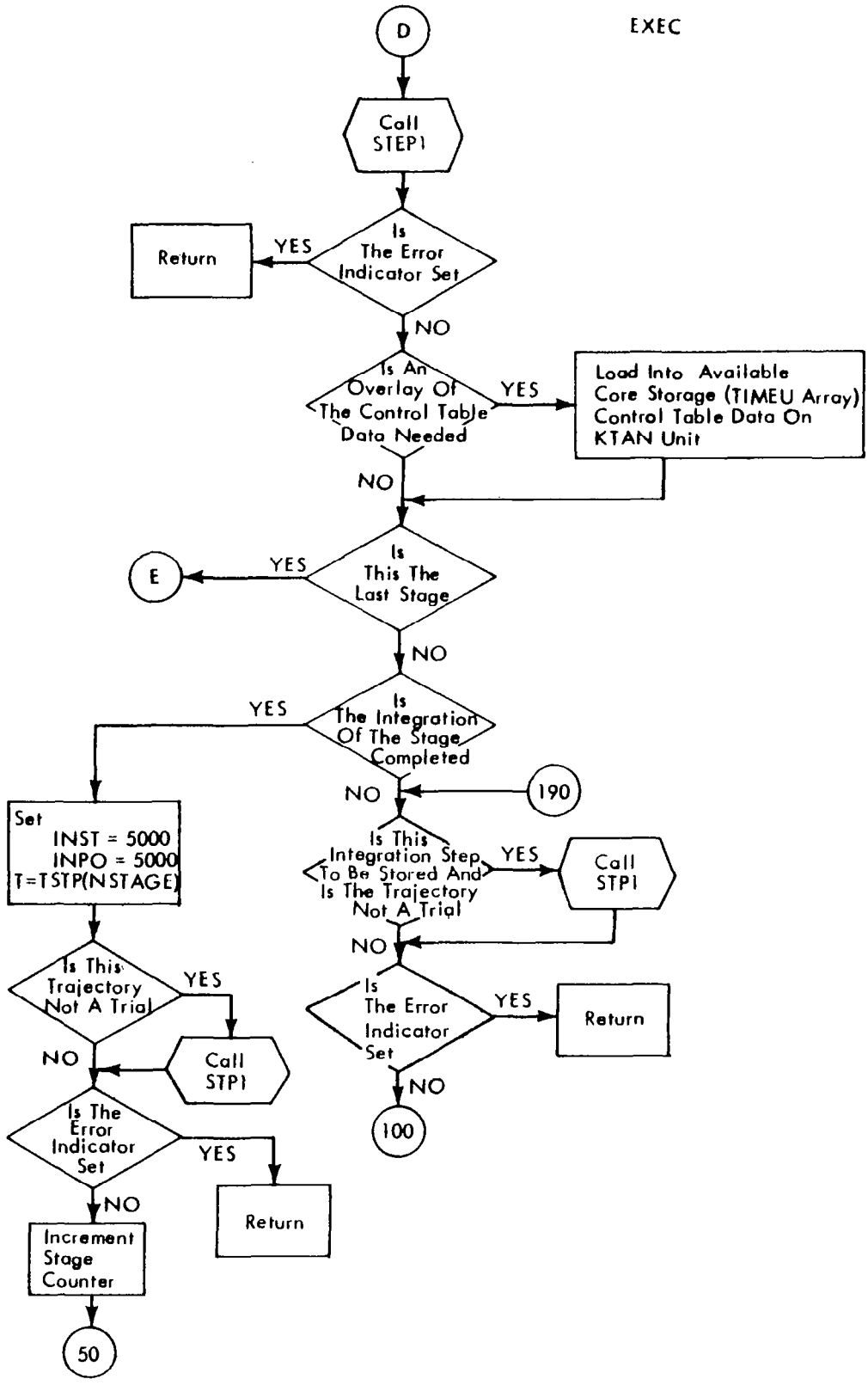
665 cells

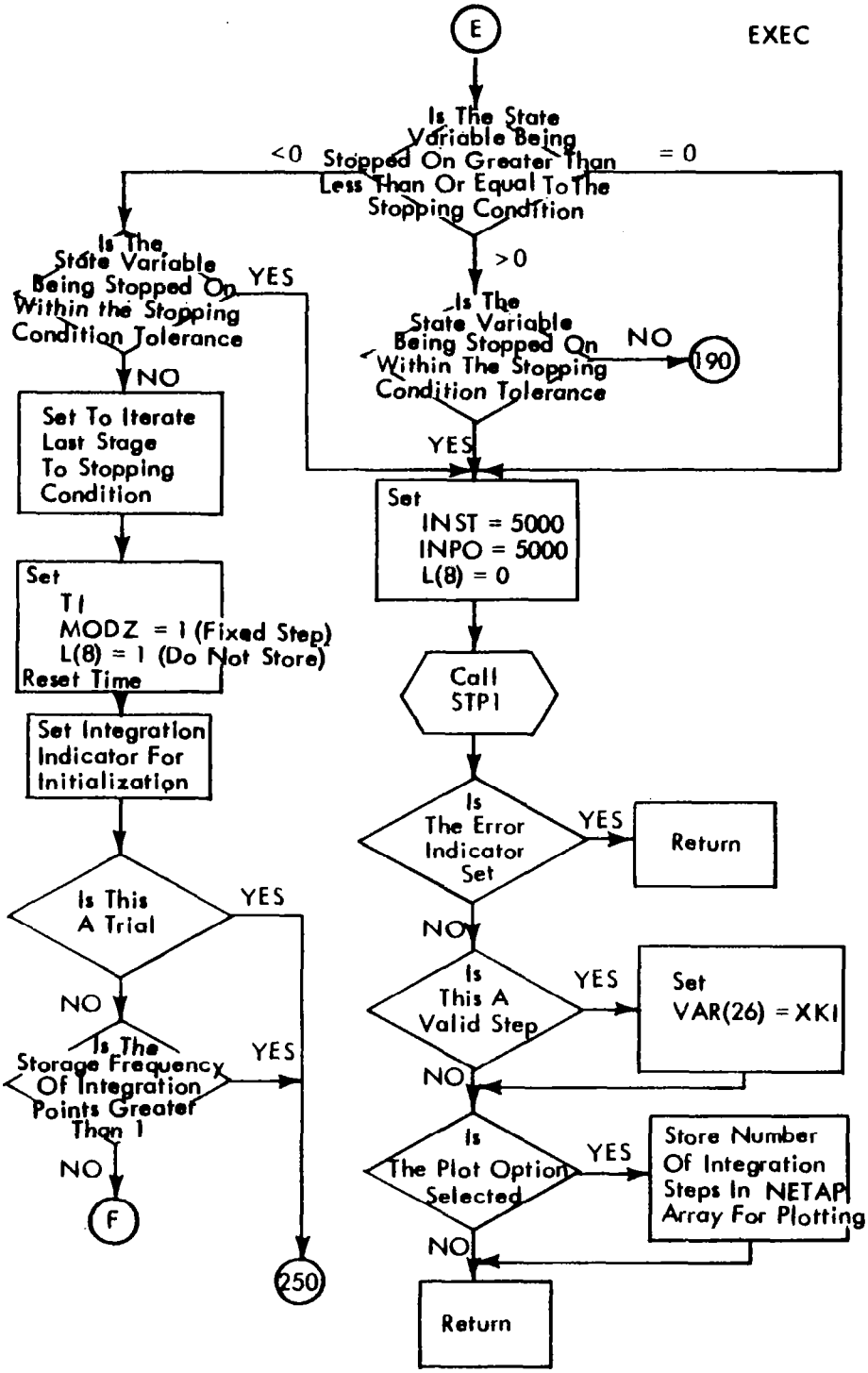

Is The Plot Option Selected
NO
YES
Is Only A Forward Trajectory Desired
YES
NO
Set L(9) = 0 (Plot Indicator)
Is This A Trial Or Have 6 Iterations Been Plotted
YES
NO
Is Every Iteration Being Plotted Or Is This The Last Iteration
YES
NO
Is This Iteration To Be Plotted
NO
YES
Set L(9) = 1 Increment Plotted Iterations Counter By1
Rewind KDAT Unit
A

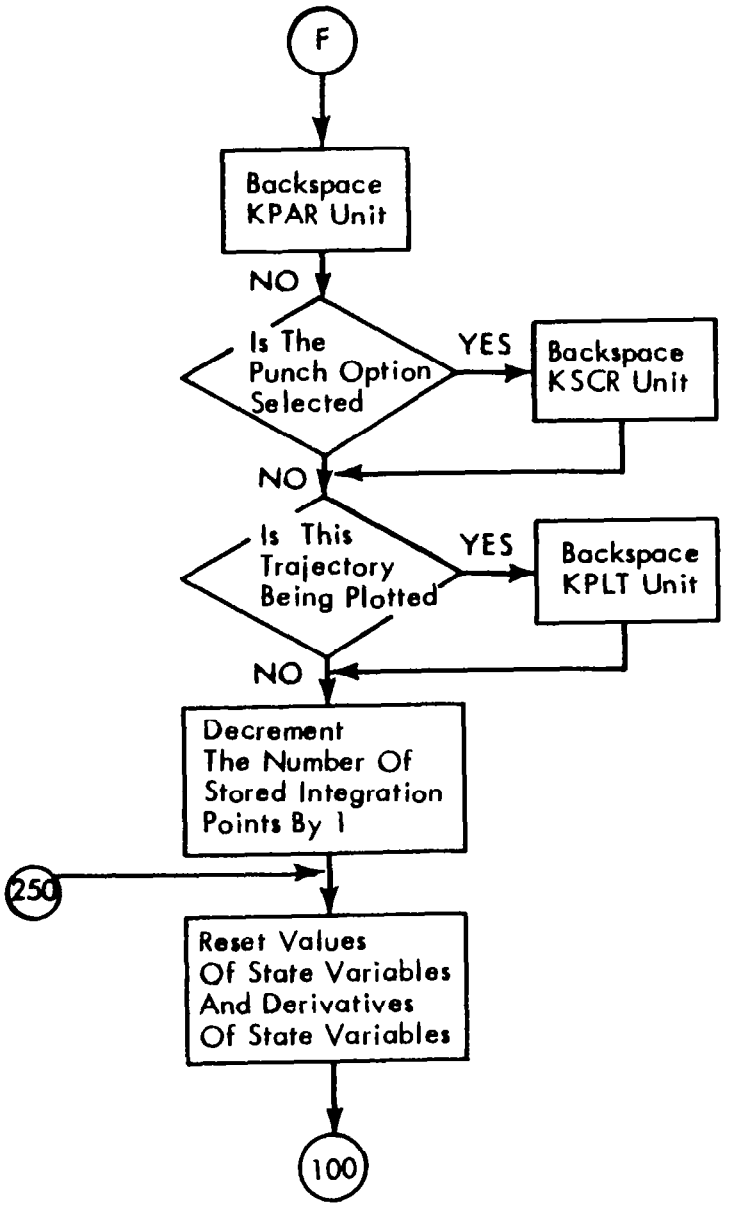

A
Is There Any Non-Stage Dependent Data On KDAT Unit
NO
YES
Read Non-Stage Dependent Table Data From KDAT Unit
Set
NETA =0 L(2) = 1
NSTAGE =1 LINES = 100
L(8) = 0
T = VAR (8)
50
Is This Trajectory Not A Trial
YES
Set
Lines = 100
NO
Set
INST = 5000
INPO = 5000
Read Stage Dependent Tables (If Any) For Appropriate Stage From KDAT Unit
Call INITCO
Is This The Last Stage
NO
80
YES
8


B
Set FSIGN= -1.
Is The Stopping Condition Approached From Above
YES
Set FSIGN =1.
NO
Is This The Nominal Trajectory
YES
NO
Read Past Trajectory Summaries On KDAT Unit
80
Set Integration Indicator To Zero
Call CONTR
Call AKSTP
Call FPROG
Is This The First Stage Or Is The Trajectory Not A Trial
YES
Call STP1
NO
C


C
Is The Error Indicator Set
YES
Return
NO
Set Integration Indicator To 1
Set Integration Mode To 4
Set Minimum Step Size To DTMIN (NSTAGE)
Is This The First Stage
YES
Set TI For Integration Of First Stage
NO
Set TI For Integration Of A Stage
100
Is This The Last Stage
NO
YES
Save Value Of Time, State Variables, And Derivatives Of State Variables
D


D
Call STEP1
Is The Error Indicator Set
YES
Return
NO
Is An Overlay Of The Control Table Data Needed
YES
Load Into Available Core Storage (TIMEU Array) Control Table Data On KTAN Unit
NO
Is This The Last Stage
YES
E
NO
Is The Integration Of The Stage Completed
YES
NO
190
Set INST = 5000 INPO = 5000 T=TSTP(NSTAGE)
Is This Integration Step To Be Stored And Is The Trajectory Not A Trial
YES
Call STP1
NO
Is This Trajectory Not A Trial
YES
Call STP1
NO
Is The Error Indicator Set
YES
Return
NO
100
Is The Error Indicator Set
YES
Return
NO
Increment Stage Counter
50


E
EXEC
Is The State Variable Being Stopped On Greater Than Less Than Or Equal To The Stopping Condition
<0
=0
>0
Is The State Variable Being Stopped On Within the Stopping Condition Tolerance
YES
NO
Set To Iterate Last Stage To Stopping Condition
Set TI MODZ = 1 (Fixed Step) L(8) = 1 (Do Not Store) Reset Time
Set Integration Indicator For Initialization
Is This A Trial
YES
NO
Is The Storage Frequency Of Integration Points Greater Than 1
YES
NO
F
250
Is The State Variable Being Stopped On Within The Stopping Condition Tolerance
NO
190
YES
Set INST = 5000 INPO = 5000 L(8) = 0
Call STP1
Is The Error Indicator Set
YES
Return
NO
Is This A Valid Step
YES
Set VAR(26) = XKI
NO
Is The Plot Option Selected
YES
Store Number Of Integration Steps In NETAP Array For Plotting
NO
Return


F
Backspace KPAR Unit
NO
Is The Punch Option Selected
YES
Backspace KSCR Unit
NO
Is This Trajectory Being Plotted
YES
Backspace KPLT Unit
NO
Decrement The Number Of Stored Integration Points By 1
250
Reset Values Of State Variables And Derivatives Of State Variables
100

FPROG — Equations of Motion (Calculation of F Array)

Purpose

FPROG calculates the entire F array, which consists of the equations defining the derivatives of the state variables and placards with respect to time. See appendix B for an algebraic definition of the F array.

Method

1) All state variable derivatives with respect to time are calculated directly in FPROG. The placard derivatives, however, are calculated in the PLAC subroutine called by FPROG.

2) The value of the placard required in the evaluation of the placard derivatives is obtained by a table look-up.

3) The array is initially zeroed so that only the calculations performed in a given case may have nonzero values.

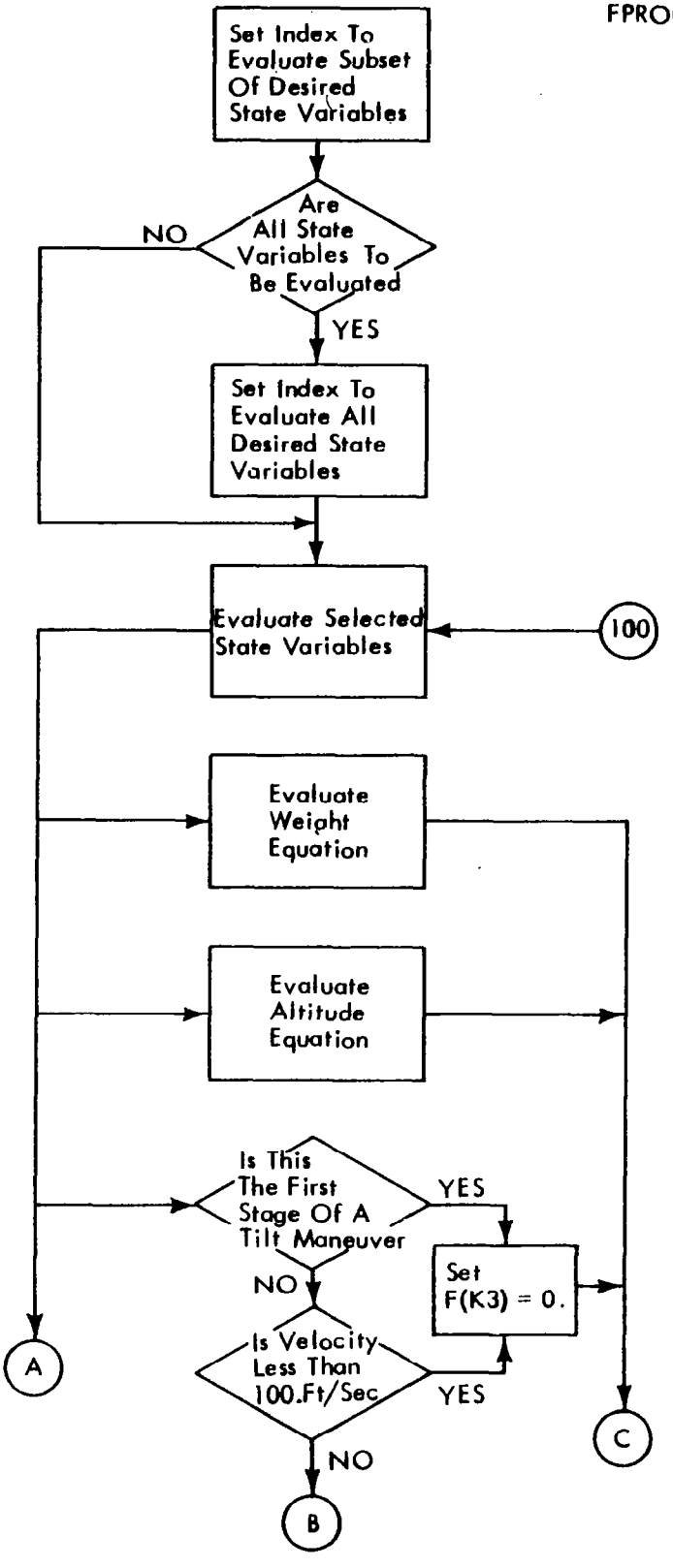
Note: The sonic boom placard, F(K35) is not determined in FPROG but by the subroutine AKSTP (which calls BOOM which, in turn, calls PLAC).

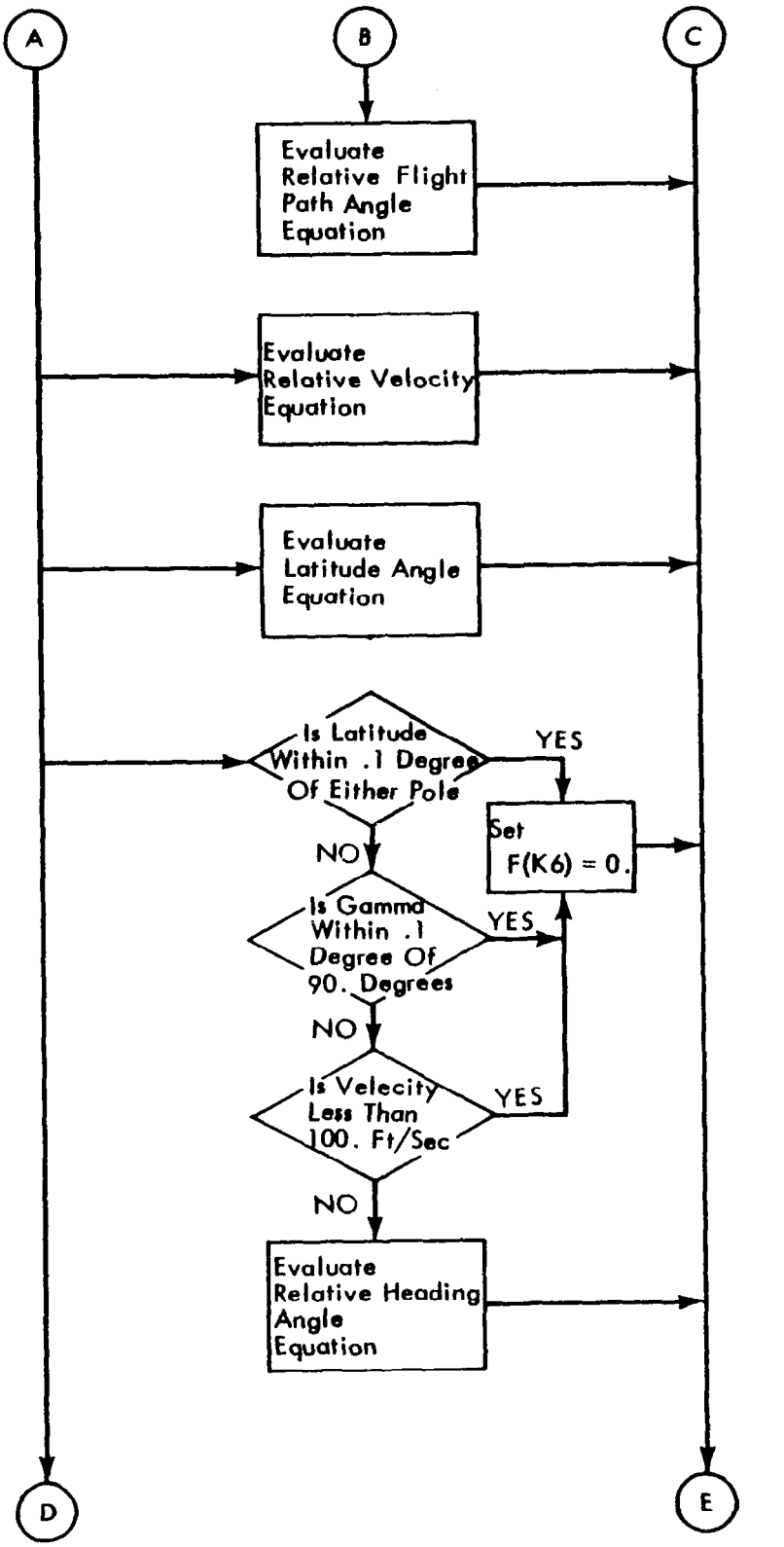
Subroutines Called

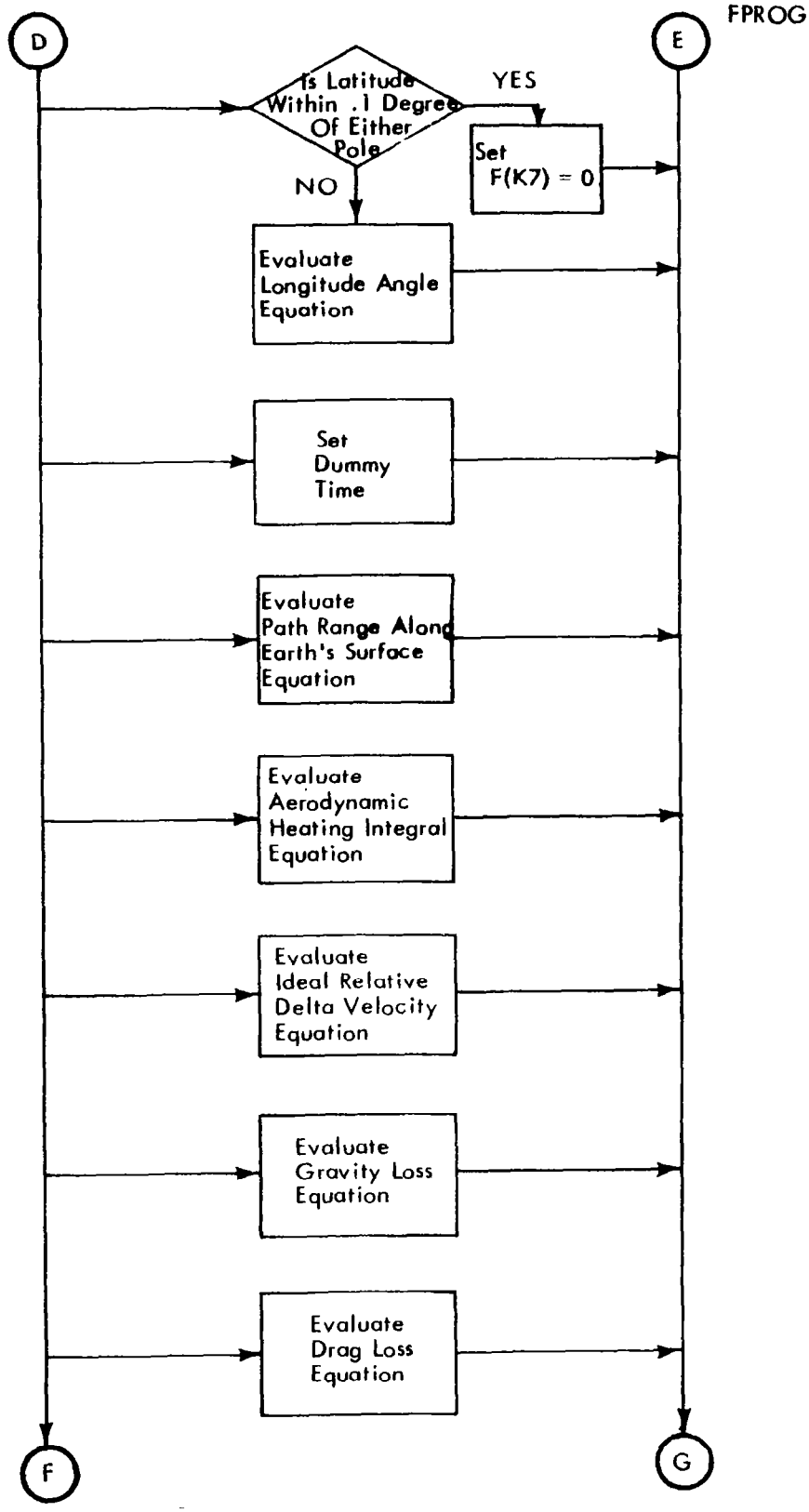
PLAC

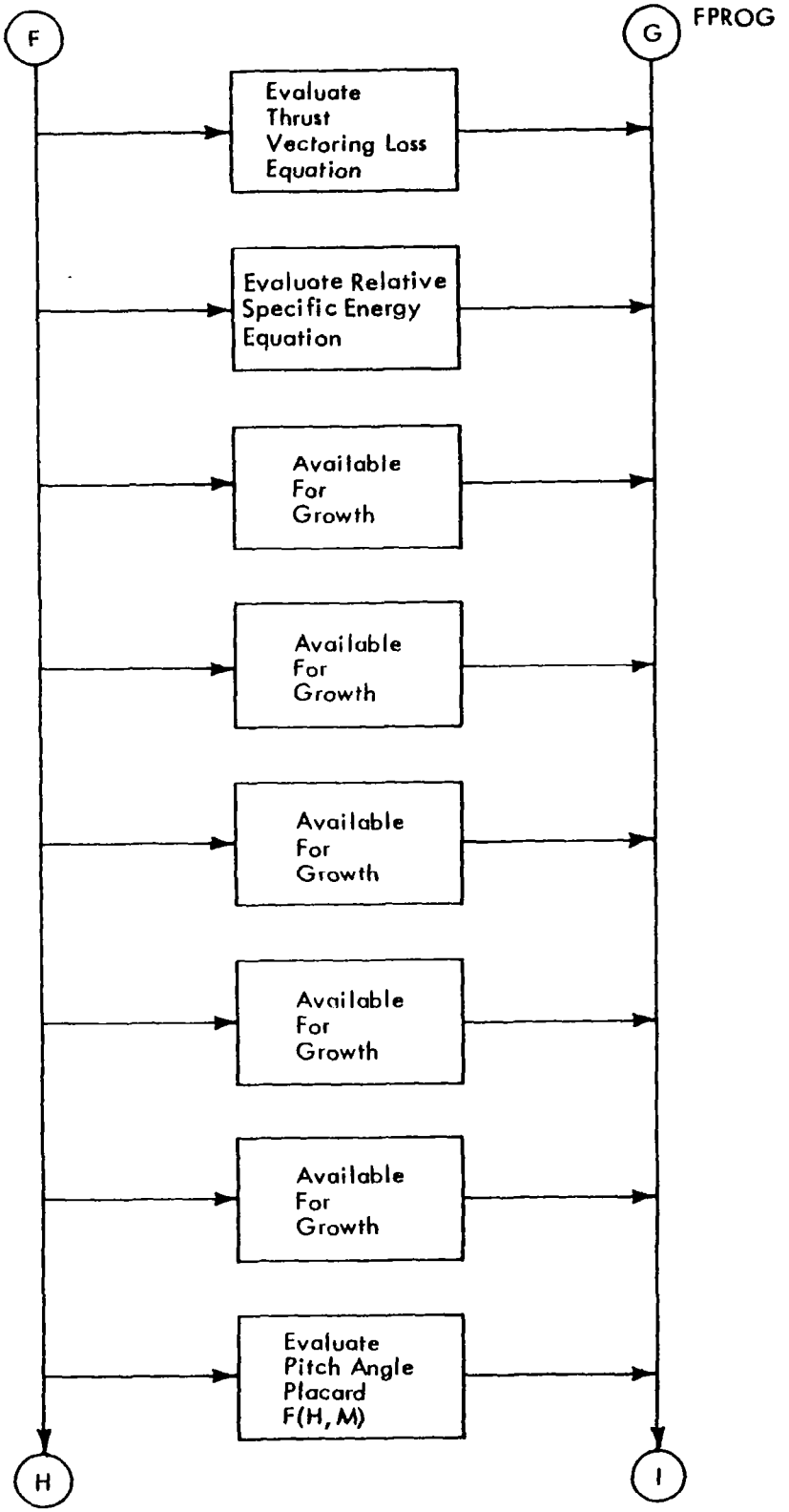
Storage Used

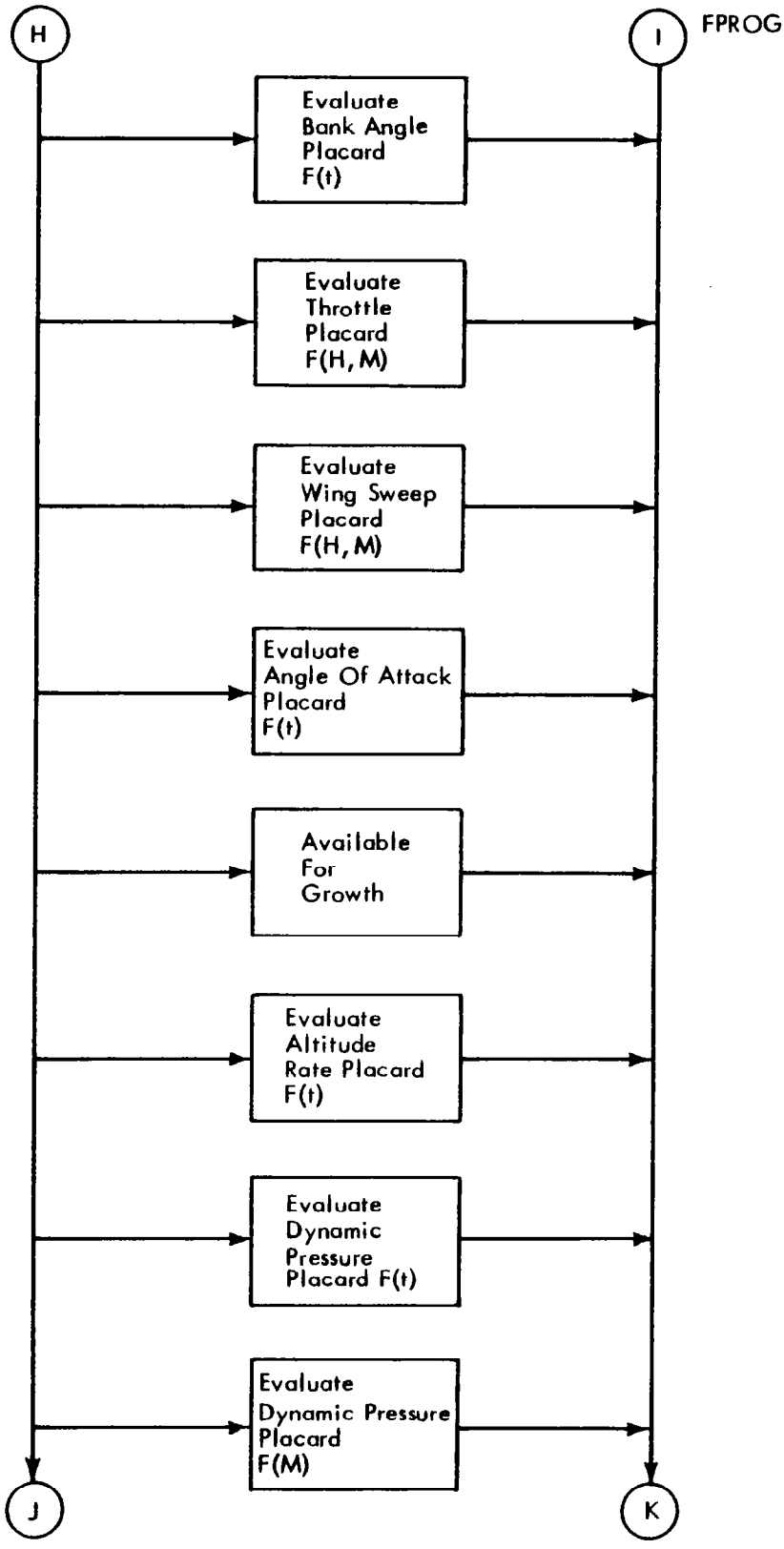
677 cells

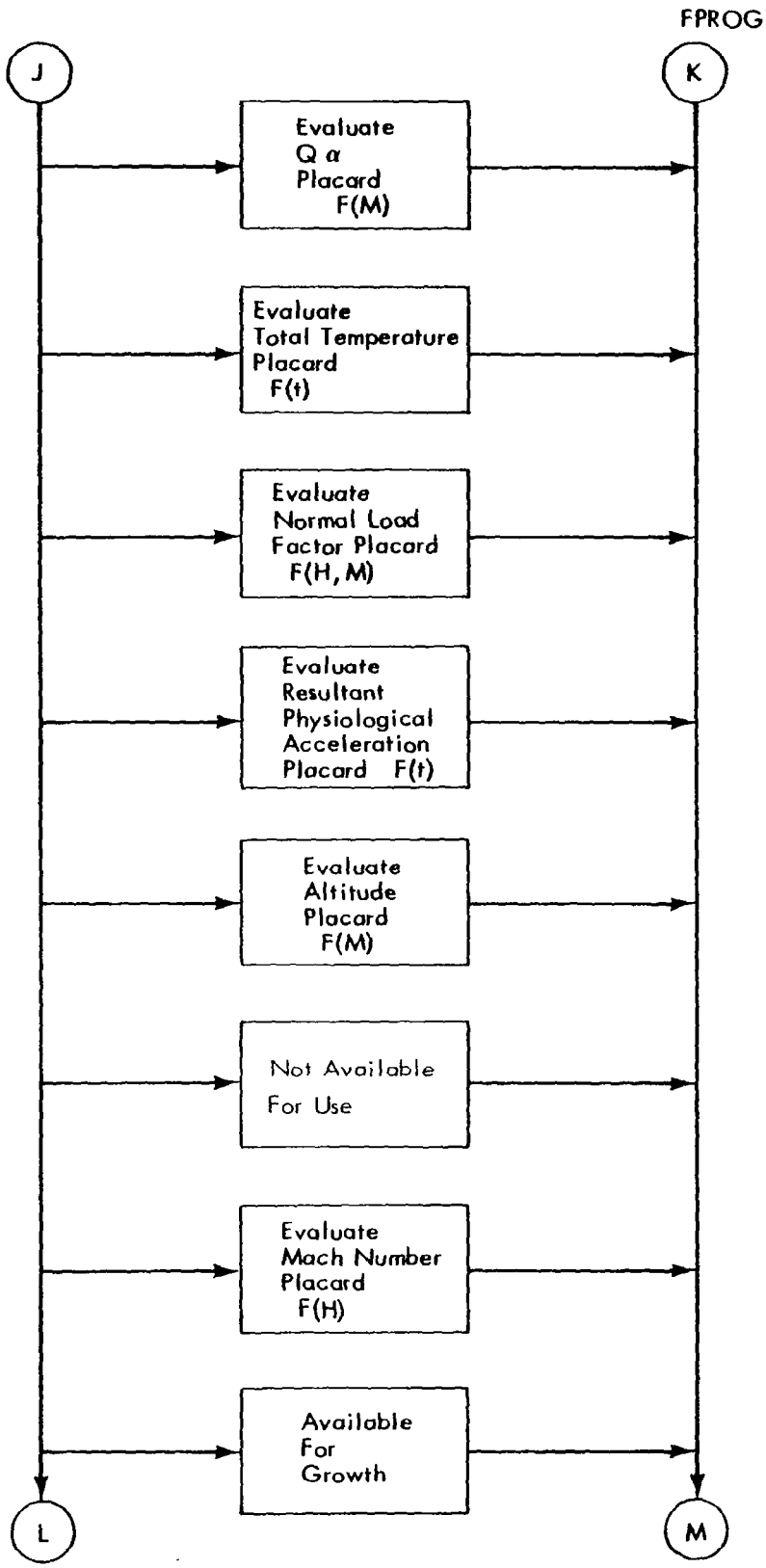

Set Index To Evaluate Subset Of Desired State Variables
Are All State Variables To Be Evaluated
NO
YES
Set Index To Evaluate All Desired State Variables
Evaluate Selected State Variables
100
Evaluate Weight Equation
Evaluate Altitude Equation
Is This The First Stage Of A Tilt Maneuver
YES
NO
Set F(K3) = 0.
Is Velocity Less Than 100.Ft/Sec
YES
NO
A
B
C

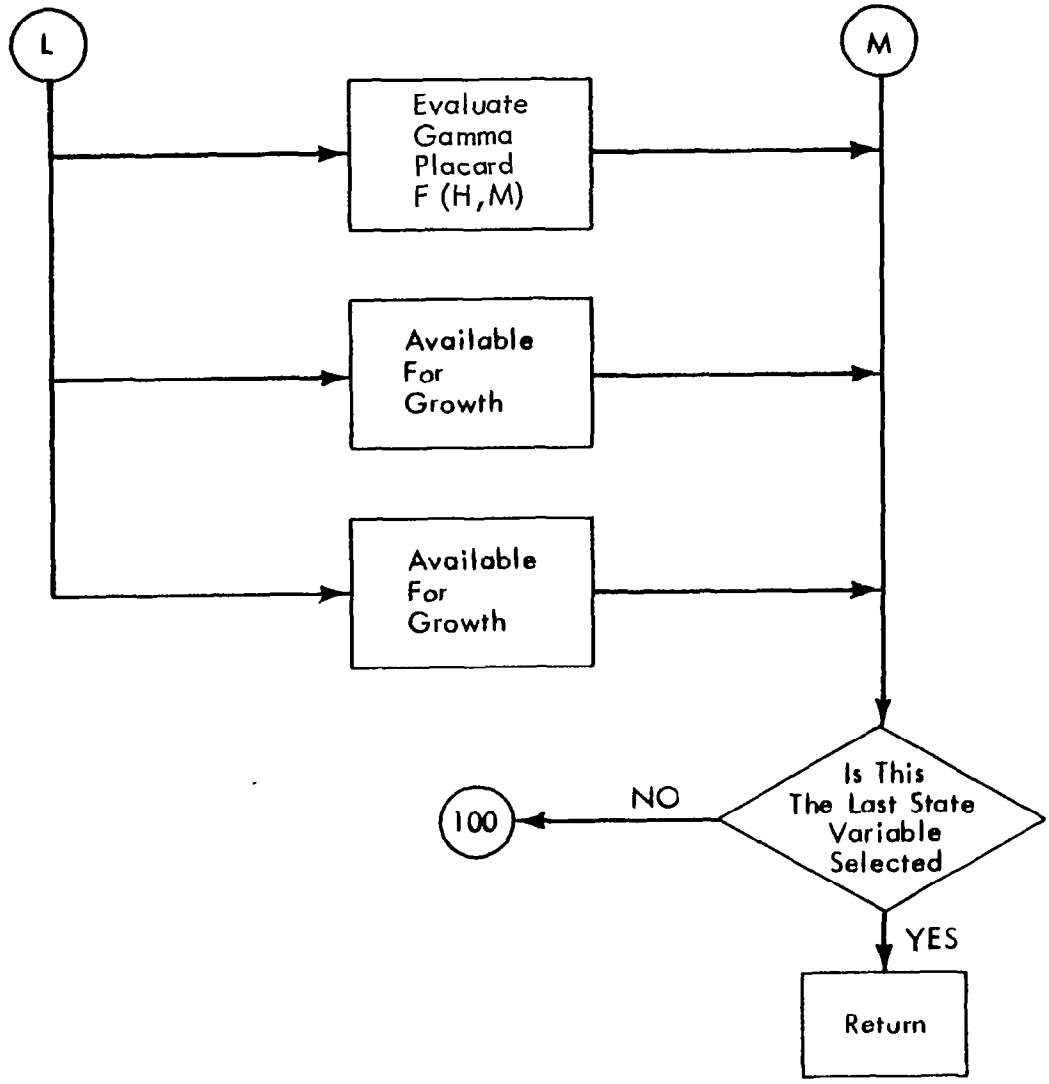

A
B
C
Evaluate Relative Flight Path Angle Equation
Evaluate Relative Velocity Equation
Evaluate Latitude Angle Equation
Is Latitude Within .1 Degree Of Either Pole
YES
NO
Set F(K6) = 0.
Is Gammd Within .1 Degree Of 90. Degrees
YES
NO
Is Velecity Less Than 100. Ft/Sec
YES
NO
Evaluate Relative Heading Angle Equation
D
E


FPROG
D
E
Is Latitude Within .1 Degree Of Either Pole
YES
NO
Set F(K7) = 0
Evaluate Longitude Angle Equation
Set Dummy Time
Evaluate Path Range Along Earth's Surface Equation
Evaluate Aerodynamic Heating Integral Equation
Evaluate Ideal Relative Delta Velocity Equation
Evaluate Gravity Loss Equation
Evaluate Drag Loss Equation
F
G


F
G
FPROG
Evaluate Thrust Vectoring Loss Equation
Evaluate Relative Specific Energy Equation
Available For Growth
Available For Growth
Available For Growth
Available For Growth
Available For Growth
Evaluate Pitch Angle Placard F(H, M)
H
I


H
I
FPROG
Evaluate Bank Angle Placard F(t)
Evaluate Throttle Placard F(H, M)
Evaluate Wing Sweep Placard F(H, M)
Evaluate Angle Of Attack Placard F(t)
Available For Growth
Evaluate Altitude Rate Placard F(t)
Evaluate Dynamic Pressure Placard F(t)
Evaluate Dynamic Pressure Placard F(M)
J
K


FPROG
J
K
Evaluate Q α Placard F(M)
Evaluate Total Temperature Placard F(t)
Evaluate Normal Load Factor Placard F(H, M)
Evaluate Resultant Physiological Acceleration Placard F(t)
Evaluate Altitude Placard F(M)
Not Available For Use
Evaluate Mach Number Placard F(H)
Available For Growth
L
M


L
M
Evaluate
Gamma
Placard
F (H,M)
Available
For
Growth
Available
For
Growth
Is This
The Last State
Variable
Selected
NO
100
YES
Return

GUIDE — Closed Loop Guidance Control Variables

Purpose

GUIDE calculates control variables for generating a nominal flight path.

Method

The NC(13) controls the manner in which nominal trajectory is generated. The GUIDE subroutine is called only for the nominal trajectory, if NC(13) $\neq$ 0. If the first stage is a tilt maneuver, GUIDE is bypassed. The number of phases used to generate the nominal is given by NC(13). Corresponding to each of the phases, a guidance mode and phase stopping condition must be specified. The provision is made to allow the phase stopping condition to be approached from below or above. The guidance modes are discussed under "Nominal Trajectory Generation."

Assumptions and Limitations

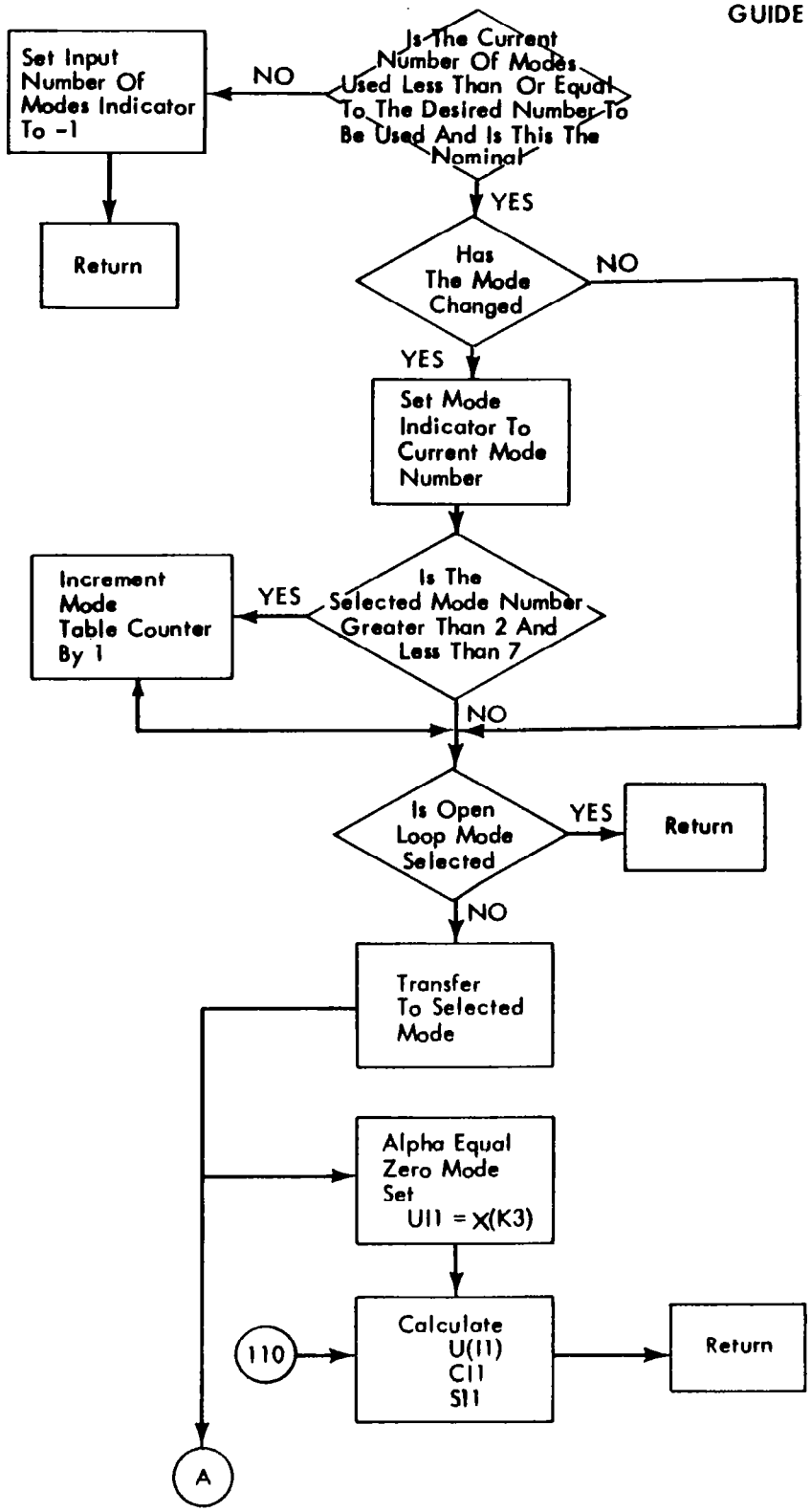
A maximum of 12 phases may be used to generate the nominal flight path. Five tables have been reserved for guidance modes. The equations solved for the control variables are linearized as described under "Nominal Trajectory Generation."

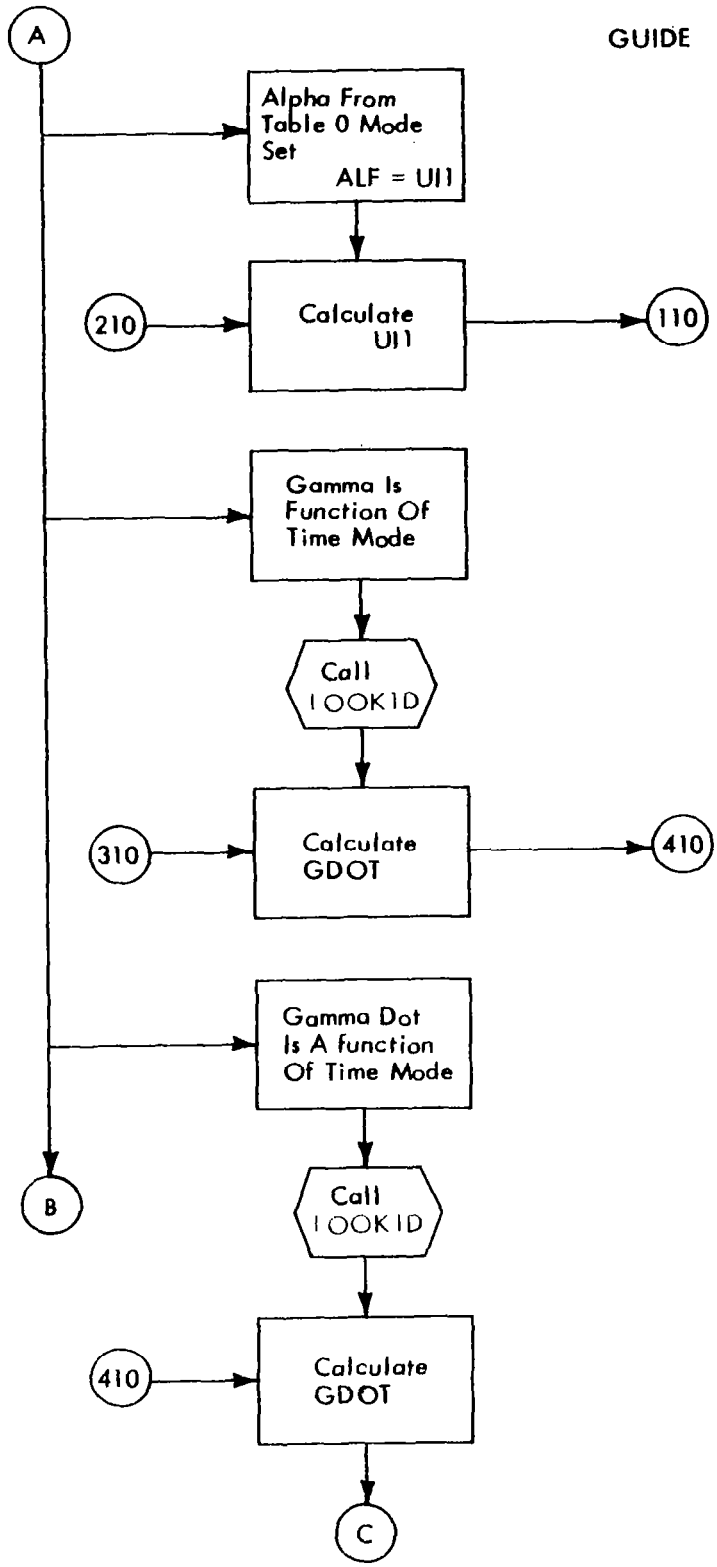
Subroutines Called

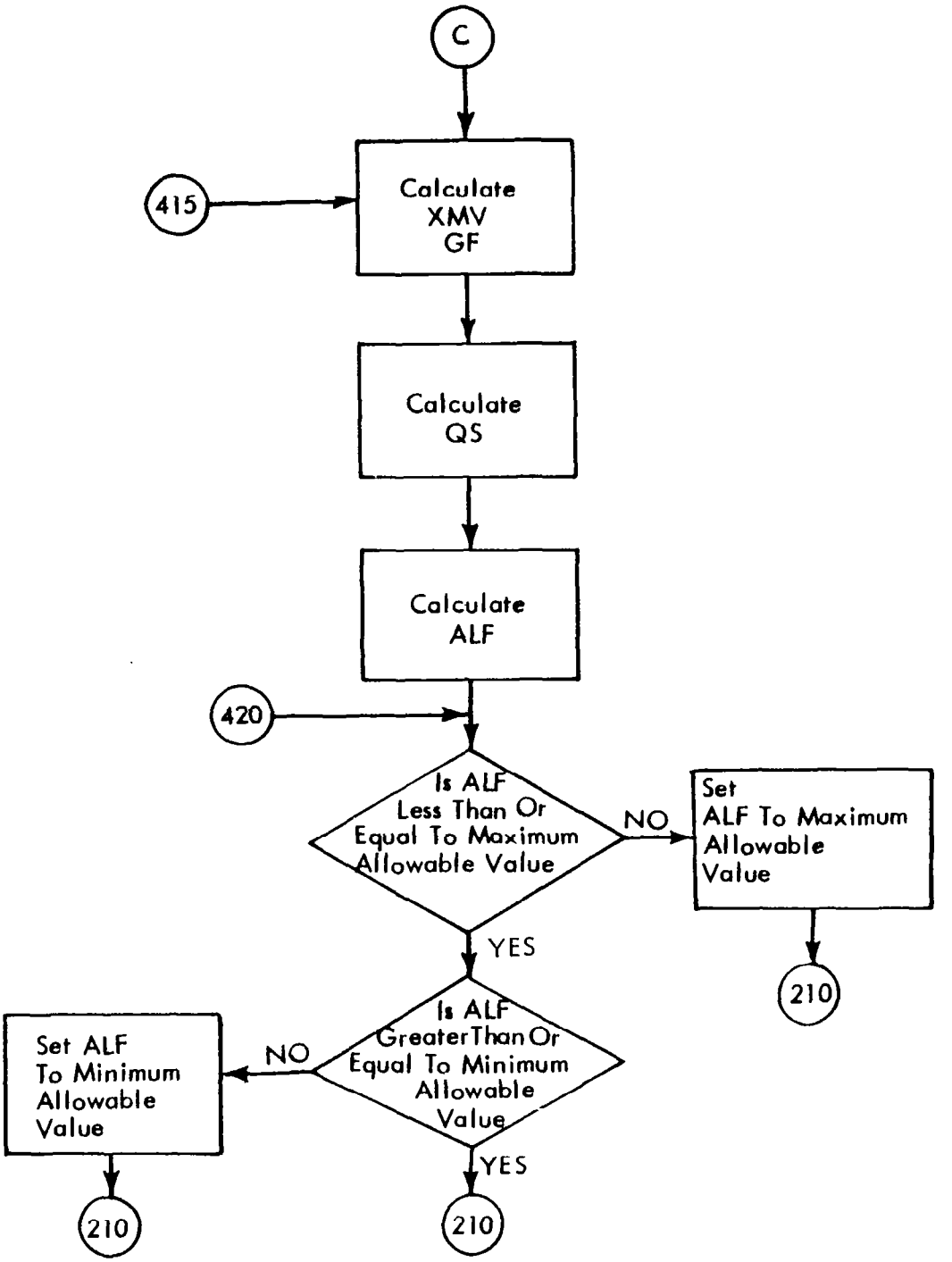
AKSTP LOOK1D

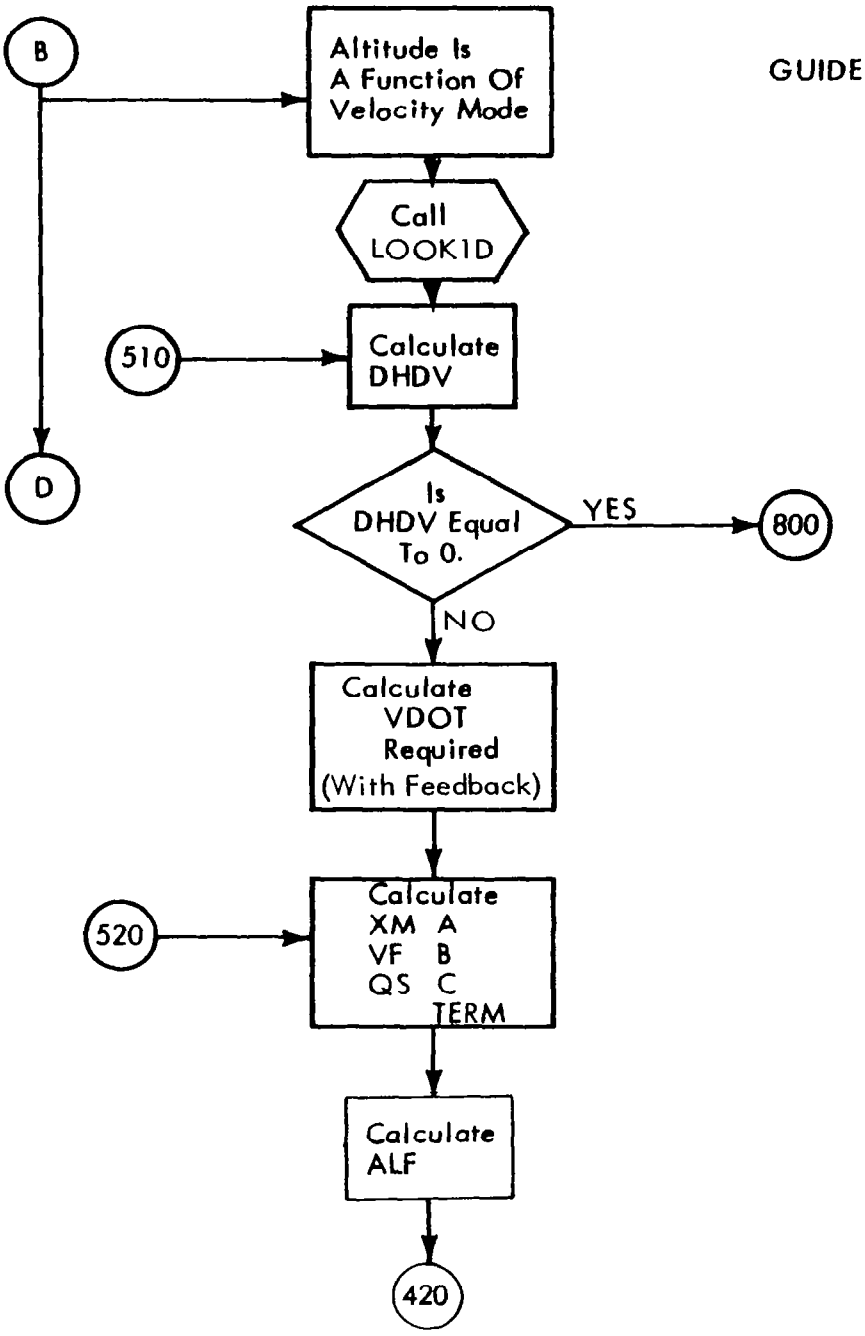
Storage Used

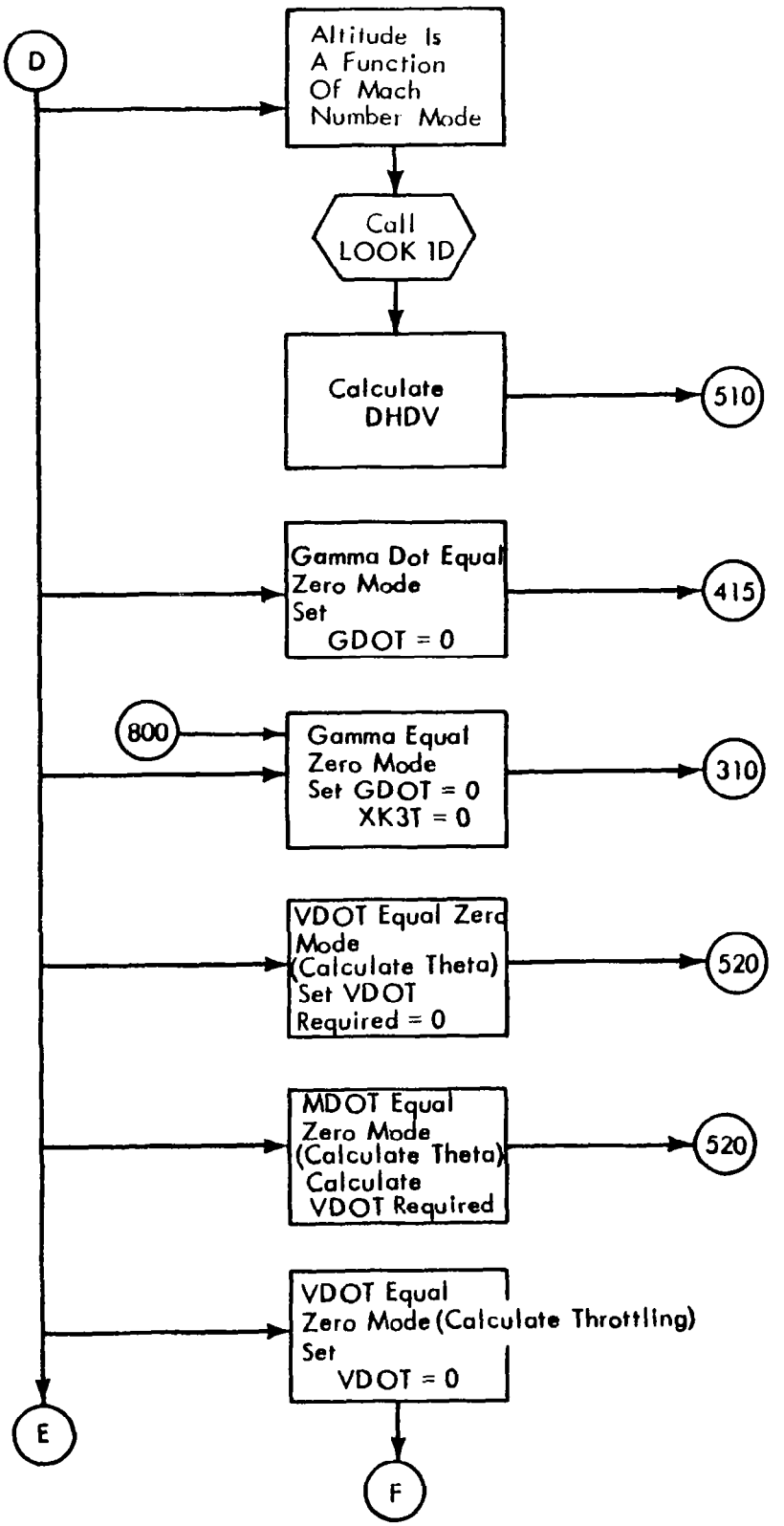
657 cells

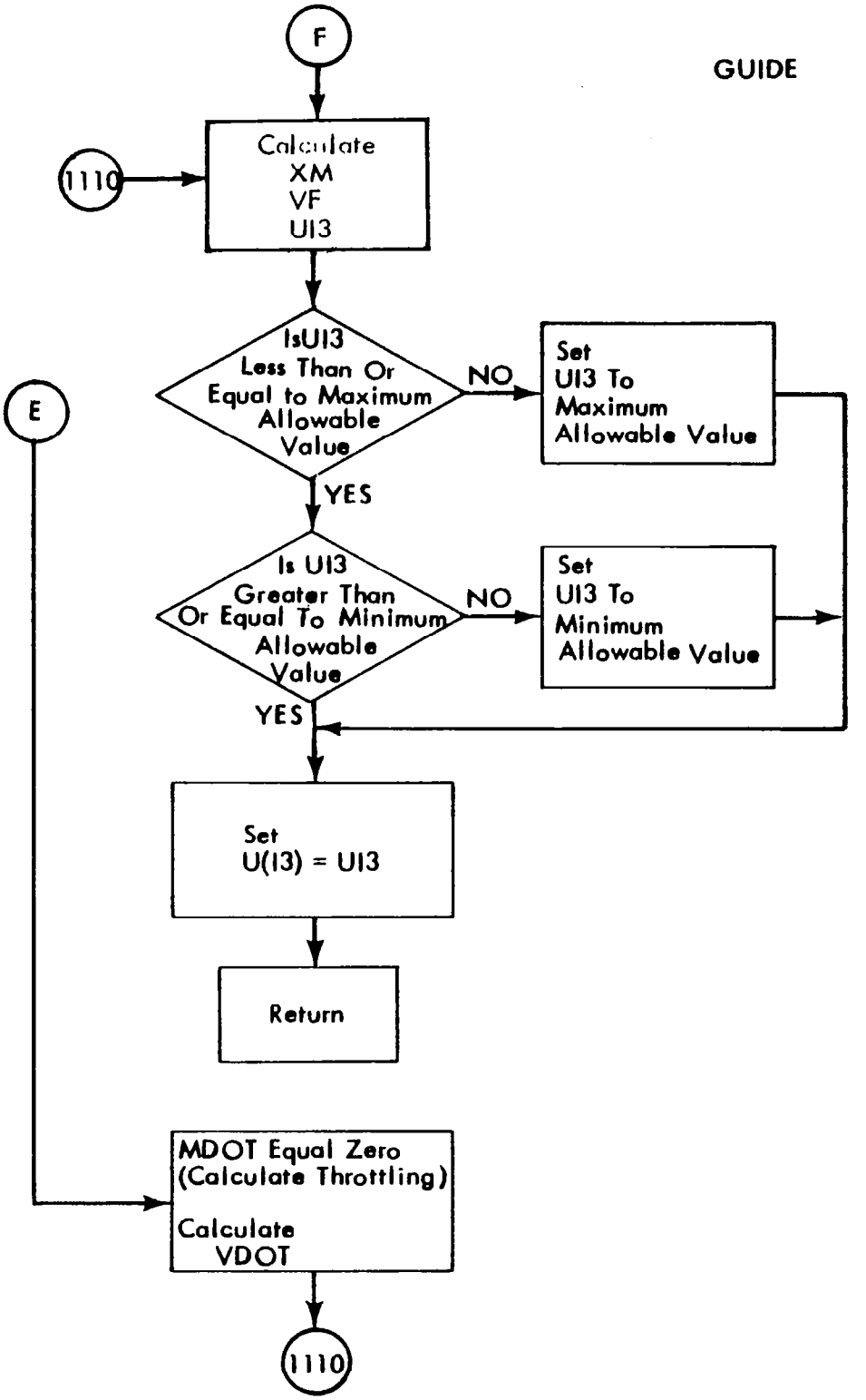

Is The Current Number Of Modes Used Less Than Or Equal To The Desired Number To Be Used And Is This The Nominal
NO
Set Input Number Of Modes Indicator To -1
Return
YES
Has The Mode Changed
NO
YES
Set Mode Indicator To Current Mode Number
Is The Selected Mode Number Greater Than 2 And Less Than 7
YES
Increment Mode Table Counter By 1
NO
Is Open Loop Mode Selected
YES
Return
NO
Transfer To Selected Mode
Alpha Equal Zero Mode Set
U11 = X(K3)
110
Calculate
U(11)
C11
S11
Return
A


A
Alpha From Table 0 Mode Set
ALF = UI1
210
Calculate UI1
110
Gamma Is Function Of Time Mode
Call LOOKID
310
Calculate GDOT
410
Gamma Dot Is A function Of Time Mode
B
Call LOOKID
410
Calculate GDOT
C


C
415
Calculate XMV GF
Calculate QS
Calculate ALF
420
Is ALF Less Than Or Equal To Maximum Allowable Value
NO
Set ALF To Maximum Allowable Value
210
YES
Is ALF GreaterThan Or Equal To Minimum Allowable Value
NO
Set ALF To Minimum Allowable Value
210
YES
210


GUIDE
B
Altitude Is
A Function Of
Velocity Mode
Call
LOOK1D
510
Calculate
DHDV
D
Is
DHDV Equal
To 0.
YES
800
NO
Calculate
VDOT
Required
(With Feedback)
520
Calculate
XM A
VF B
QS C
TERM
Calculate
ALF
420


D
Altitude Is A Function Of Mach Number Mode
Call LOOK 1D
Calculate DHDV
510
Gamma Dot Equal Zero Mode Set GDOT = 0
415
800
Gamma Equal Zero Mode Set GDOT = 0 XK3T = 0
310
VDOT Equal Zero Mode (Calculate Theta) Set VDOT Required = 0
520
MDOT Equal Zero Mode (Calculate Theta) Calculate VDOT Required
520
VDOT Equal Zero Mode (Calculate Throttling) Set VDOT = 0
E
F


F
1110
Calculate
XM
VF
UI3
IsUI3
Less Than Or
Equal to Maximum
Allowable
Value
NO
Set
UI3 To
Maximum
Allowable Value
YES
Is UI3
Greater Than
Or Equal To Minimum
Allowable
Value
NO
Set
UI3 To
Minimum
Allowable Value
YES
E
Set
U(I3) = UI3
Return
MDOT Equal Zero
(Calculate Throttling)
Calculate
VDOT
1110

INITAL — Initial Data

Purpose

INITAL is called once at the beginning of a data case and performs the following tasks in the order listed below.

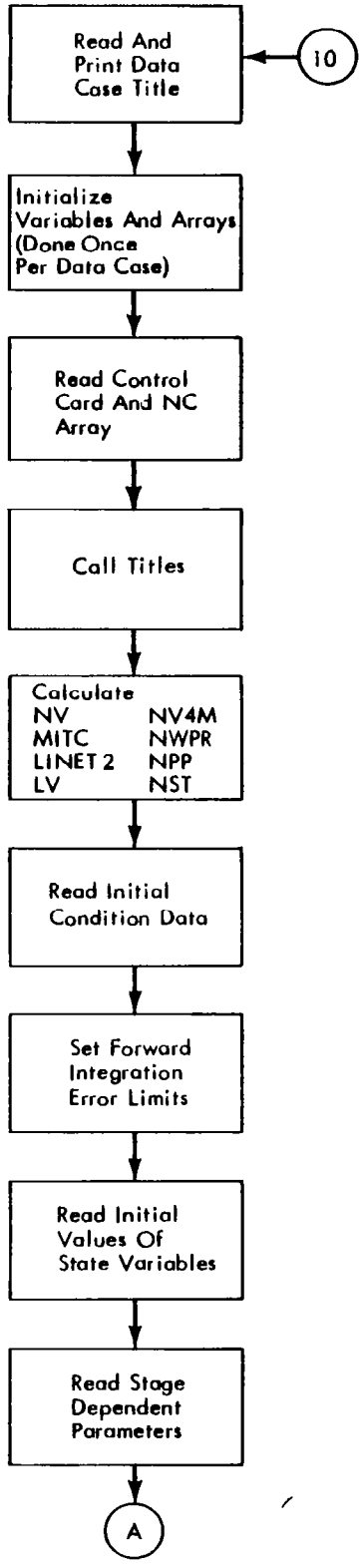
1) Read the title card;
2) Initialization of important program indicators and arrays;
3) Read the control card, NC array, and initial condition data;
4) Set the forward integration error limits;
5) Read the stage-dependent and constraint-dependent parameters;
6) Print the initial conditions and stage-dependent parameters;
7) Construct heading block for printout from subroutine MATOUT;
8) Print optimization and stopping condition parameters;
9) Read and print free initial condition parameters if necessary;
10) Read and print phasing guidance parameters if necessary;
11) Print control parameters;
12) Read and print nonstage-dependent tables (including control variable and restart tables). Store part of control variable table on KTAN unit if necessary;
13) Store nonstage-dependent tables (except control variable and restart tables) on KDAT unit;
14) Read and print stage-dependent tables and store on KDAT unit.

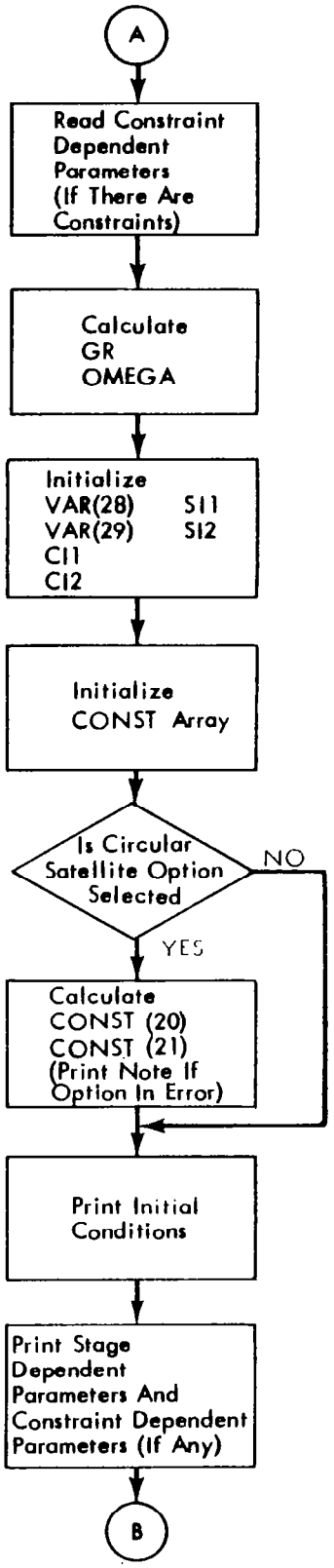
Subroutines Called

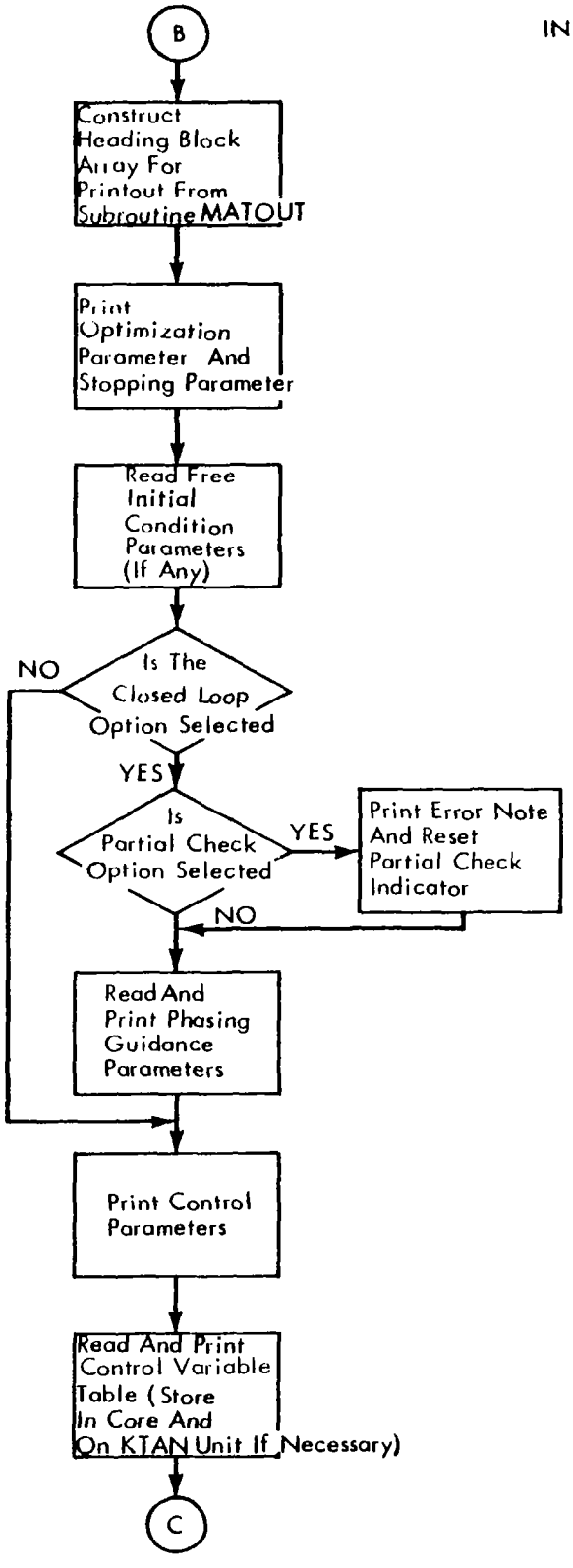
TITLES

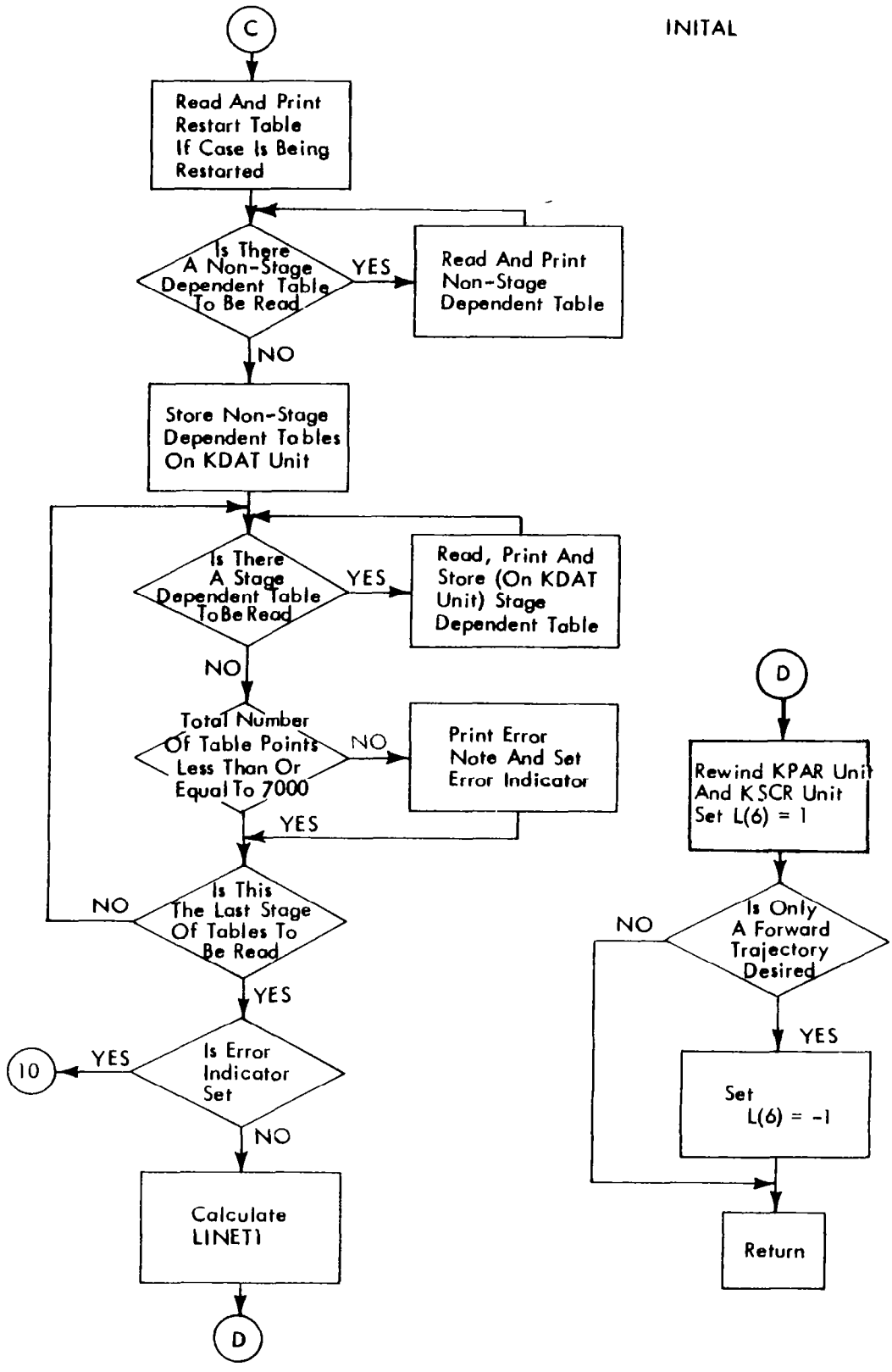
Storage Used

2692 cells


Read And
Print Data
Case Title
10
Initialize
Variables And Arrays
(Done Once
Per Data Case)
Read Control
Card And NC
Array
Call Titles
Calculate
NV
MITC
LINET 2
LV
NV4M
NWPR
NPP
NST
Read Initial
Condition Data
Set Forward
Integration
Error Limits
Read Initial
Values Of
State Variables
Read Stage
Dependent
Parameters
A


A
Read Constraint Dependent Parameters (If There Are Constraints)
Calculate GR OMEGA
Initialize VAR(28) S11 VAR(29) S12 CI1 CI2
Initialize CONST Array
Is Circular Satellite Option Selected
NO
YES
Calculate CONST (20) CONST (21) (Print Note If Option In Error)
Print Initial Conditions
Print Stage Dependent Parameters And Constraint Dependent Parameters (If Any)
B


B
Construct Heading Block Array For Printout From Subroutine MATOUT
Print Optimization Parameter And Stopping Parameter
Read Free Initial Condition Parameters (If Any)
Is The Closed Loop Option Selected
NO
YES
Is Partial Check Option Selected
YES
Print Error Note And Reset Partial Check Indicator
NO
Read And Print Phasing Guidance Parameters
Print Control Parameters
Read And Print Control Variable Table (Store In Core And On KTAN Unit If Necessary)
C


C
Read And Print Restart Table If Case Is Being Restarted
Is There A Non-Stage Dependent Table To Be Read
YES
Read And Print Non-Stage Dependent Table
NO
Store Non-Stage Dependent Tables On KDAT Unit
Is There A Stage Dependent Table To Be Read
YES
Read, Print And Store (On KDAT Unit) Stage Dependent Table
NO
Total Number Of Table Points Less Than Or Equal To 7000
NO
Print Error Note And Set Error Indicator
YES
Is This The Last Stage Of Tables To Be Read
NO
YES
Is Error Indicator Set
YES
10
NO
Calculate LINET1
D
D
Rewind KPAR Unit And KSCR Unit Set L(6) = 1
Is Only A Forward Trajectory Desired
NO
YES
Set L(6) = -1
Return

INITCO — Initialize State Variables

Purpose

This subroutine is called at the start of each stage and is assigned the following tasks:

1) Initialize the X array at the start of stage 1;

2) Update the free elements in the X array at the start of stage 1;

3) Initialize weight at the start of every stage;

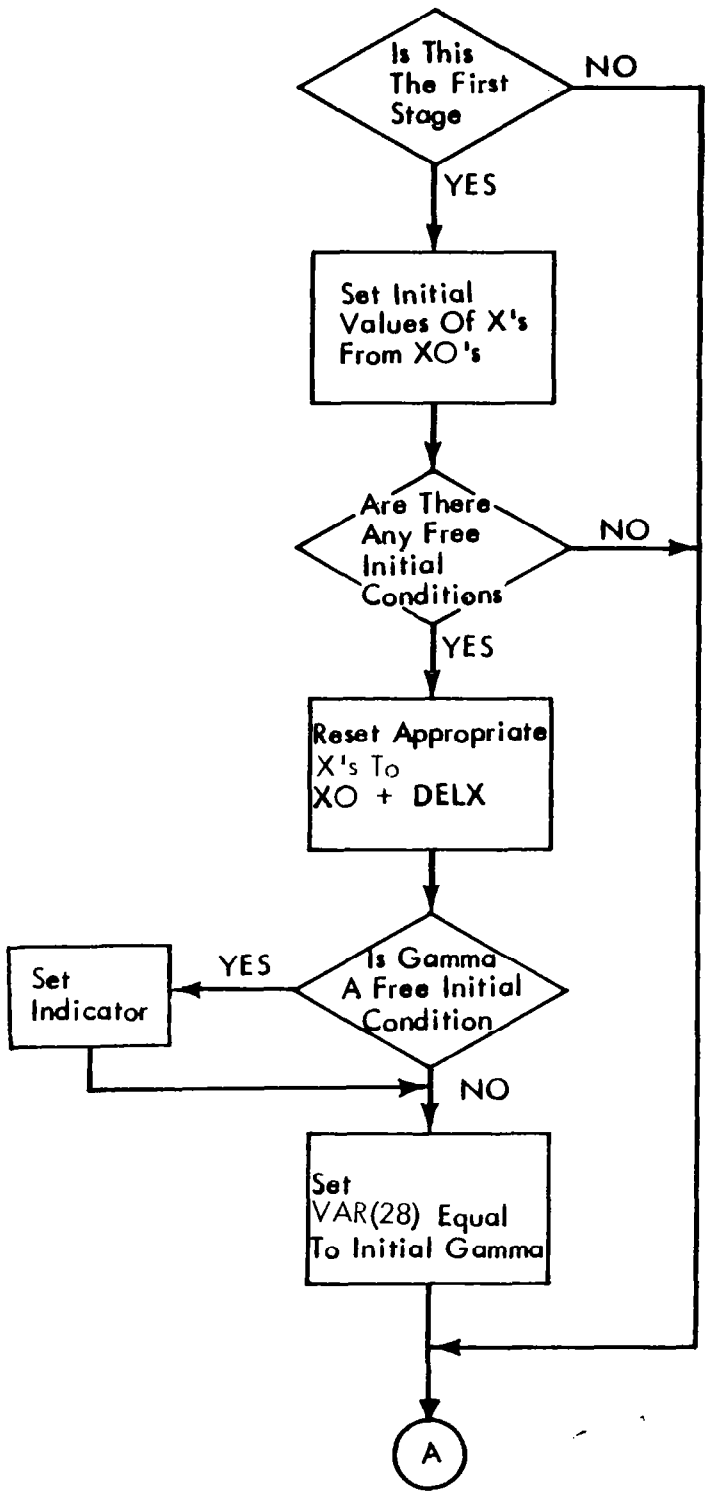
4) Update weight at the start of each stage if weight is a free initial condition.

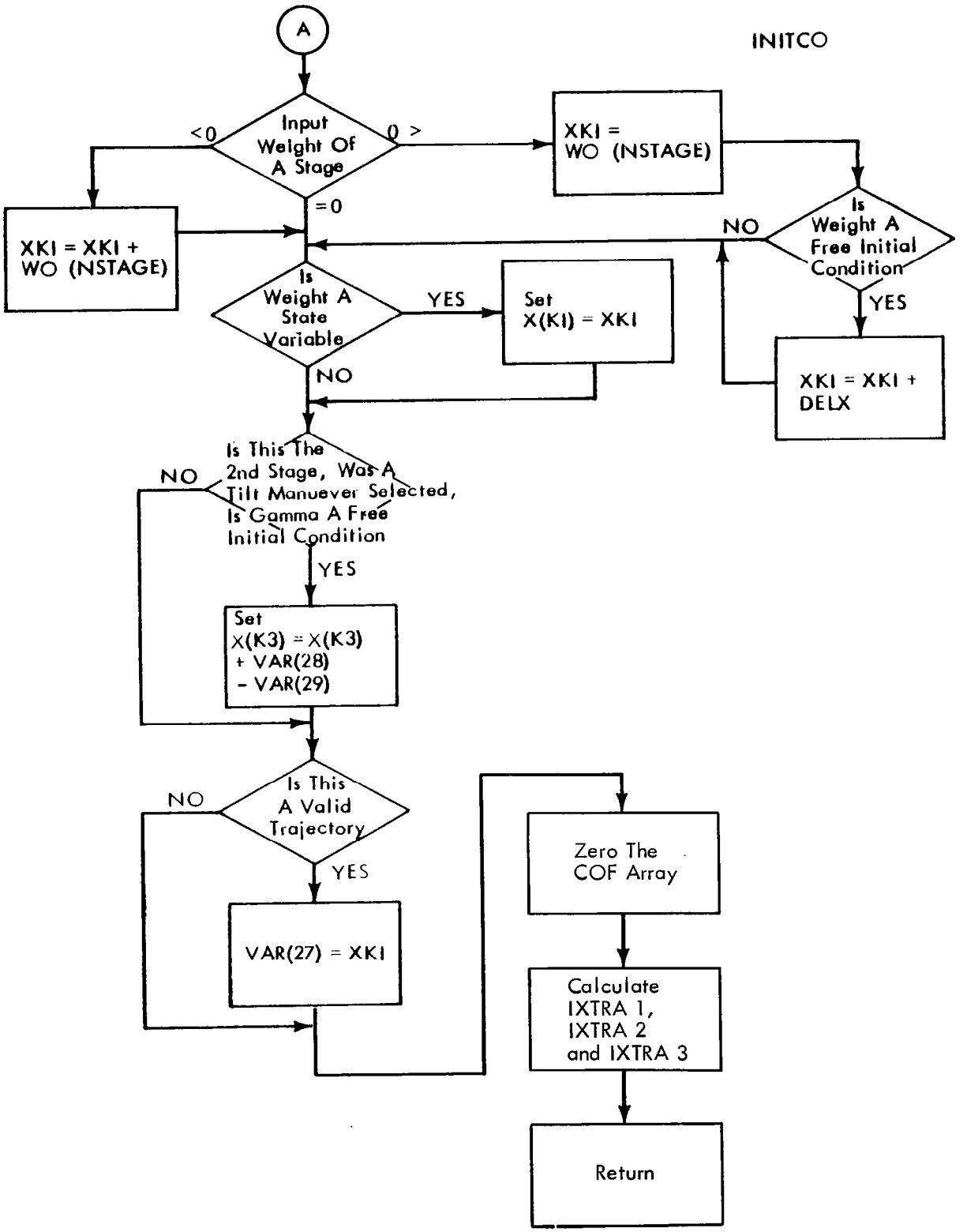
Method

The XO array is used to initialize X at the start of stage 1 and the WO array is used to initialize the weight at the start of each stage. The input WO and XO arrays are updated in VALID using the values of DELX computed by VARIC.

Storage Used

117 cells


Is This The First Stage
NO
YES
Set Initial Values Of X's From XO's
Are There Any Free Initial Conditions
NO
YES
Reset Appropriate X's To XO + DELX
Is Gamma A Free Initial Condition
YES
Set Indicator
NO
Set VAR(28) Equal To Initial Gamma
A


INITCO
A
Input Weight Of A Stage
<0
0 >
=0
XKI = WO (NSTAGE)
XKI = XKI + WO (NSTAGE)
Is Weight A Free Initial Condition
NO
YES
XKI = XKI + DELX
Is Weight A State Variable
YES
Set X(KI) = XKI
NO
Is This The 2nd Stage, Was A Tilt Manuever Selected, Is Gamma A Free Initial Condition
NO
YES
Set X(K3) = X(K3) + VAR(28) - VAR(29)
Is This A Valid Trajectory
NO
YES
VAR(27) = XKI
Zero The COF Array
Calculate IXTRA 1, IXTRA 2 and IXTRA 3
Return

KCALC — Step Size Logic Subroutine

Purpose

This subroutine is called following every try if MATRIX determines a non-negative majority vote. KCALC is assigned the following tasks:

1) Compute maximum allowable constraint motion;

2) Set L(6) for next trajectory;

3) Compute step size coefficient (COEFK) for next trajectory.

METHOD

KCALC computes the step size coefficient based on performance and constraint linearities using the parabolic curve fitting method described in appendix C. Maximum step size coefficients are also determined based on the maximum allowable travel of each constraint. The best step size is then determined for the next trajectory.

Variables not in COMMON are transmitted between MATRIX and KCALC through the calling sequence

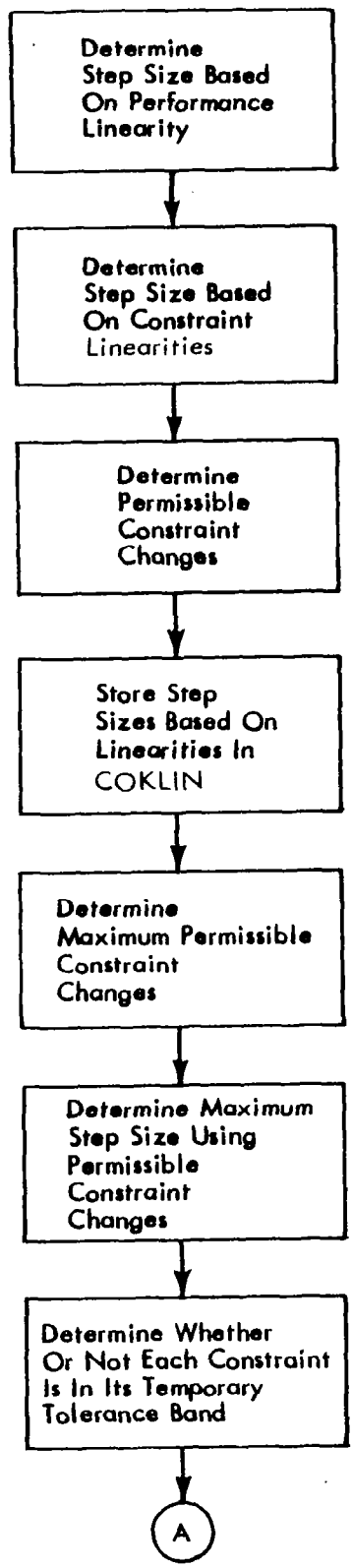
CALL KCALC (DPSI, DPSIP, PSIBWD, PSIFWD, PSIK, PSIKTR, PSINL, XX)

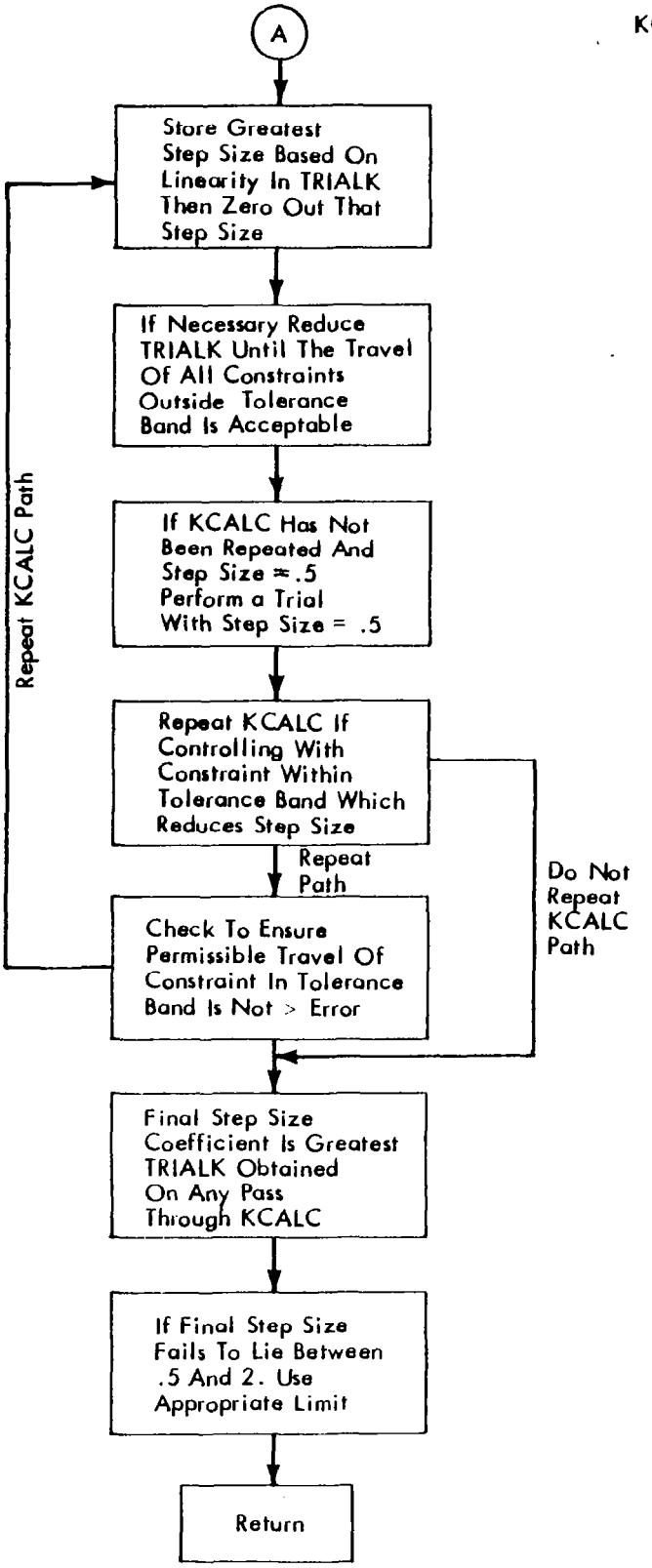
where:

DPSI	= change in constraint end points
DPSIP	= predicted change in constraint end points
PSIBWD	= nondimensional allowable change of a constraint in direction of desired constraint value
PSIFWD	= nondimensional allowable change of a constraint in direction away from desired constraint value
PSIK	= constraint step size coefficients
PSIKTR	= maximum step size coefficients of constraints due to maximum permissible travel
PSINL	= constraint nonlinearities
XX	= current end point values

Storage Used

832 cells


Determine Step Size Based On Performance Linearity
Determine Step Size Based On Constraint Linearities
Determine Permissible Constraint Changes
Store Step Sizes Based On Linearities In COKLIN
Determine Maximum Permissible Constraint Changes
Determine Maximum Step Size Using Permissible Constraint Changes
Determine Whether Or Not Each Constraint Is In Its Temporary Tolerance Band
A


A
Store Greatest Step Size Based On Linearity In TRIALK Then Zero Out That Step Size
If Necessary Reduce TRIALK Until The Travel Of All Constraints Outside Tolerance Band Is Acceptable
If KCALC Has Not Been Repeated And Step Size = .5 Perform a Trial With Step Size = .5
Repeat KCALC If Controlling With Constraint Within Tolerance Band Which Reduces Step Size
Repeat Path
Do Not Repeat KCALC Path
Repeat KCALC Path
Check To Ensure Permissible Travel Of Constraint In Tolerance Band Is Not > Error
Final Step Size Coefficient Is Greatest TRIALK Obtained On Any Pass Through KCALC
If Final Step Size Fails To Lie Between .5 And 2. Use Appropriate Limit
Return

LAMBDA — Reverse Integration Flow Controller

Purpose

LAMBDA controls the flow of the program during the backward integration.

Method

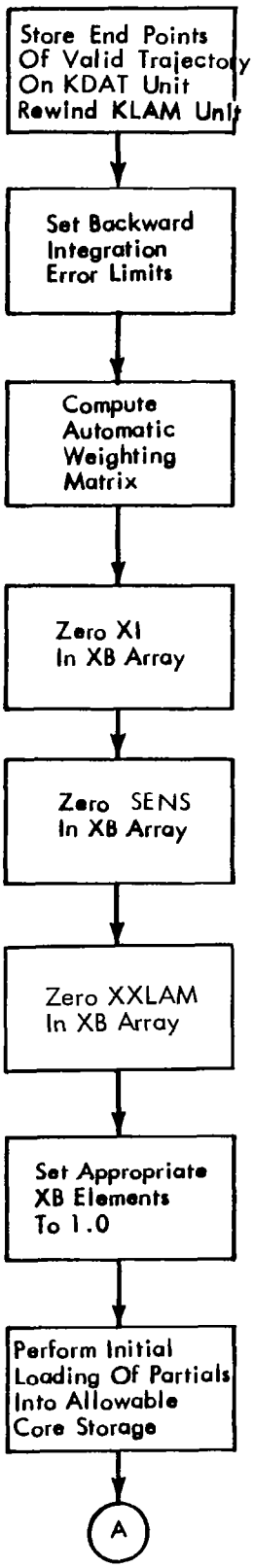
The adjoint variables are initialized prior to performing the backward integration. The automatic weighting matrix is calculated using the method described under "Automatic Weighting Matrices."

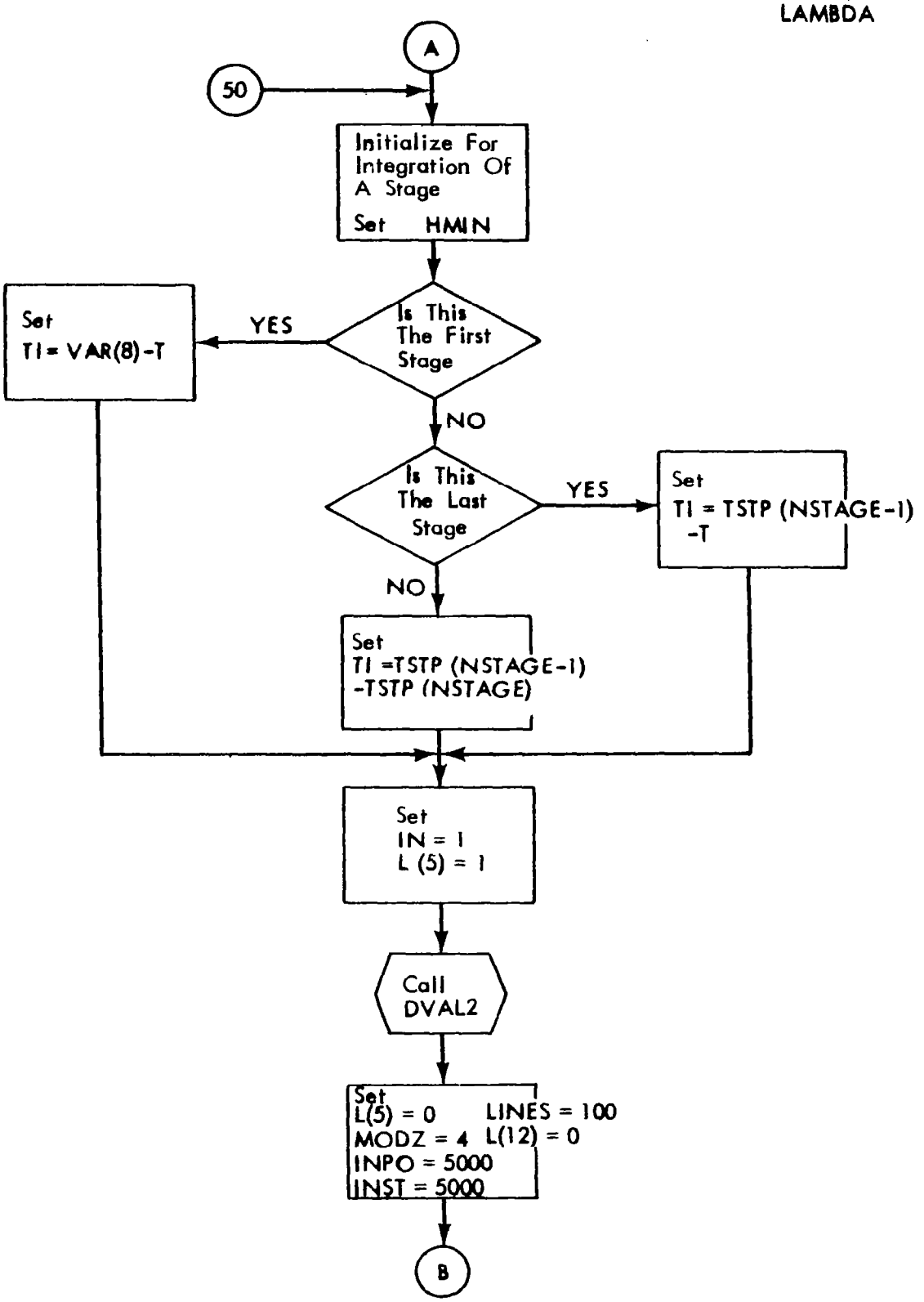
Subroutines Called

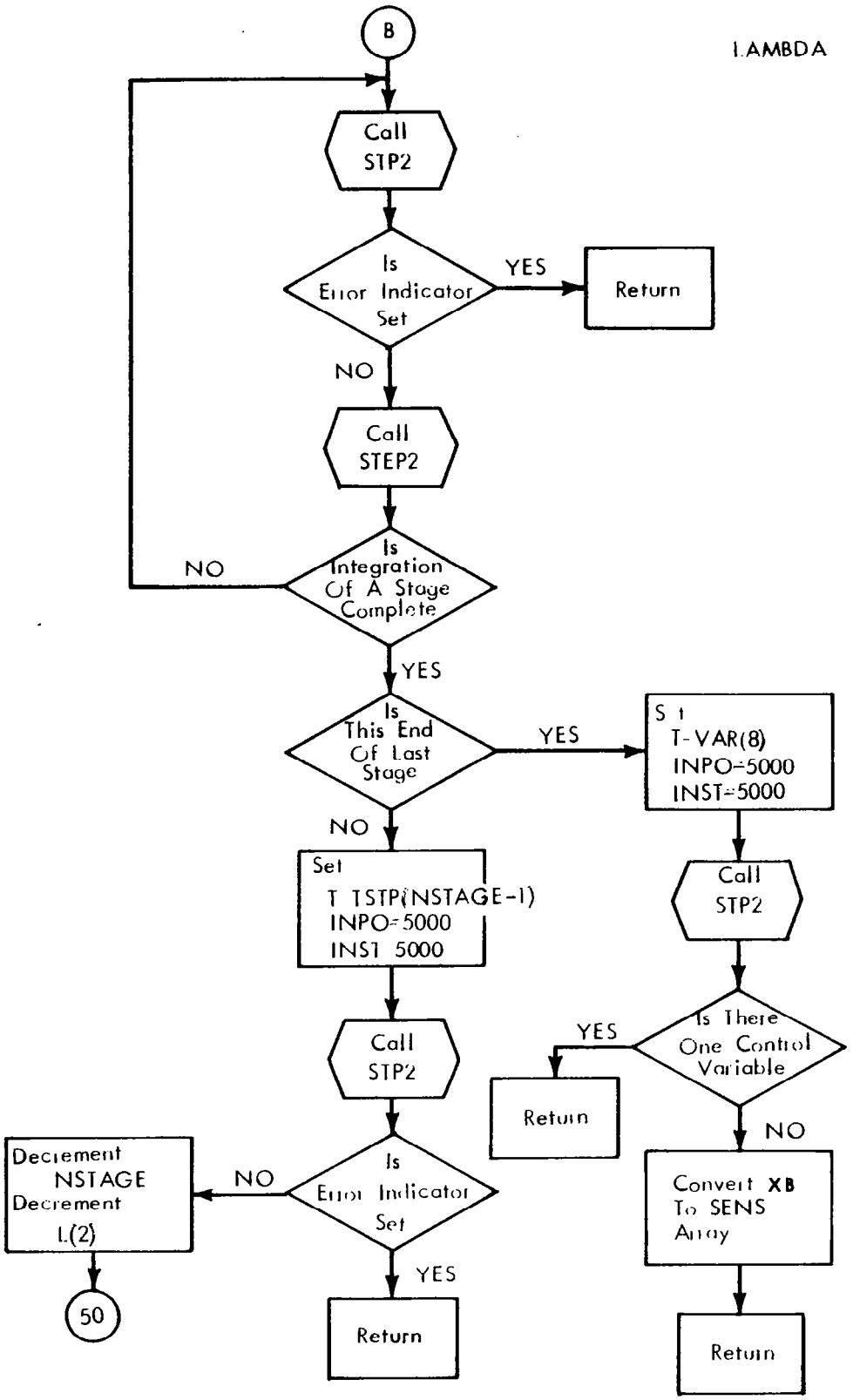
DVAL2	STP2	STEP2

Storage Used

518 cells


Store End Points
Of Valid Trajectory
On KDAT Unit
Rewind KLAM Unit
Set Backward
Integration
Error Limits
Compute
Automatic
Weighting
Matrix
Zero XI
In XB Array
Zero SENS
In XB Array
Zero XXLAM
In XB Array
Set Appropriate
XB Elements
To 1.0
Perform Initial
Loading Of Partials
Into Allowable
Core Storage
A


A
50
Initialize For Integration Of A Stage
Set HMIN
Is This The First Stage
YES
Set TI = VAR(8)-T
NO
Is This The Last Stage
YES
Set TI = TSTP (NSTAGE-1) -T
NO
Set TI =TSTP (NSTAGE-1) -TSTP (NSTAGE)
Set IN = 1 L (5) = 1
Call DVAL2
Set L(5) = 0 LINES = 100 MODZ = 4 L(12) = 0 INPO = 5000 INST = 5000
B


B
Call STP2
Is Error Indicator Set
YES
Return
NO
Call STEP2
Is Integration Of A Stage Complete
NO
YES
Is This End Of Last Stage
YES
S t T-VAR(8) INPO=5000 INST=5000
NO
Set T TSTP(NSTAGE-1) INPO=5000 INST 5000
Call STP2
Call STP2
Is There One Control Variable
YES
Return
NO
Convert XB To SENS Array
Return
Is Error Indicator Set
NO
Decrement NSTAGE Decrement I.(2)
50
YES
Return

LOAD — Main Program

Purpose

LOAD controls the program flow from link to link and causes links to be overlayed.

Method

The program is broken into nine links, each performing a specific function.

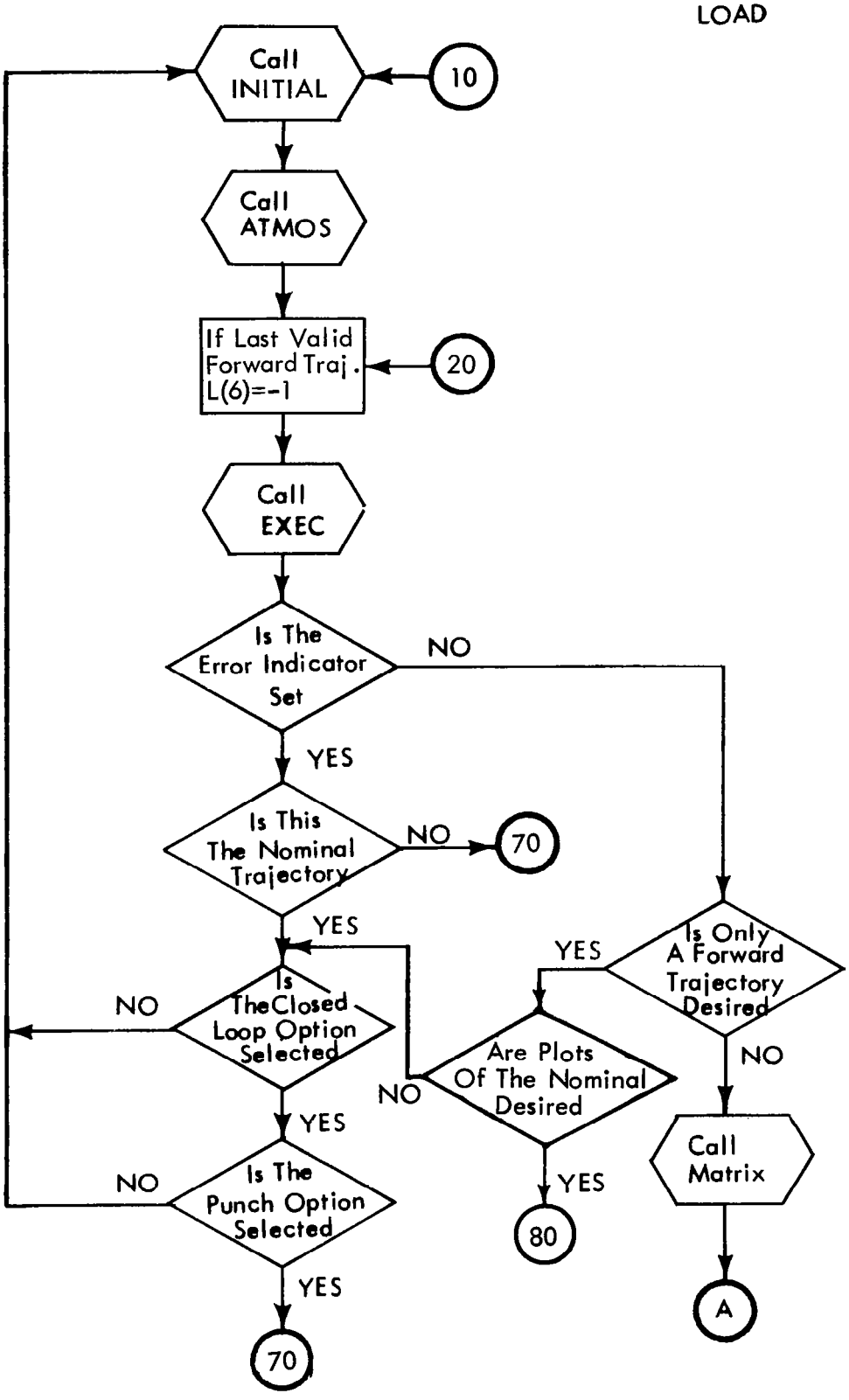
Link No.	Purpose of Link
0	General program flow
1	Performs forward integration; called once per trial
2	Numerical partial check link
3	Closed-loop guidance link
4	Performs the steepest-ascent calculations; called once per trial and twice per valid trajectory
5	Controls program flow for reverse integration; called once per valid trajectory
6	Initilization, called once per data case
7	Performs final output of trajectory data; called once per data case
8	Controls the plotting routines; called once per computer run

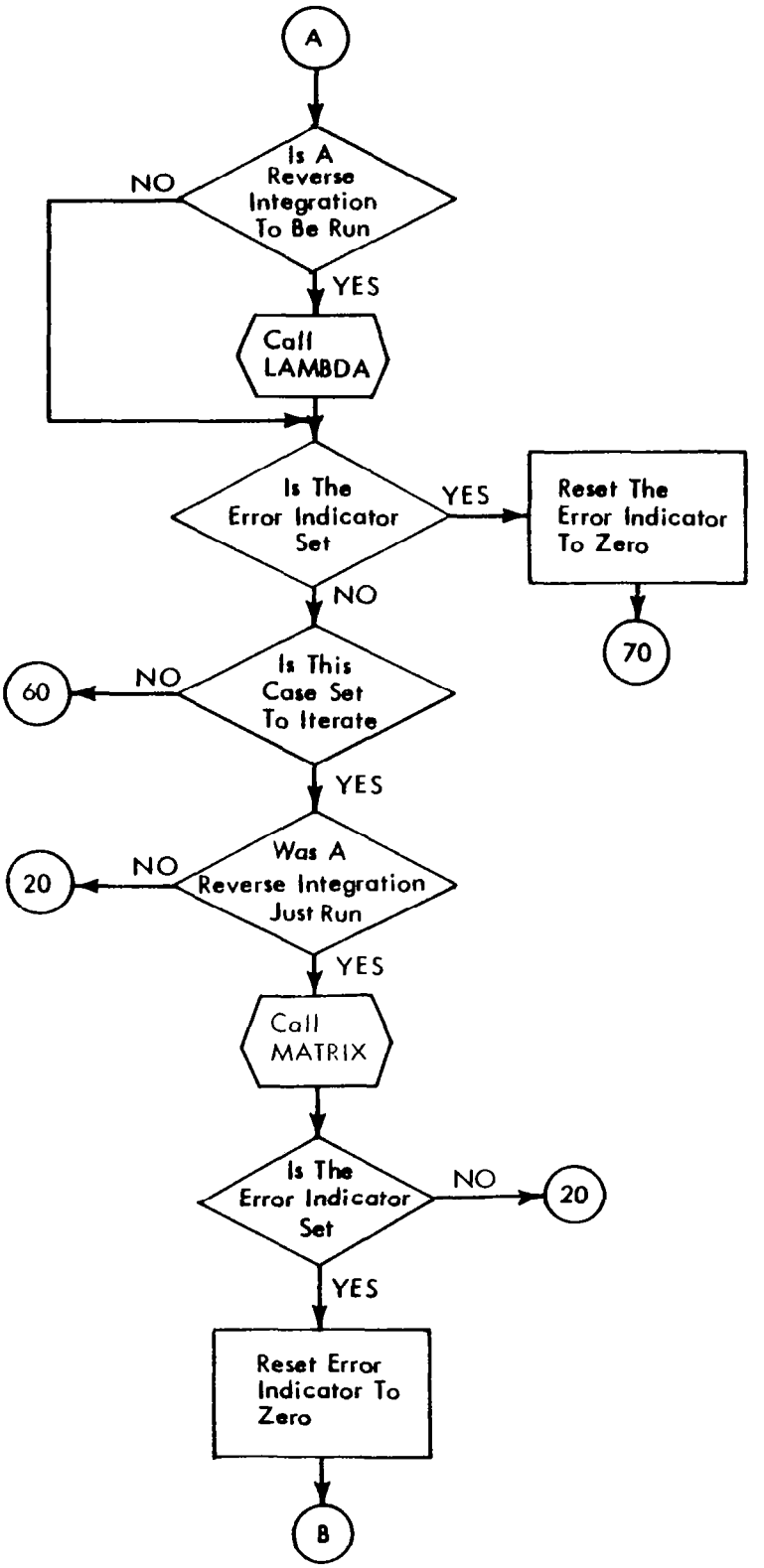
Subroutines Called

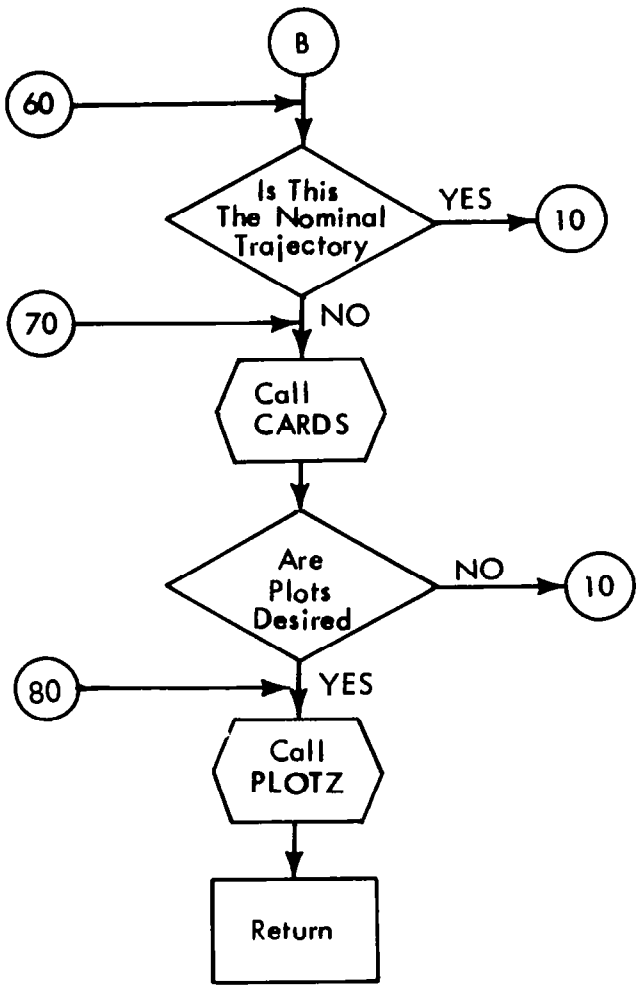
ATMOS CARDS EXEC INITAL MATRIX LAMBDA PLOTZ

Storage Used

206 cells


Call INITIAL
10
Call ATMOS
If Last Valid Forward Traj. L(6)=-1
20
Call EXEC
Is The Error Indicator Set
NO
YES
Is This The Nominal Trajectory
NO
70
YES
Is Only A Forward Trajectory Desired
YES
NO
Call Matrix
A
Is The Closed Loop Option Selected
NO
YES
Are Plots Of The Nominal Desired
NO
YES
80
Is The Punch Option Selected
NO
YES
70


A
Is A Reverse Integration To Be Run
NO
YES
Call LAMBDA
Is The Error Indicator Set
YES
Reset The Error Indicator To Zero
70
NO
Is This Case Set To Iterate
NO
60
YES
Was A Reverse Integration Just Run
NO
20
YES
Call MATRIX
Is The Error Indicator Set
NO
20
YES
Reset Error Indicator To Zero
B


B
60
Is This The Nominal Trajectory
YES
10
70
NO
Call CARDS
Are Plots Desired
NO
10
80
YES
Call PLOTZ
Return

LOOK1D — One-Dimensional Table Lookup

Purpose

LOOK1D provides a table lookup technique for tables that have one independent variable and NDV dependent variables.

Method

Given a value for the independent variable, the subroutine will furnish the dependent variable values and their partial derivatives with respect to the independent variable. Extrapolation is performed if the table limits are exceeded.

The call is made to LOOK1D through a calling sequence.

Call LOOK1D (NDV, KK, XQ, Y1, Y2, Y3, Y4, Y5, S1, S2, S3, S4, S5)

where NDV = number of dependent variables

KK = location of the first entry in the tables, NDS (table no.)

XQ = value of the independent variable

Y1, Y2, Y3, Y4, Y5 = values of the dependent variables

S1, S2, S3, S4, S5 = values of the slopes of the dependent variables with respect to the independent variable

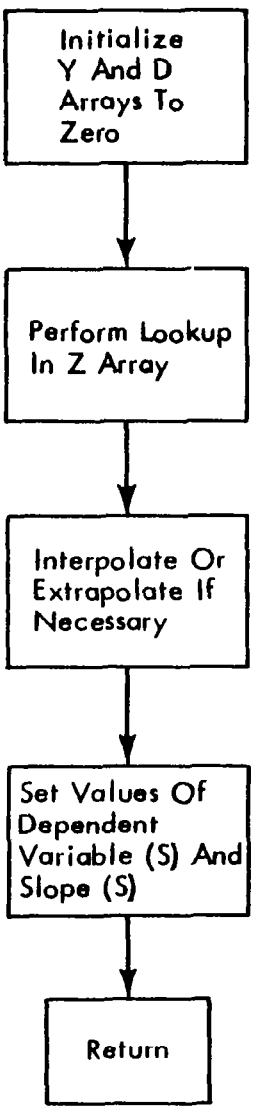
Assumptions and Limitations

1) The lookup method is based on linear interpolation.

2) The maximum number of dependent variables (NDV) is five.

Storage Used

229 cells


Initialize Y And D Arrays To Zero
Perform Lookup In Z Array
Interpolate Or Extrapolate If Necessary
Set Values Of Dependent Variable (S) And Slope (S)
Return

LOOK3D — 3D Table Lookup

Purpose

LOOK3D provides a method for linearly interpolating in a table to obtain a dependent variable as a function of up to three independent variables, i.e., $W = F(X, Y, Z)$.

Method

Given the three values of the independent variables, the routine calculates the dependent variable and the slopes (derivatives) of the dependent variable with respect to the three independent variables. Linear extrapolation is performed in cases where the table limits are exceeded.

The call to LOOK3D is made through a calling sequence.

LOOK3D (NTAB, ZIND, XIND, AIND, YY, SLZ SLY, SLX)

where

NTAB = table number

ZIND = value of the independent variable out of the plane

XIND = value of the independent variable between curves on a given plane

AIND = value of the independent variable that is the abscissa of the curves

YY = value of the dependent variable

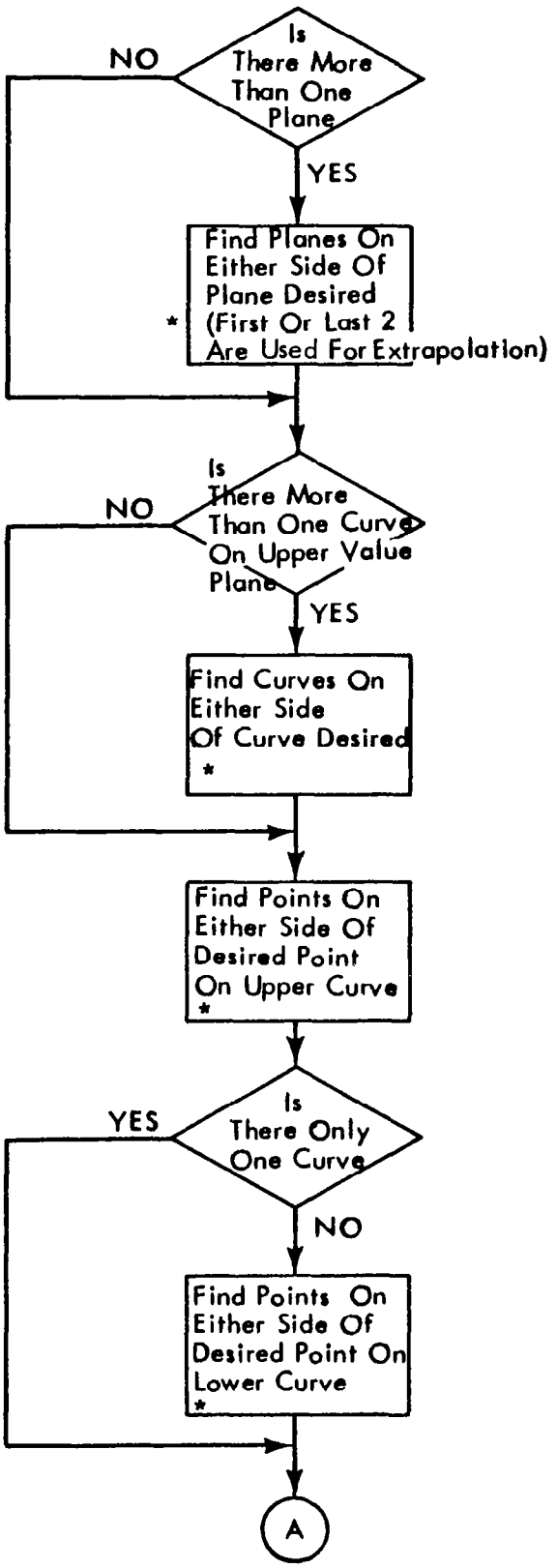
SLZ, SLY, SLX = slopes of the dependent variable corresponding to ZIND, XIND, AND AIND, respectively

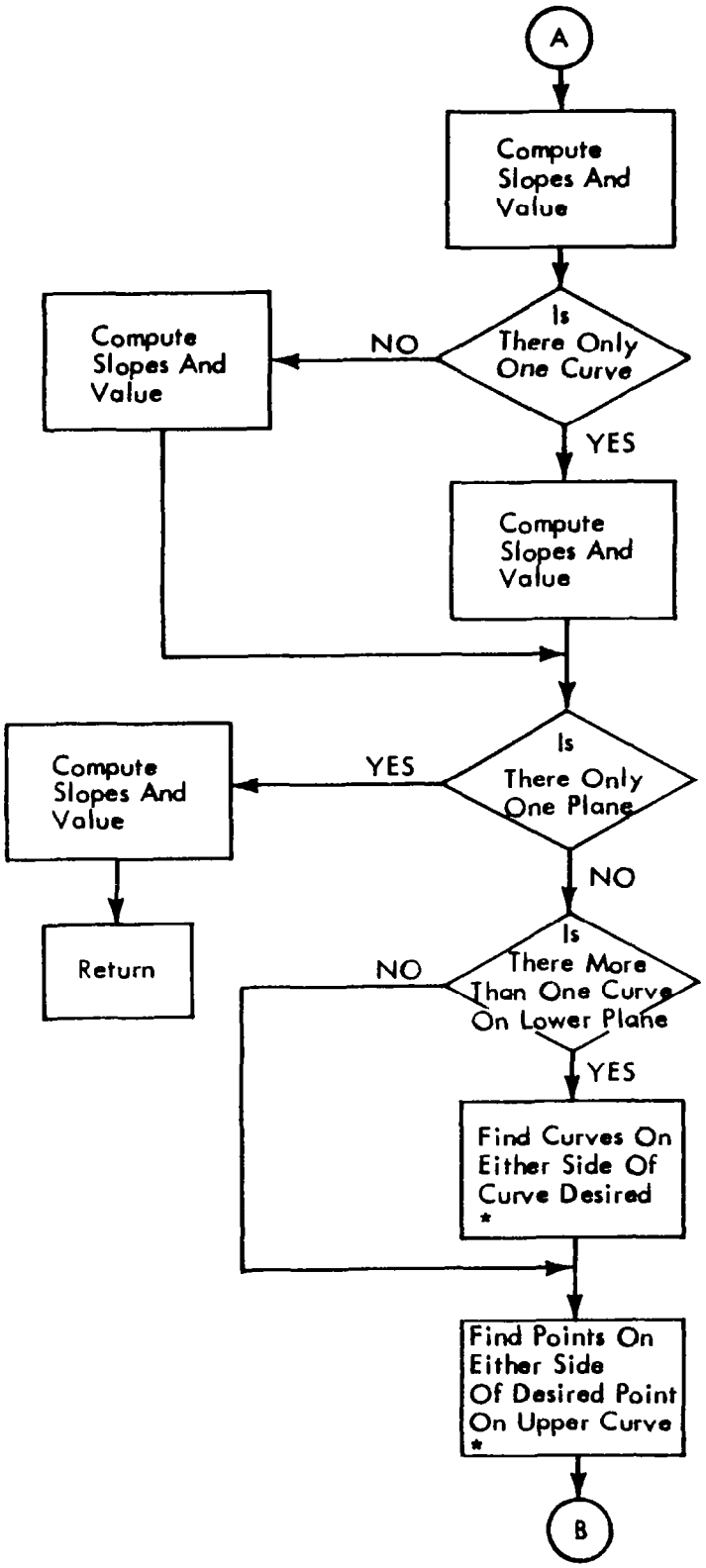
Assumptions and Limitations

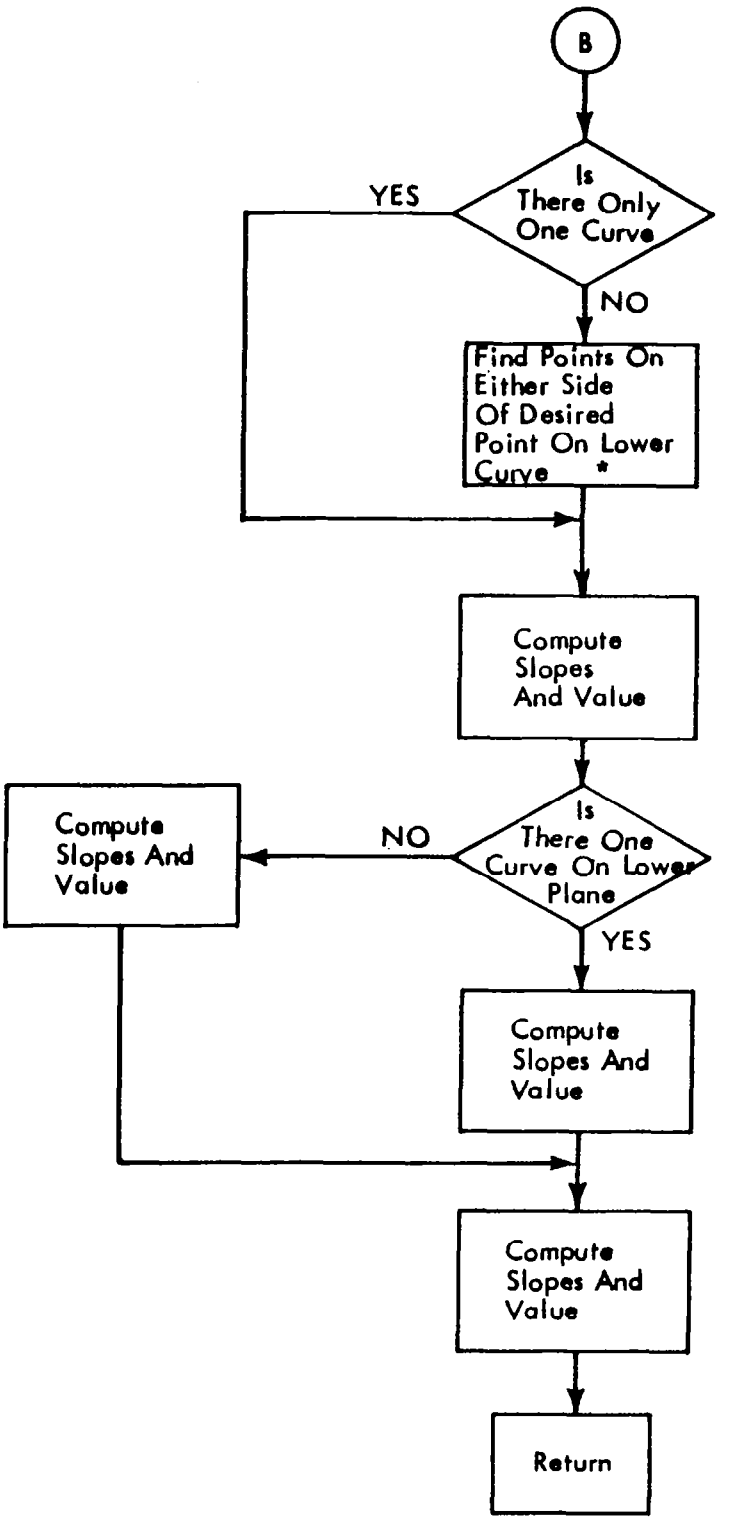
The dependent variable is obtained from the input data by linear interpolation and the slopes by a forward difference technique. The lookup package can be used to obtain the dependent variable as a function of one or two independent variables also.

Storage Used

617 cells


Is There More Than One Plane
NO
YES
Find Planes On Either Side Of Plane Desired
* (First Or Last 2 Are Used For Extrapolation)
Is There More Than One Curve On Upper Value Plane
NO
YES
Find Curves On Either Side Of Curve Desired *
Find Points On Either Side Of Desired Point On Upper Curve *
Is There Only One Curve
YES
NO
Find Points On Either Side Of Desired Point On Lower Curve *
A


A
Compute Slopes And Value
Is There Only One Curve
NO
Compute Slopes And Value
YES
Compute Slopes And Value
Is There Only One Plane
YES
Compute Slopes And Value
Return
NO
Is There More Than One Curve On Lower Plane
NO
YES
Find Curves On Either Side Of Curve Desired *
Find Points On Either Side Of Desired Point On Upper Curve *
B


B
Is There Only One Curve
YES
NO
Find Points On Either Side Of Desired Point On Lower Curve *
Compute Slopes And Value
Is There One Curve On Lower Plane
NO
YES
Compute Slopes And Value
Compute Slopes And Value
Compute Slopes And Value
Return

MATOUT — Matrix Output

Purpose

MATOUT prints convergence information that appears following each trial or valid step. The printout includes information such as current end-point values, constraint travel indicators, constraint changes, constraint tolerances, predicted constraint changes, step size coefficients, and other data.

Information not in COMMON is transmitted between MATRIX and MATOUT through the calling sequence

CALL MATOUT (DBETA, DPSI, DPSIP, IMAJ, INDTVL, PSIBWD, PSIFWD, PSIK, PSIKTR, PSINL, XLAMDX, XX)

where:

DBETA = the change in constraints modified by the end point changes due to the variation of initial conditions

DPSI = actual change in constraints measured from a previous valid step

DPSIP = predicted change in the constraint end points

IMAJ = value of the majority vote

INDTVL = constraint travel indicator

PSIBWD = nondimensional allowable change of a constraint in the direction away from the desired constraint value

PSIFWD = nondimensional allowable change of a constraint in the direction of the desired constraint value

PSIK = constraint step size coefficients

PSIKTR = constraint step size coefficients based on maximum permissible constraint travel

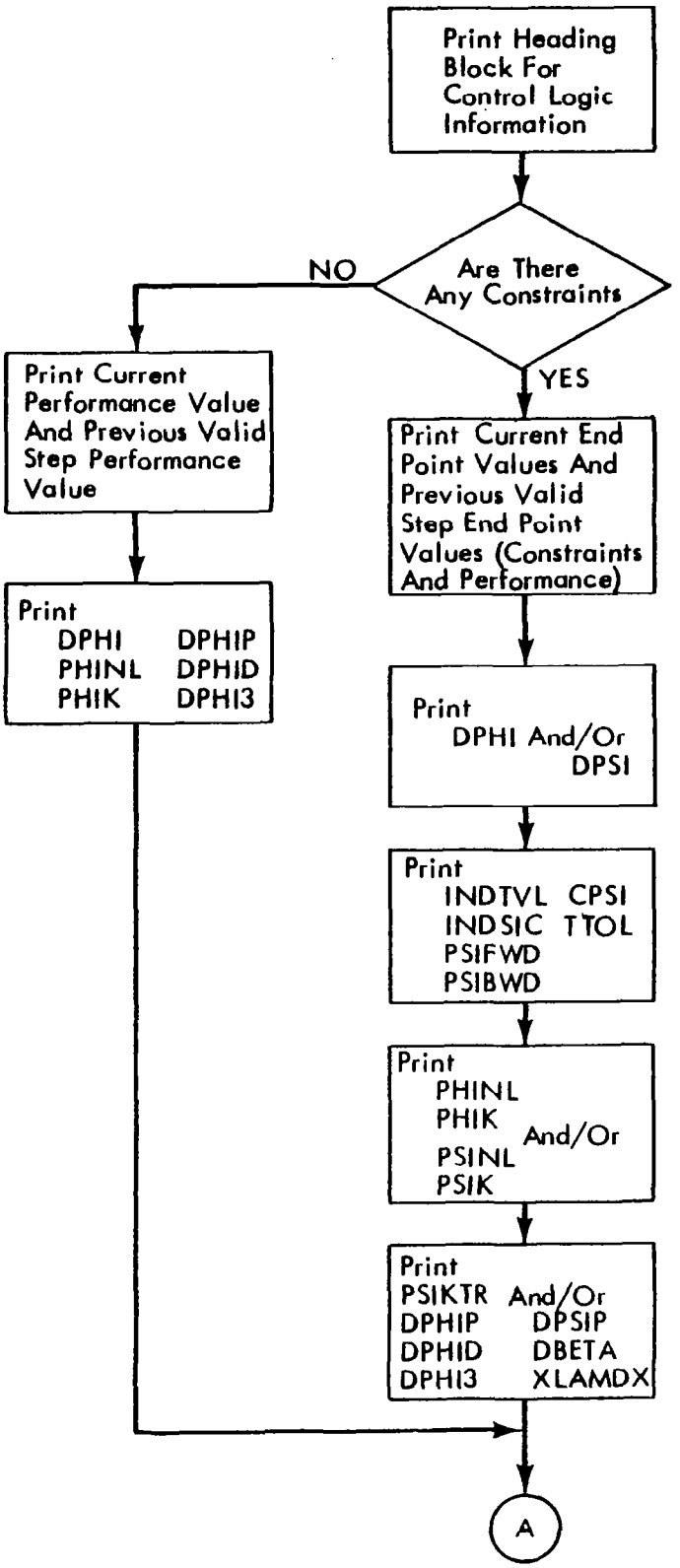
PSINL = constraint nonlinearities

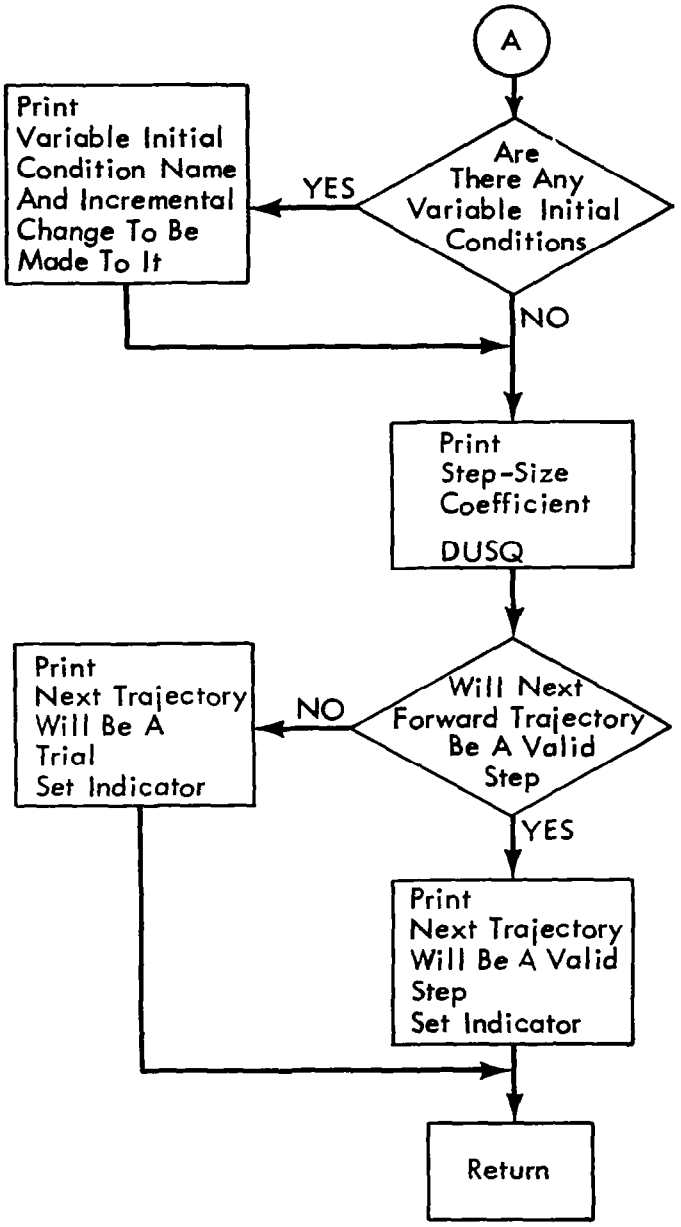
XLAMDX = predicted change in the constraints due to variable initial conditions.

XX = array of constraint end points

Storage Used

1202 cells


Print Heading Block For Control Logic Information
Are There Any Constraints
NO
YES
Print Current Performance Value And Previous Valid Step Performance Value
Print Current End Point Values And Previous Valid Step End Point Values (Constraints And Performance)
Print
DPHI DPHIP
PHINL DPHID
PHIK DPHI3
Print
DPHI And/Or DPSI
Print
INDTVL CPSI
INDSIC TTOL
PSIFWD
PSIBWD
Print
PHINL
PHIK
PSINL
PSIK
And/Or
Print
PSIKTR And/Or
DPHIP DPSIP
DPHID DBETA
DPHI3 XLAMDX
A


A
Are There Any Variable Initial Conditions
YES
Print Variable Initial Condition Name And Incremental Change To Be Made To It
NO
Print Step-Size Coefficient DUSQ
Will Next Forward Trajectory Be A Valid Step
NO
Print Next Trajectory Will Be A Trial Set Indicator
YES
Print Next Trajectory Will Be A Valid Step Set Indicator
Return

MATRIX — Basic Steepest Ascent Subroutine

Purpose

MATRIX is called at the end of every valid trajectory, reverse integration, and trial trajectory. After a valid trajectory, MATRIX calls VALID to examine the results (see VALID). After a reverse integration, MATRIX performs the following tasks:

1) Recovers the I matrix from the XB array
2) Calls MATRX2 to invert $I_{\psi\psi}$ matrix
3) Computes minimum allowable DUSQ
4) Calls VARIC to compute new DELX array
5) Updates iteration counter (ITC)
6) Writes arrays required for trials on KPAR unit for temporary storage
7) Computes Lagrange multipliers
8) Calls UCALC to update control history
9) Calls MATOUT to print convergence information

Following each trial, MATRIX performs the following tests:

1) Reads arrays from KPAR unit written after the previous reverse integration
2) Examines the try to see if it passes the majority vote test. If the test fails, a new trial is attempted
3) Calls KCALC to determine the step size coefficient for the next rajectory
4) Performs steps 7, 8, and 9 above.

Method

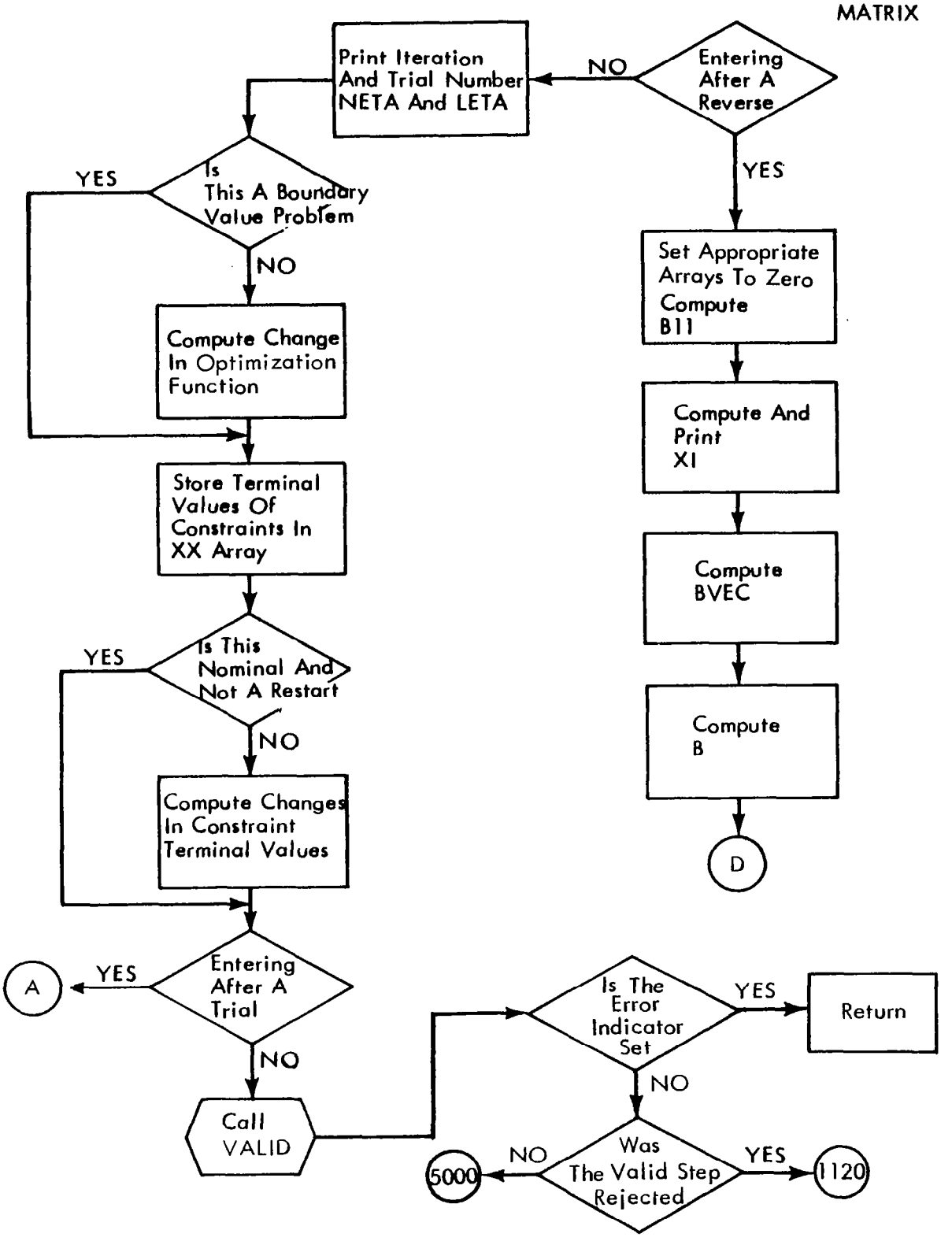
The steepest ascent method and the convergence logic used in MATRIX are explained in detail in the analytical development section and appendix C

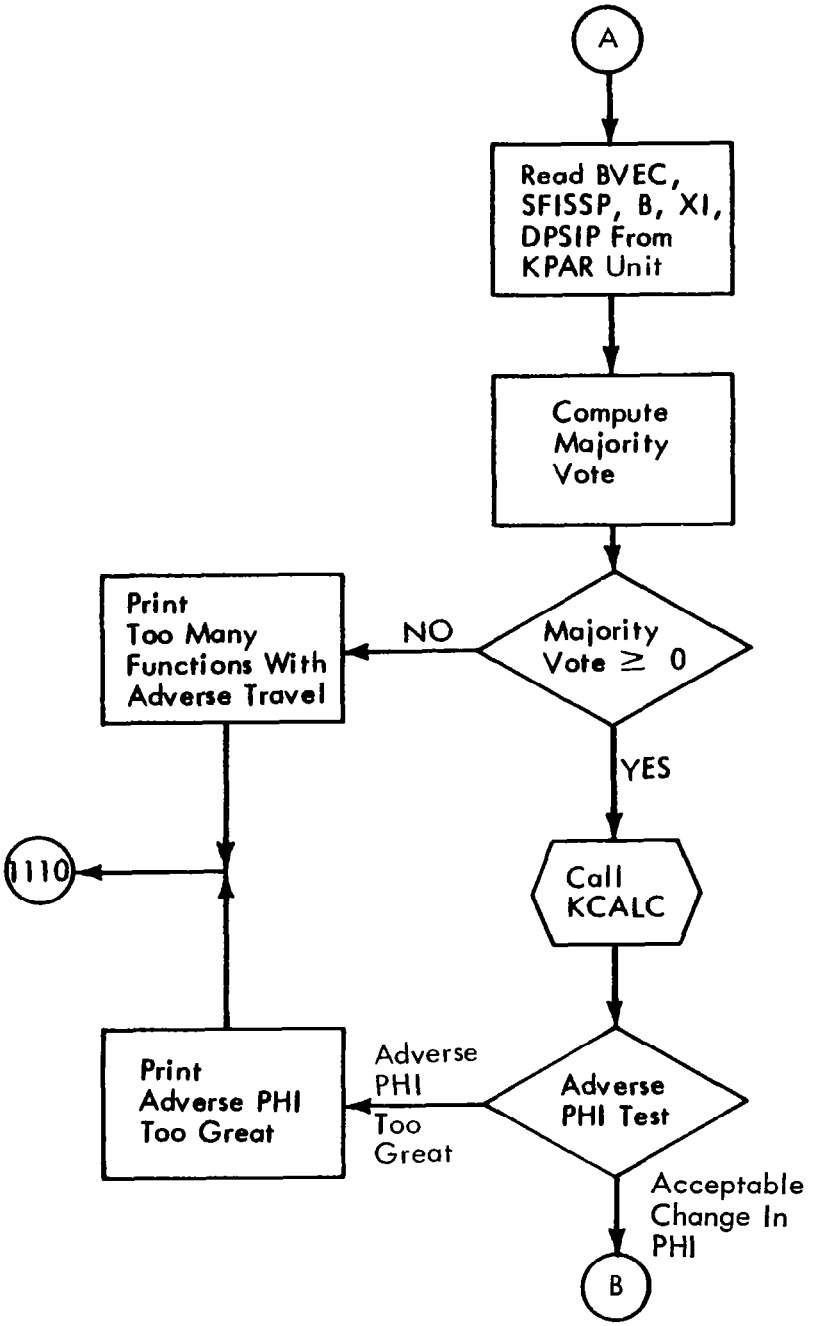
Subroutines Called

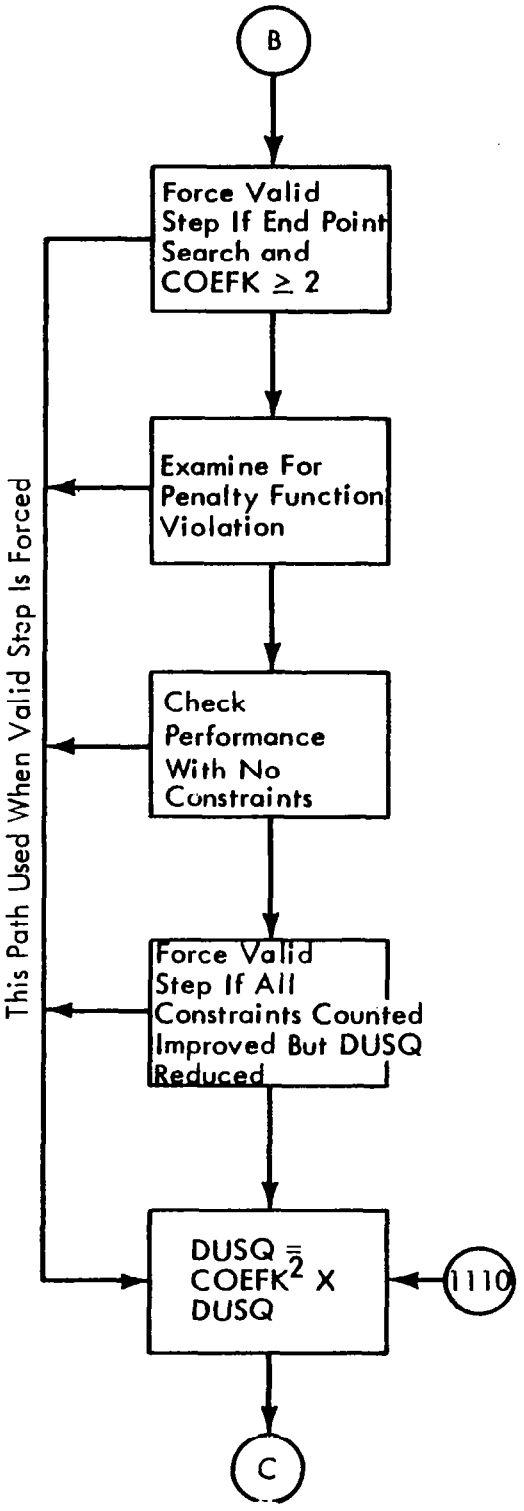
KCALC	MATRX2	VALID
MATOUT	UCALC	VARIC

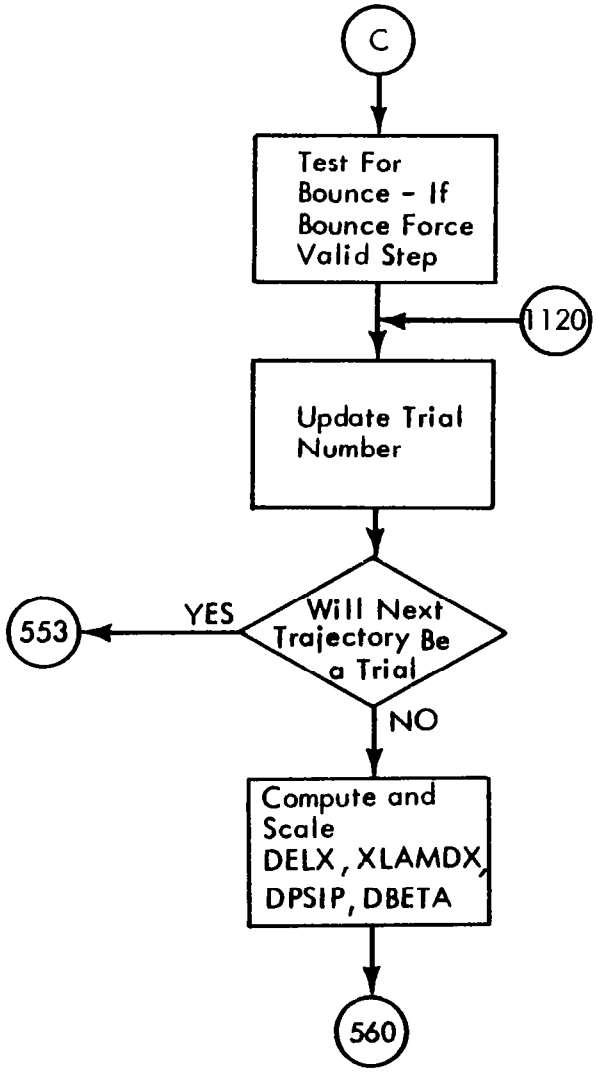
Storage Used

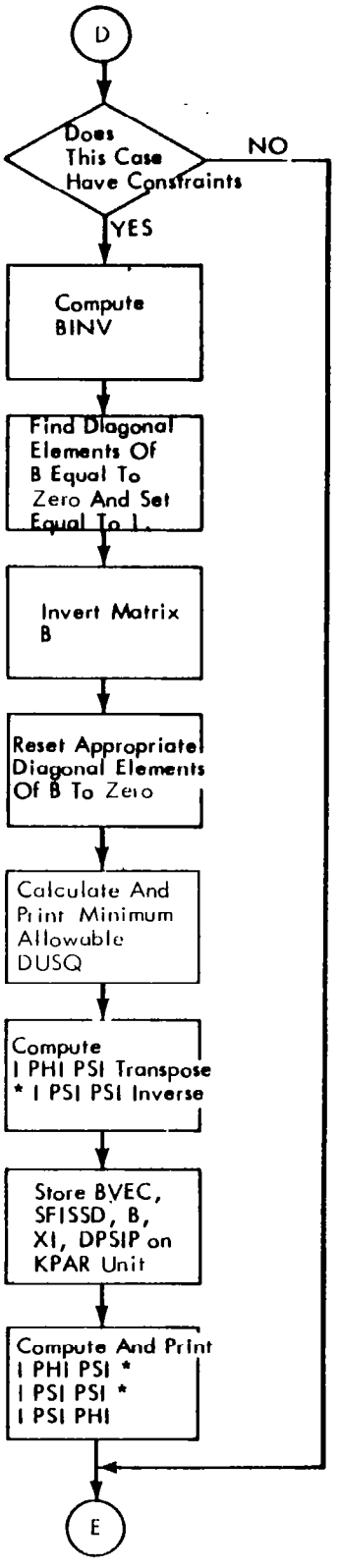
3454 cells

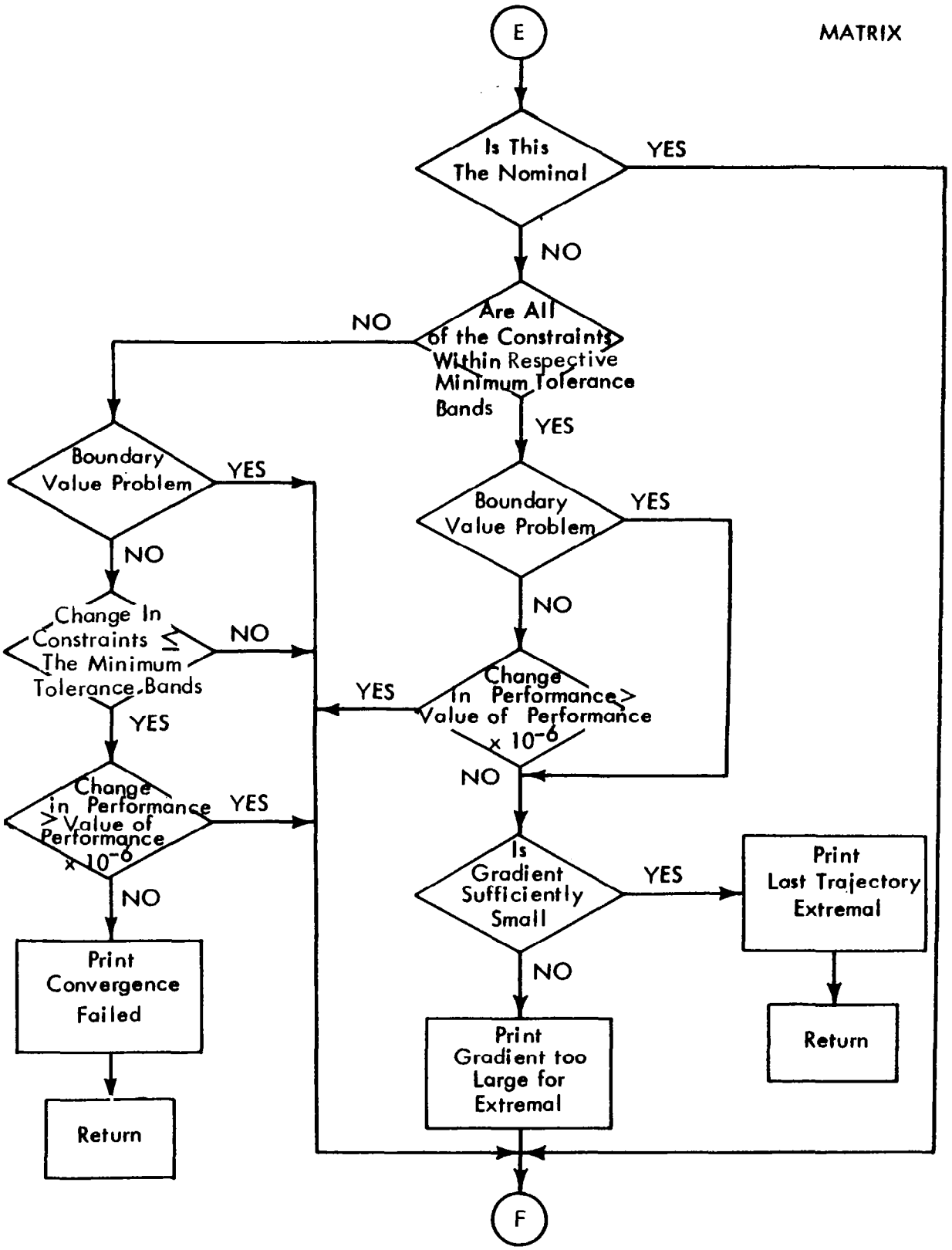

Print Iteration And Trial Number NETA And LETA
NO
Entering After A Reverse
YES
Is This A Boundary Value Problem
YES
NO
Compute Change In Optimization Function
Store Terminal Values Of Constraints In XX Array
Is This Nominal And Not A Restart
YES
NO
Compute Changes In Constraint Terminal Values
Set Appropriate Arrays To Zero Compute B11
Compute And Print XI
Compute BVEC
Compute B
D
A
YES
Entering After A Trial
NO
Call VALID
Is The Error Indicator Set
YES
Return
NO
Was The Valid Step Rejected
NO
5000
YES
1120

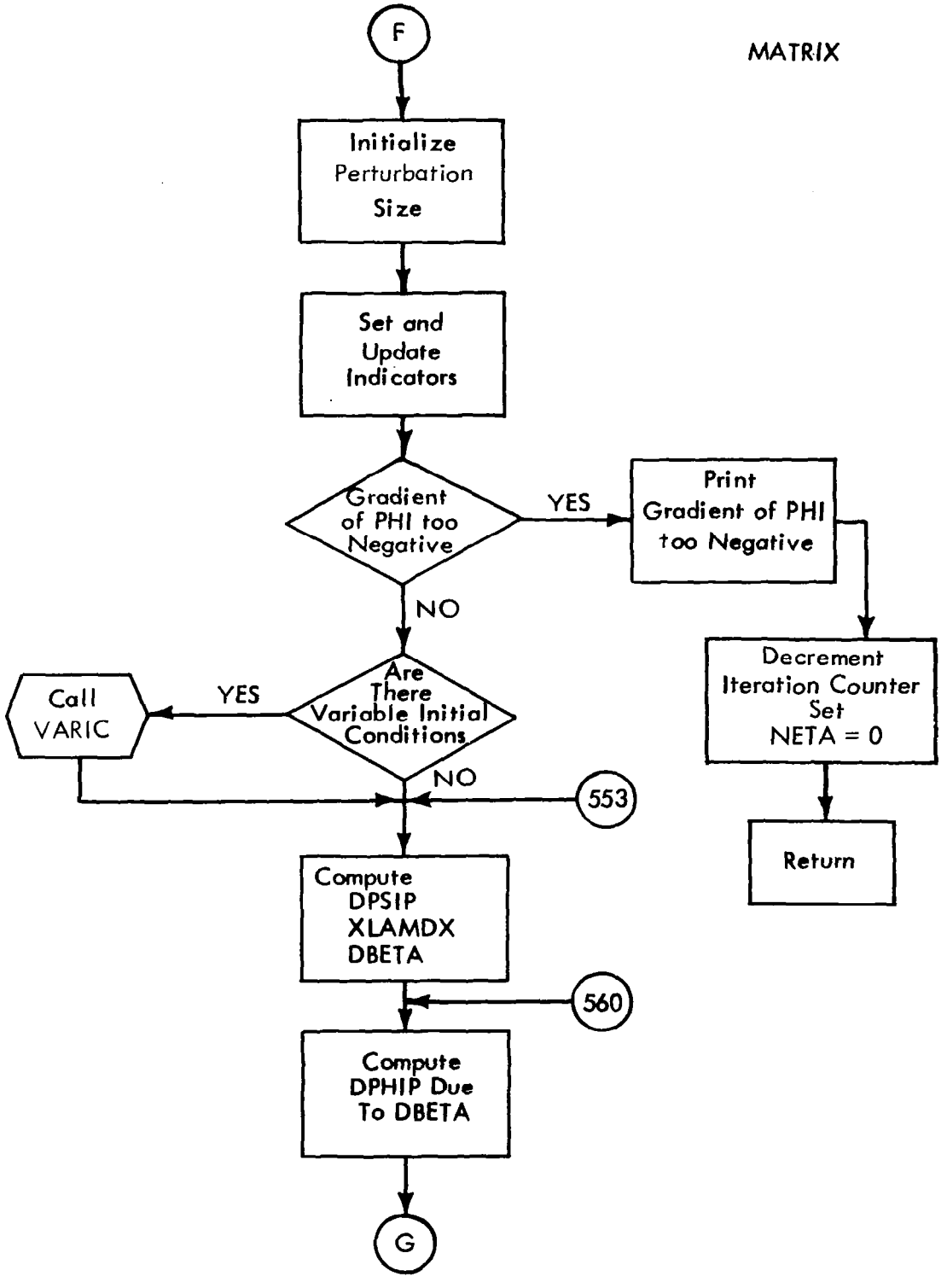

A
Read BVEC, SFISSP, B, XI, DPSIP From KPAR Unit
Compute Majority Vote
Majority Vote ≥ 0
NO
Print Too Many Functions With Adverse Travel
YES
1110
Call KCALC
Adverse PHI Test
Adverse PHI Too Great
Print Adverse PHI Too Great
Acceptable Change In PHI
B

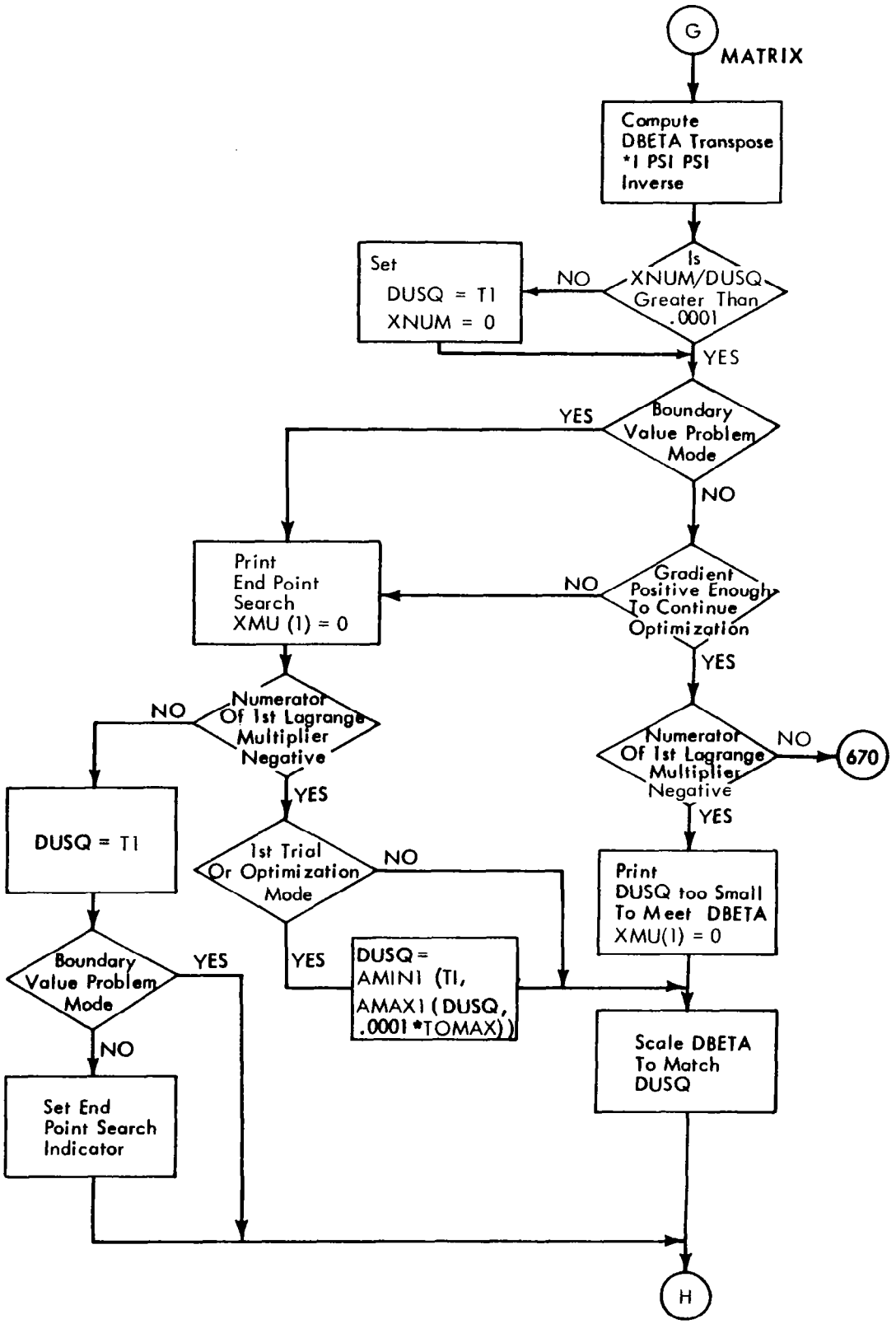

B
Force Valid Step If End Point Search and COEFK ≥ 2
Examine For Penalty Function Violation
Check Performance With No Constraints
Force Valid Step If All Constraints Counted Improved But DUSQ Reduced
DUSQ = COEFK² X DUSQ
1110
C
This Path Used When Valid Step Is Forced

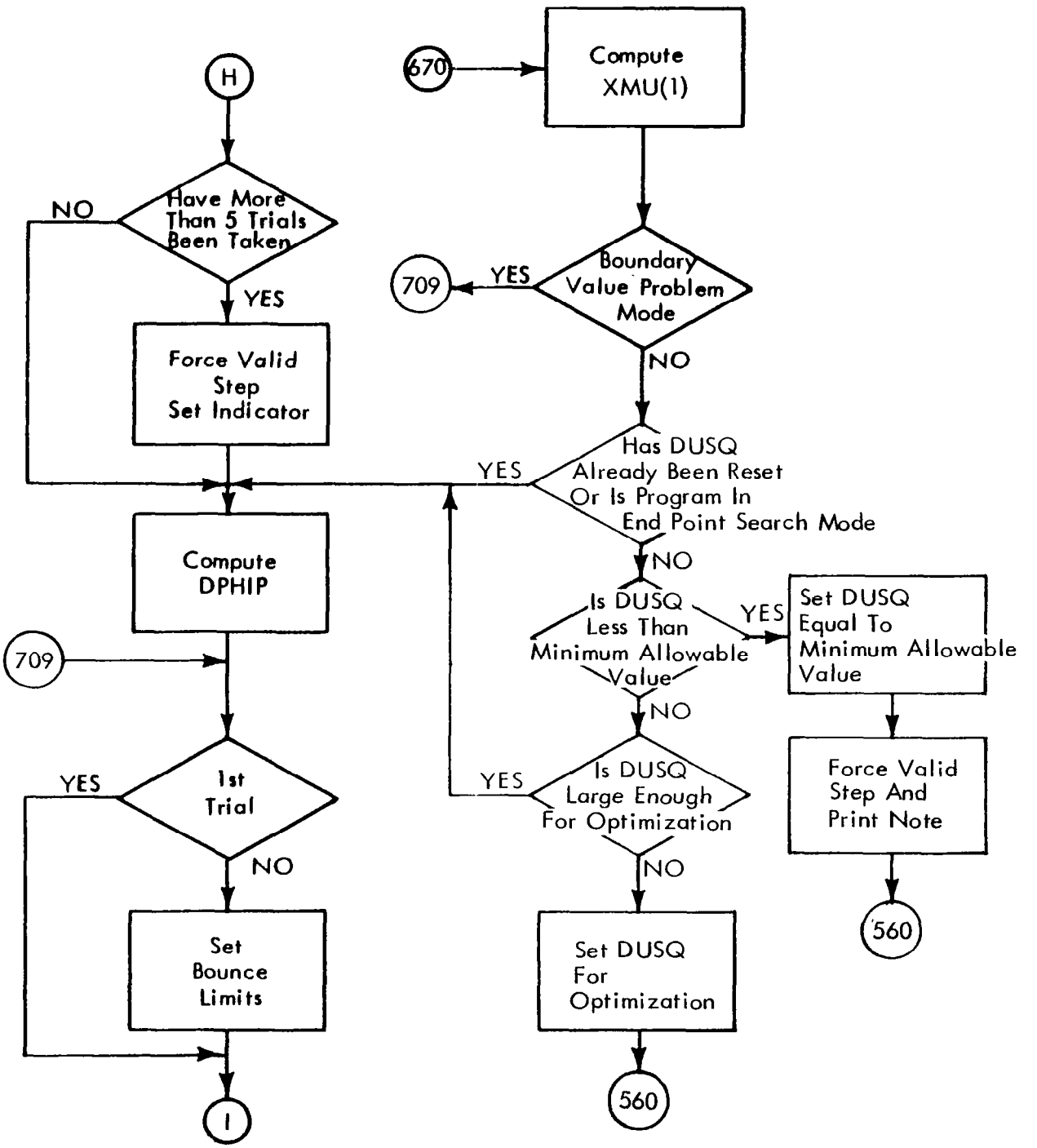

C
Test For Bounce - If Bounce Force Valid Step
1120
Update Trial Number
Will Next Trajectory Be a Trial
YES
553
NO
Compute and Scale DELX, XLAMDX, DPSIP, DBETA
560

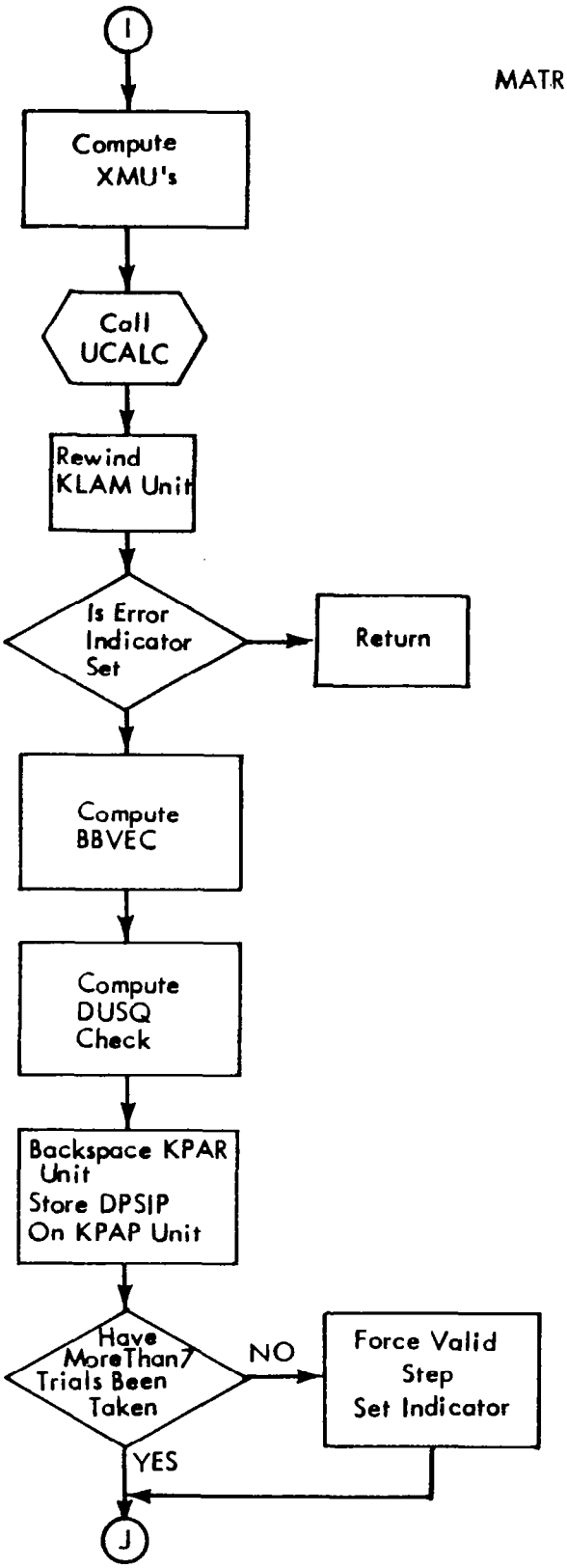

D
Does This Case Have Constraints
NO
YES
Compute BINV
Find Diagonal Elements Of B Equal To Zero And Set Equal To 1.
Invert Matrix B
Reset Appropriate Diagonal Elements Of B To Zero
Calculate And Print Minimum Allowable DUSQ
Compute I PHI PSI Transpose * I PSI PSI Inverse
Store BVEC, SFISSD, B, XI, DPSIP on KPAR Unit
Compute And Print I PHI PSI * I PSI PSI * I PSI PHI
E

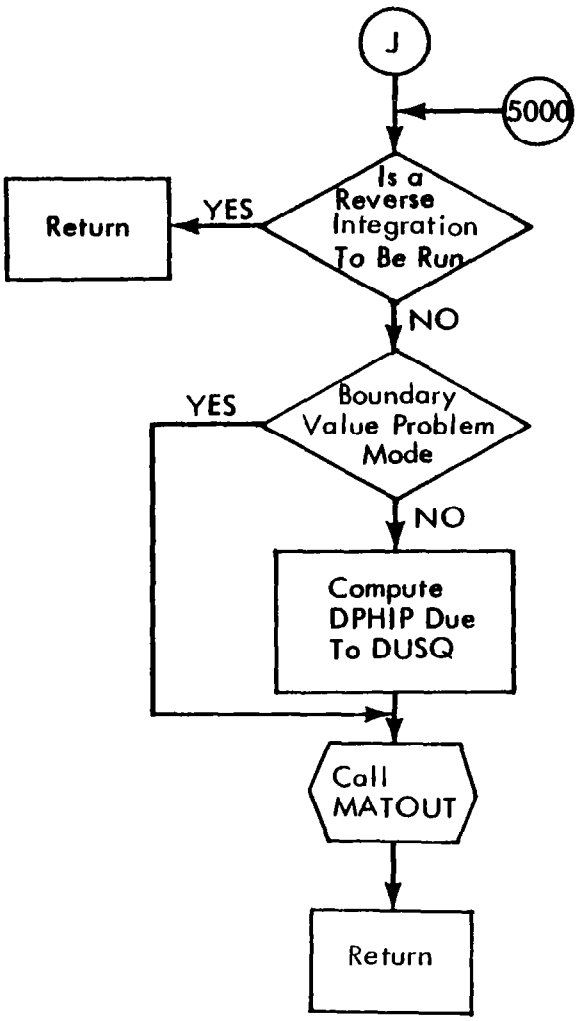

E
Is This The Nominal
YES
NO
Are All of the Constraints Within Respective Minimum Tolerance Bands
NO
YES
Boundary Value Problem
YES
NO
Change In Constraints ≤ The Minimum Tolerance Bands
NO
YES
Change in Performance ≥ Value of Performance x 10⁻⁶
YES
NO
Boundary Value Problem
YES
NO
Change In Performance ≥ Value of Performance x 10⁻⁶
YES
NO
Is Gradient Sufficiently Small
YES
NO
Print Last Trajectory Extremal
Return
Print Convergence Failed
Return
Print Gradient too Large for Extremal
F


F
Initialize Perturbation Size
Set and Update Indicators
Gradient of PHI too Negative
YES
Print Gradient of PHI too Negative
NO
Decrement Iteration Counter Set NETA = 0
Return
Are There Variable Initial Conditions
YES
Call VARIC
NO
553
Compute DPSIP XLAMDX DBETA
560
Compute DPHIP Due To DBETA
G


G
MATRIX
Compute DBETA Transpose *I PSI PSI Inverse
Is XNUM/DUSQ Greater Than .0001
NO
Set DUSQ = TI XNUM = 0
YES
Boundary Value Problem Mode
YES
NO
Print End Point Search XMU (1) = 0
Gradient Positive Enough To Continue Optimization
NO
YES
Numerator Of 1st Lagrange Multiplier Negative
NO
YES
Numerator Of 1st Lagrange Multiplier Negative
NO
670
YES
DUSQ = TI
1st Trial Or Optimization Mode
NO
YES
Print DUSQ too Small To Meet DBETA XMU(1) = 0
Boundary Value Problem Mode
YES
DUSQ = AMIN1 (TI, AMAX1 (DUSQ, .0001 *TOMAX))
NO
Scale DBETA To Match DUSQ
Set End Point Search Indicator
H


H
Have More Than 5 Trials Been Taken
NO
YES
Force Valid Step Set Indicator
Compute DPHIP
709
1st Trial
YES
NO
Set Bounce Limits
1
670
Compute XMU(1)
Boundary Value Problem Mode
709
YES
NO
Has DUSQ Already Been Reset Or Is Program In End Point Search Mode
YES
NO
Is DUSQ Less Than Minimum Allowable Value
YES
Set DUSQ Equal To Minimum Allowable Value
NO
Is DUSQ Large Enough For Optimization
YES
NO
Set DUSQ For Optimization
560
Force Valid Step And Print Note
560


I
Compute XMU's
Call UCALC
Rewind KLAM Unit
Is Error Indicator Set
Return
Compute BBVEC
Compute DUSQ Check
Backspace KPAR Unit
Store DPSIP On KPAP Unit
Have MoreThan7 Trials Been Taken
NO
Force Valid Step Set Indicator
YES
J


J
5000
Is a Reverse Integration To Be Run
YES
Return
NO
Boundary Value Problem Mode
YES
NO
Compute DPHIP Due To DUSQ
Call MATOUT
Return

MATRX2 — Matrix Inversion Subroutine

Purpose

MATRX2 computes the solution of a set of simultaneous linear equations, the inverse of a matrix, or the value of a determinant.

Method

The solution of a simultaneous linear equations follows from

$$B = AX$$

or solving for X

$$X = A^{-1} B$$

Thus, the inverse is required for the solution, and the determinant of A is required for the inverse of A. The matrix operations are all performed in double-precision. Matrix inversion uses the Gaussian elimination method.

The call to MATRX2 is made through a calling sequence

CALL MATRX2 (N, LN, LM, A, B, E, D, MM)

where
- N = the order of the matrix A
- LN = number of rows in the dimension statement for matrices A and B, i.e., A is an LN x LM matrix.
- LM = number of columns in matrix B
- A = array designation for matrix A
- B = array designation for matrix B
- E = an array used as temporary storage of a column matrix
- D = value of the determinant of A
  - if D = 0 the determinant is not computed
  - D = 1 the determinant is computed
  - D = a scale factor, the determinant equals the value of the determinant times the scale factor
- MM = error indicator
  - 1 if solution is successful
  - 2 if overflow occurred
  - 3 if matrix is singular

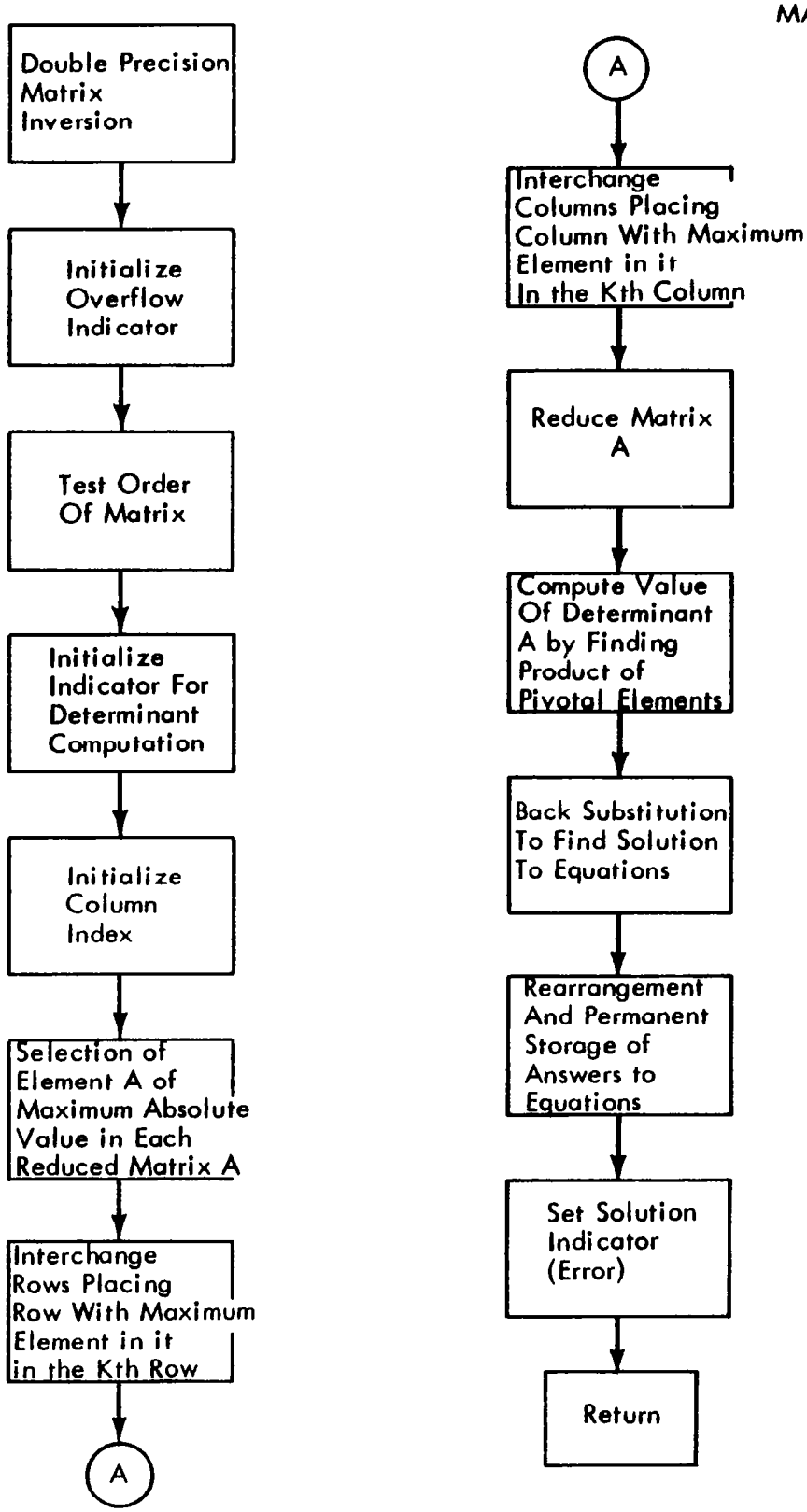
The answers or the X matrix replace the original A matrix.

Assumption and Limitations

The size of the problem that can be solved is limited only by machine storage space.

Storage Used

596 cells


Double Precision Matrix Inversion
Initialize Overflow Indicator
Test Order Of Matrix
Initialize Indicator For Determinant Computation
Initialize Column Index
Selection of Element A of Maximum Absolute Value in Each Reduced Matrix A
Interchange Rows Placing Row With Maximum Element in it in the Kth Row
A
A
Interchange Columns Placing Column With Maximum Element in it In the Kth Column
Reduce Matrix A
Compute Value Of Determinant A by Finding Product of Pivotal Elements
Back Substitution To Find Solution To Equations
Rearrangement And Permanent Storage of Answers to Equations
Set Solution Indicator (Error)
Return

PLAC — Placard Equation Evaluation

Purpose

Subroutine PLAC is called from FPROG for each selected enroute constraint and performs the following tasks.

1) Compares the constrained variable with the allowable limits from the input placard table
2) Computes the placard equation of motion value and updates the appropriate element of the F array
3) Stores the placard limit value and the slopes of the limit value with respect to its independent variables (obtained from the input placard table) in the S array.

Method

The call to PLAC is made through the following calling sequence:

CALL PLAC (II, ITAB, KTAB, A1, A2, A3, A4)

where
- II = F array index corresponding to the selected placard. II is computed by FPROG.
- ITAB = first table number containing the placard values for the selected placard. ITAB is computed by FPROG.
- KTAB = number of independent variables defining the input placard table. KTAB may be 1, 2, or 3, depending on the placard formulation.
- A1 = Value of the independent variable for a one dimensional table or the value of the "plane" for a three dimensional table.
- A2 = Value of the "curve" for a 3-D table. A2 = 0. for a 1-D table.
- A3 = Value of the abscissa ("point") for a 3-D table. A3 = 0. for a 1-D table.

As an example, the calling sequence for the $Q_{LIMIT} = F(t)$ placard is

CALL PLAC (II, ITAB, 1, T, 0., 0., AK(4)).

For the $N_{LIMIT} = F(H, M)$, the call is

CALL PLAC (II, ITAB, 2, 0., X(K2), AK(5), UI3).

Note that if the 3-D option is used for a 2-D table, the value of the "plane" is set to zero.

PLAC will set F(II) = 0. if the limit value is not exceeded or to (LIMIT - A4)^2 if the limit is exceeded.

PLAC also computes elements in the S array as follows:

S(I, 2) = limit value exceeded (max. or min.).

S(I, 3) = slope of limit value with respect to A1 (from the input placard table).

S(I, 4) = slope of limit value with respect to A2.

S(I, 5) = slope of limit value with respect to A3.

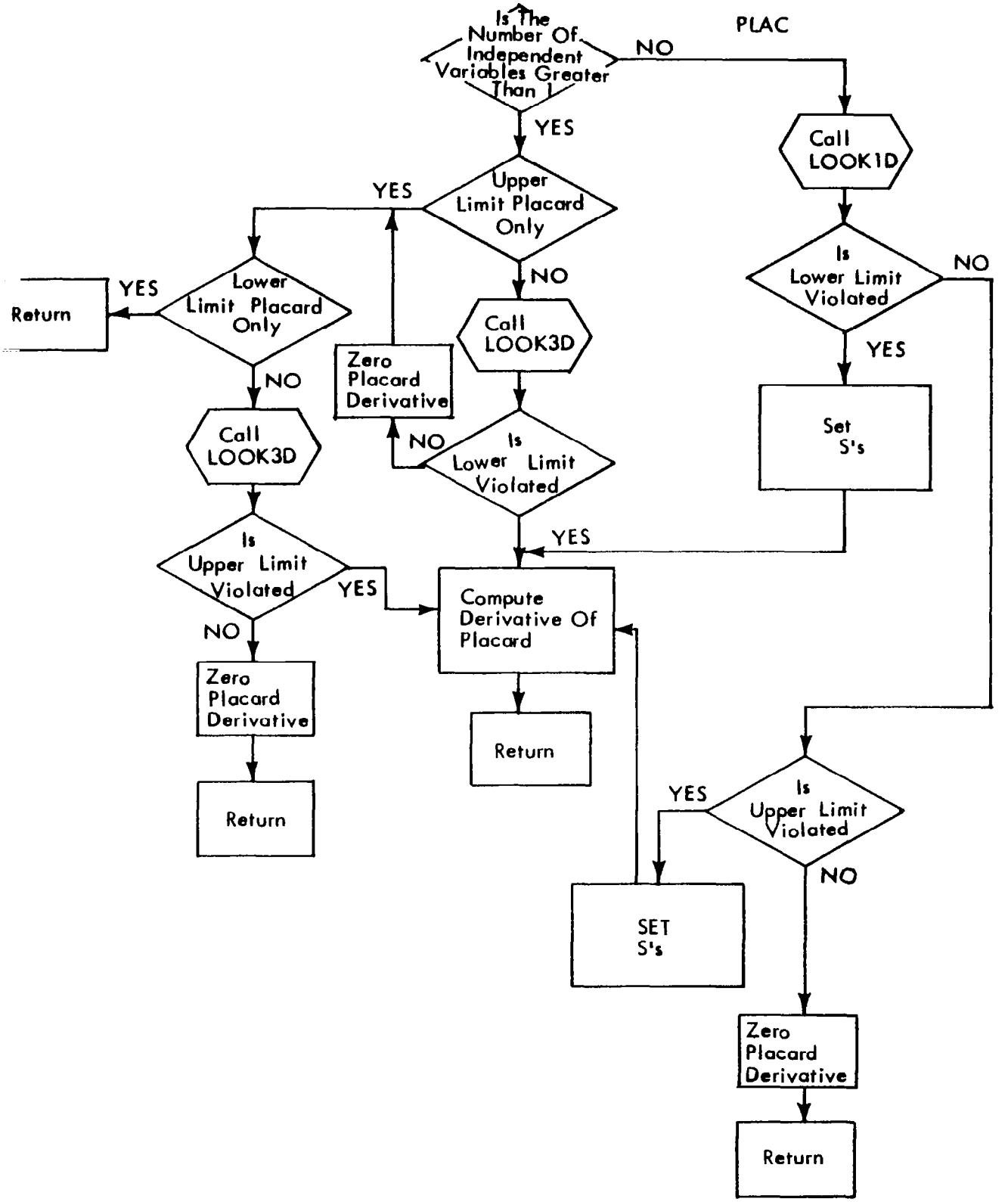
I is the internal index for the selected placard (I = 1, NPF) and is computed by PLAC.

Subroutines Called

LOOK1D LOOK3D

Storage Used

216 cells


Is The Number Of Independent Variables Greater Than 1
NO
PLAC
YES
Call LOOK1D
Upper Limit Placard Only
YES
Is Lower Limit Violated
NO
YES
Lower Limit Placard Only
YES
Return
NO
Call LOOK3D
Zero Placard Derivative
Set S's
Call LOOK3D
NO
Is Lower Limit Violated
Is Upper Limit Violated
YES
YES
Compute Derivative Of Placard
NO
Zero Placard Derivative
Return
Return
YES
Is Upper Limit Violated
SET S's
NO
Zero Placard Derivative
Return

PLOTZ — Plotting Subroutine

Purpose

PLOTZ controls the plotting operations for the Orthomat and SC 4020 plotters. Operations for scales, headings, titles, etc., are included.

Limitations

1) The maximum number of plots is limited to seven
2) Six curves maximum per plot
3) Multiple-data cases cannot be plotted
4) X- and Y-axes annotation occurs every 2 cm
5) Titles consist of the following:

   LINE 1 -

   DEP. VAR. NAME VS. IND. VAR. NAME

   LINE 2 -

   The first 72 characters taken from title card on the data deck.

Subroutines Called

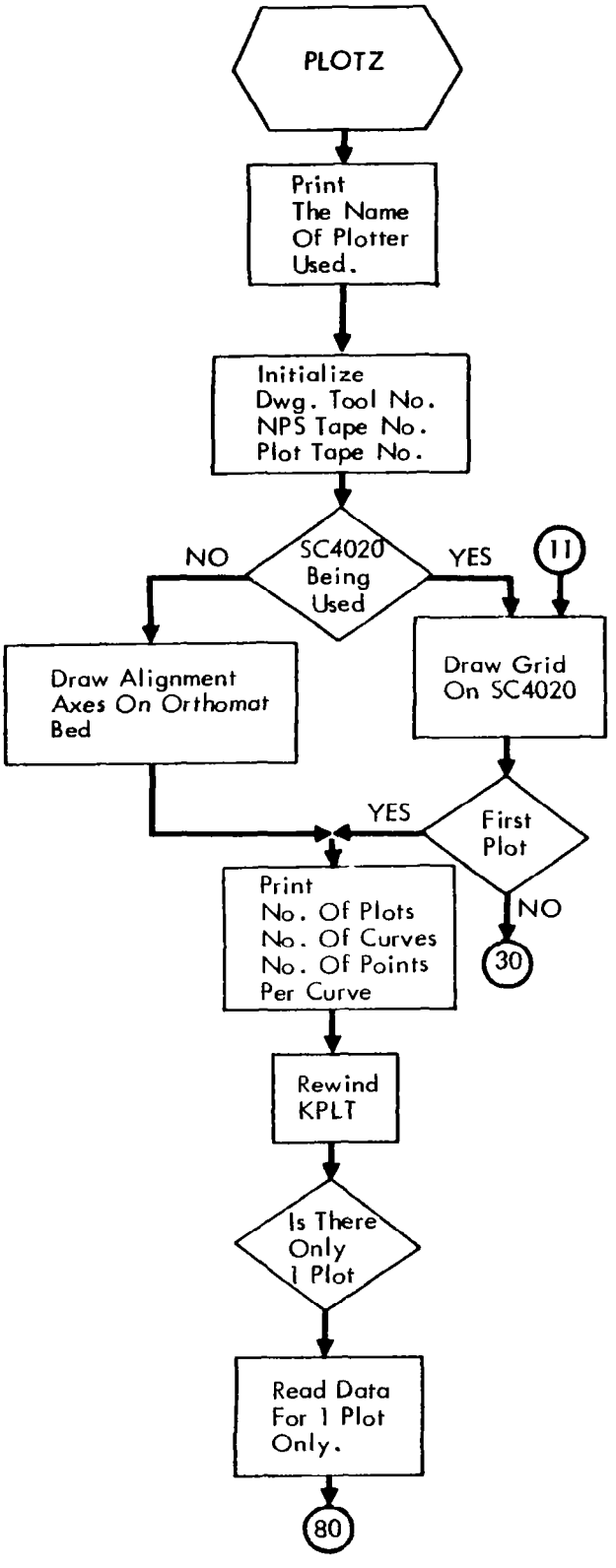
SKALZ

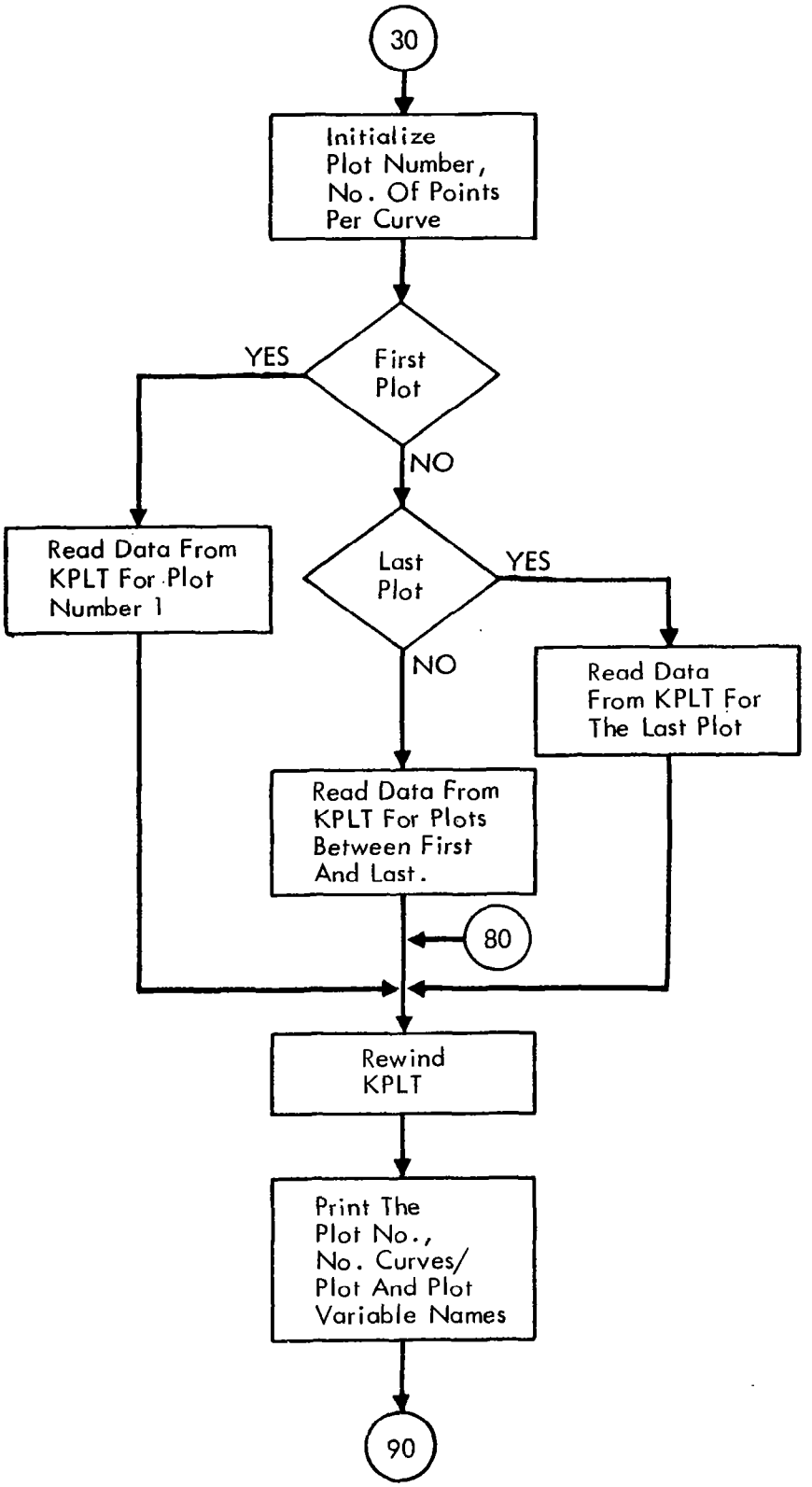
Boeing numerical plotting system (NPS) subroutines called

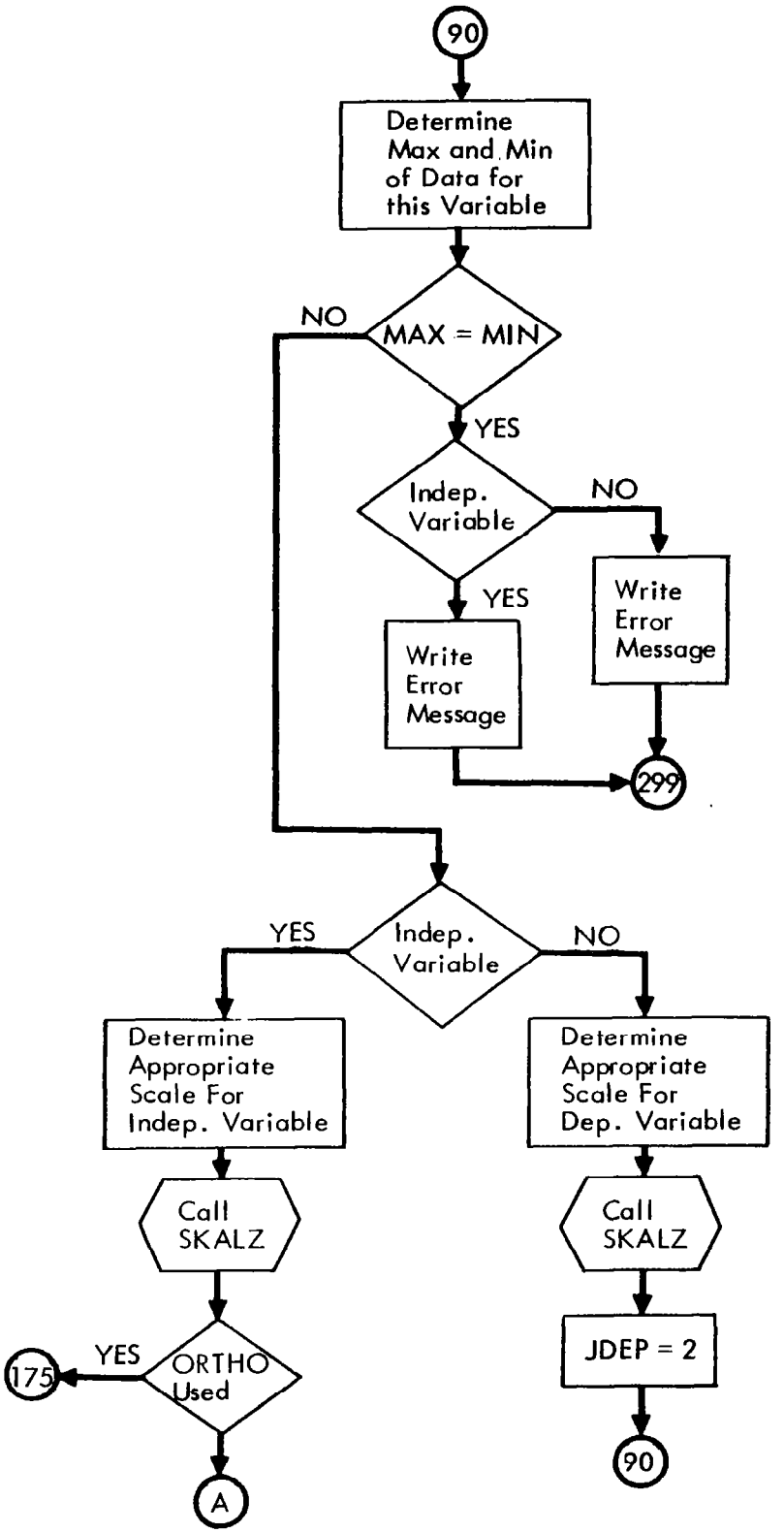
CAMERA	NOSLIB	TITLEB
ORTHPP	AXLILI	BORDER
SLLILI	NWPAGE	SCPP
PAUSE	GDLILI	NOSLIL
PSLILI	FORM	

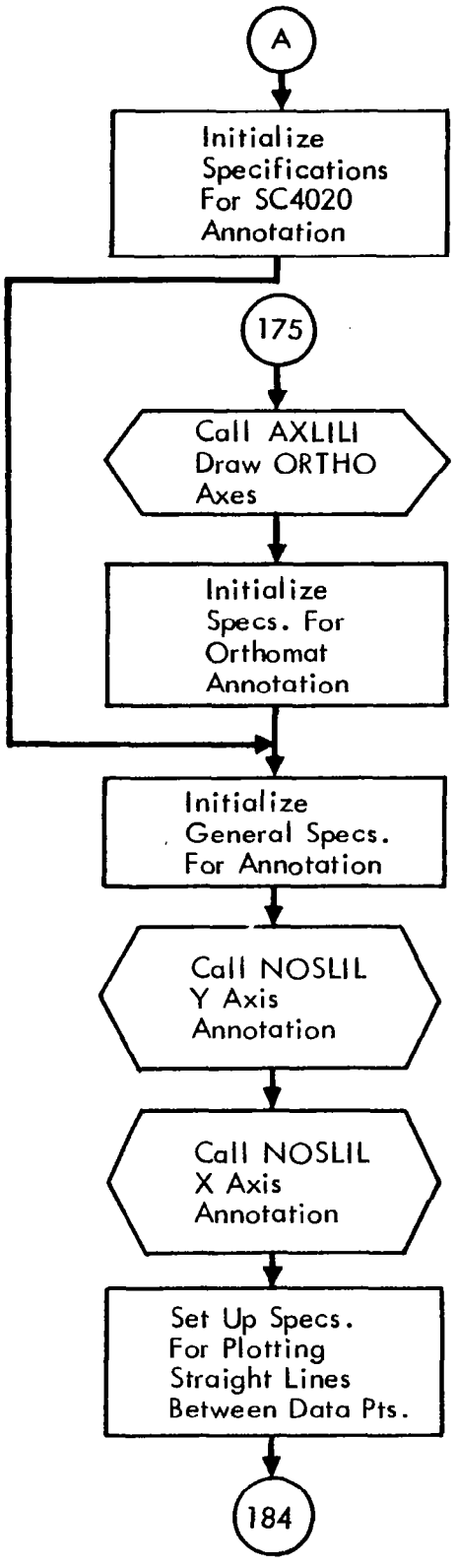
Storage Used

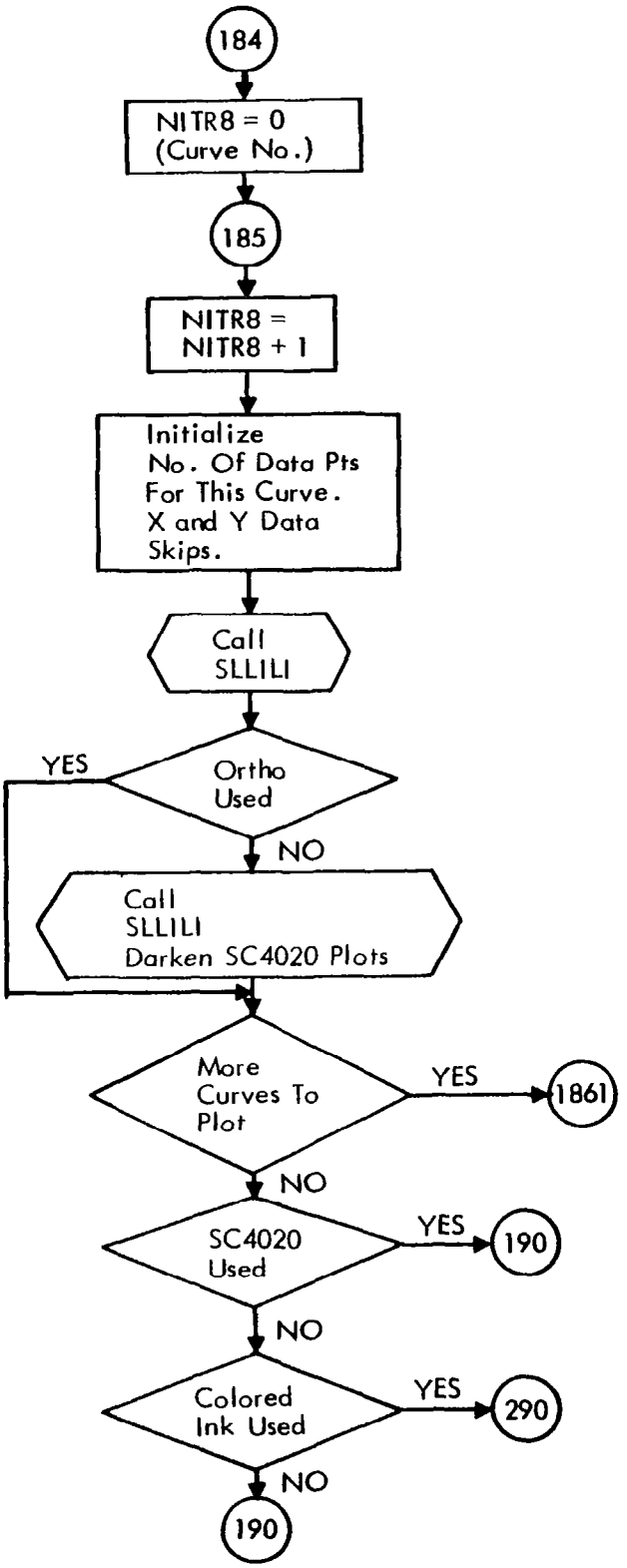
1436 cells

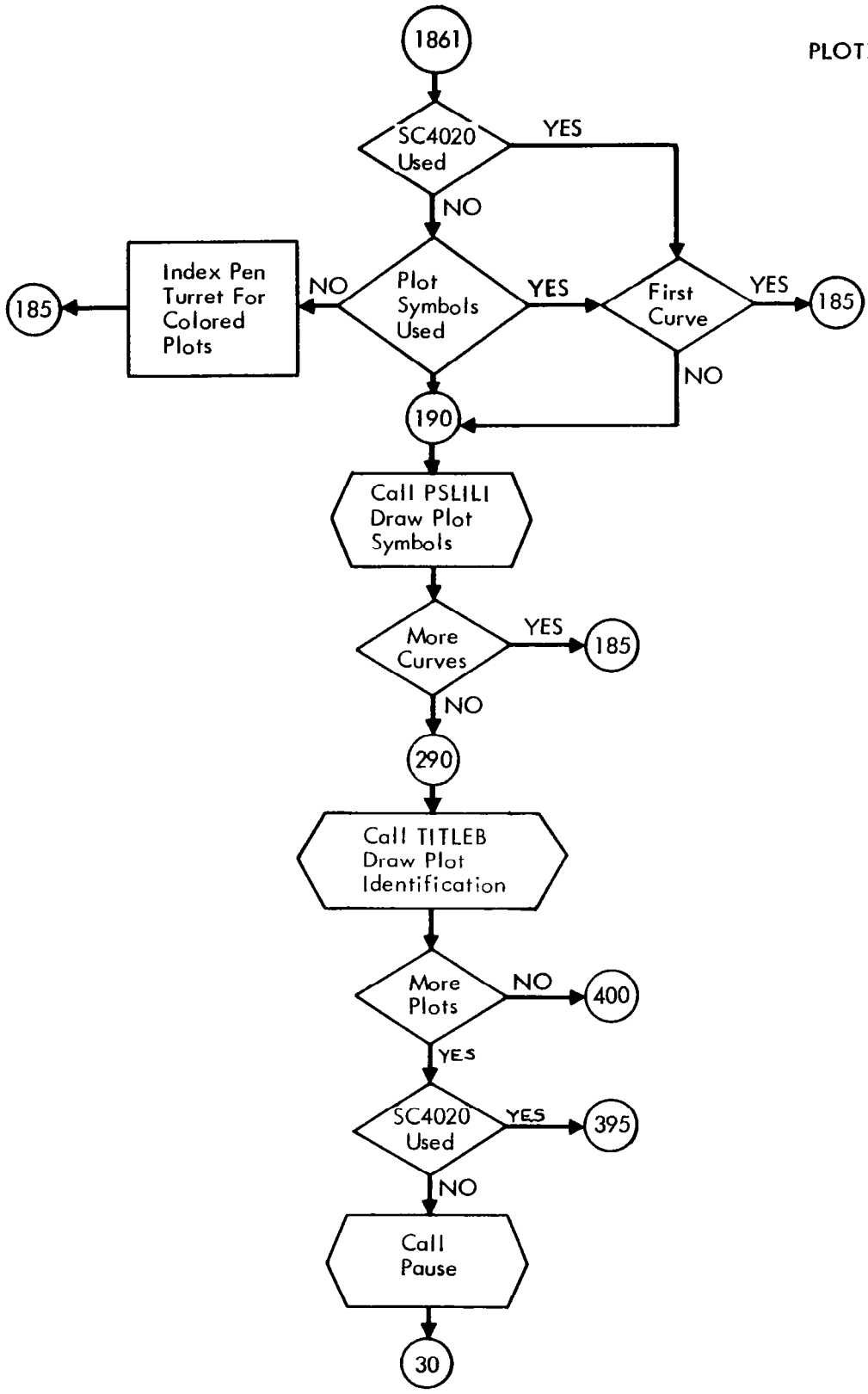

PLOTZ
Print The Name Of Plotter Used.
Initialize Dwg. Tool No. NPS Tape No. Plot Tape No.
SC4020 Being Used
NO
YES
11
Draw Alignment Axes On Orthomat Bed
Draw Grid On SC4020
First Plot
YES
NO
30
Print No. Of Plots No. Of Curves No. Of Points Per Curve
Rewind KPLT
Is There Only 1 Plot
Read Data For 1 Plot Only.
80

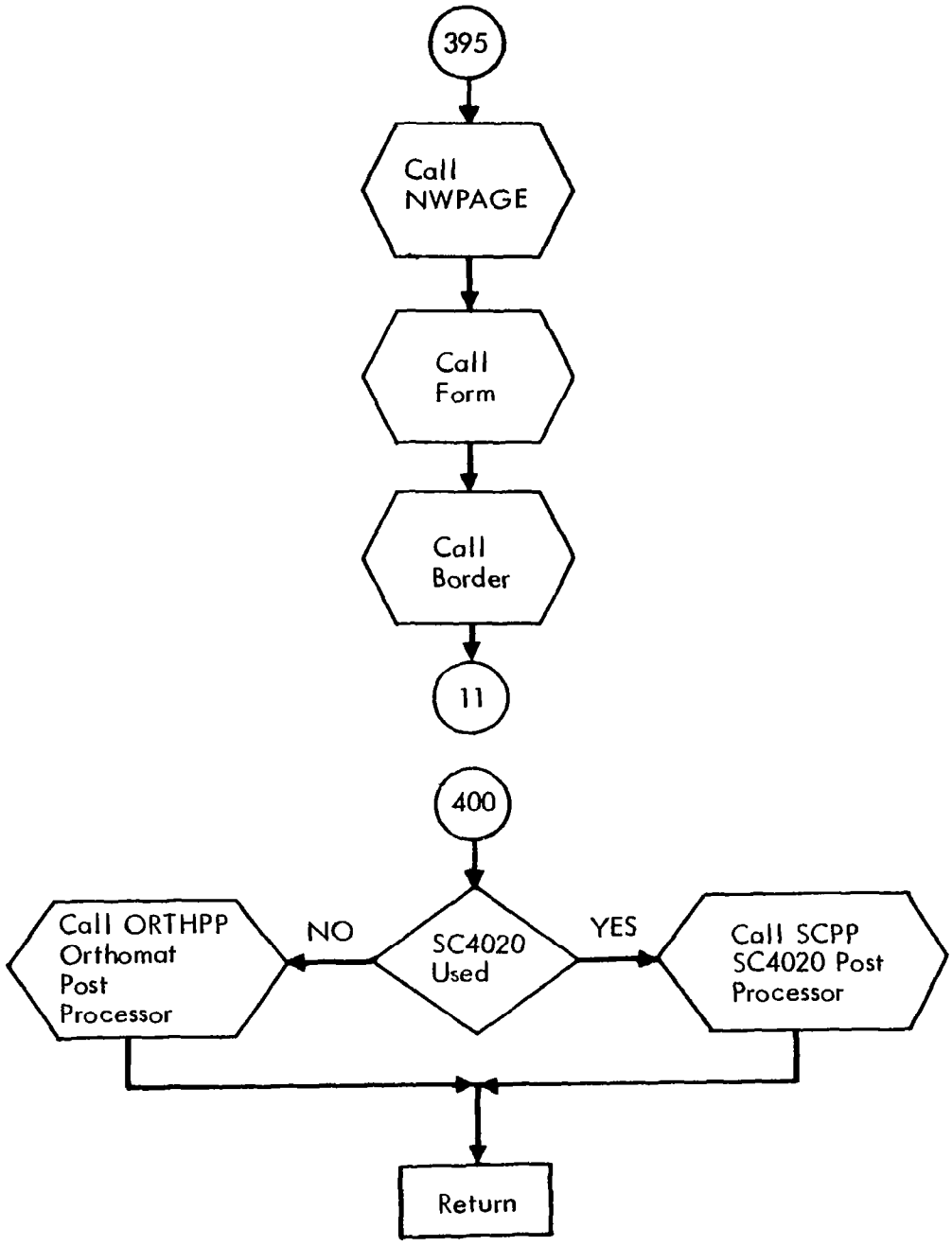

30
Initialize Plot Number, No. Of Points Per Curve
First Plot
YES
NO
Read Data From KPLT For Plot Number 1
Last Plot
YES
NO
Read Data From KPLT For The Last Plot
Read Data From KPLT For Plots Between First And Last.
80
Rewind KPLT
Print The Plot No., No. Curves/ Plot And Plot Variable Names
90


90
Determine Max and Min of Data for this Variable
NO
MAX = MIN
YES
Indep. Variable
NO
YES
Write Error Message
Write Error Message
299
YES
Indep. Variable
NO
Determine Appropriate Scale For Indep. Variable
Determine Appropriate Scale For Dep. Variable
Call SKALZ
Call SKALZ
YES
175
ORTHO Used
JDEP = 2
90
A


A
Initialize Specifications For SC4020 Annotation
175
Call AXLILI Draw ORTHO Axes
Initialize Specs. For Orthomat Annotation
Initialize General Specs. For Annotation
Call NOSLIL Y Axis Annotation
Call NOSLIL X Axis Annotation
Set Up Specs. For Plotting Straight Lines Between Data Pts.
184


184
NITR8 = 0
(Curve No.)
185
NITR8 =
NITR8 + 1
Initialize
No. Of Data Pts
For This Curve.
X and Y Data
Skips.
Call
SLLILI
YES
Ortho
Used
NO
Call
SLLILI
Darken SC4020 Plots
More
Curves To
Plot
YES
1861
NO
SC4020
Used
YES
190
NO
Colored
Ink Used
YES
290
NO
190


1861
SC4020 Used
YES
NO
Index Pen Turret For Colored Plots
185
NO
Plot Symbols Used
YES
First Curve
YES
185
NO
190
Call PSLILI Draw Plot Symbols
More Curves
YES
185
NO
290
Call TITLEB Draw Plot Identification
More Plots
NO
400
YES
SC4020 Used
YES
395
NO
Call Pause
30


395
Call NWPAGE
Call Form
Call Border
11
400
SC4020 Used
NO
Call ORTHPP Orthomat Post Processor
YES
Call SCPP SC4020 Post Processor
Return

PRTIAL —Numerical Partial Derivative Check

Purpose

PRTIAL calculates the partial derivatives of all of the equations of motion for state and placard variables with respect to each of the state variables and control variables. These are used for checking the analytical partial derivatives used in the steepest-ascent method (see appendix B).

Method

The numerical partial derivatives are evaluated using a linear approximation by perturbing the state or control variable about the present value. The perturbations are made both positive and negative from the mean value. For example, if $z = f(y_1, y_2, \ldots, y_n)$, where y_i is either a state or control variable, the numerical partial derivative is given by

$$\frac{\partial z}{\partial y_i} = \frac{f(y_1, y_2, \ldots, y_i + h, \ldots, y_n) - f(y_1, y_2, \ldots, y_i - h, \ldots, y_n)}{2h}$$

where h is the perturbation increment.

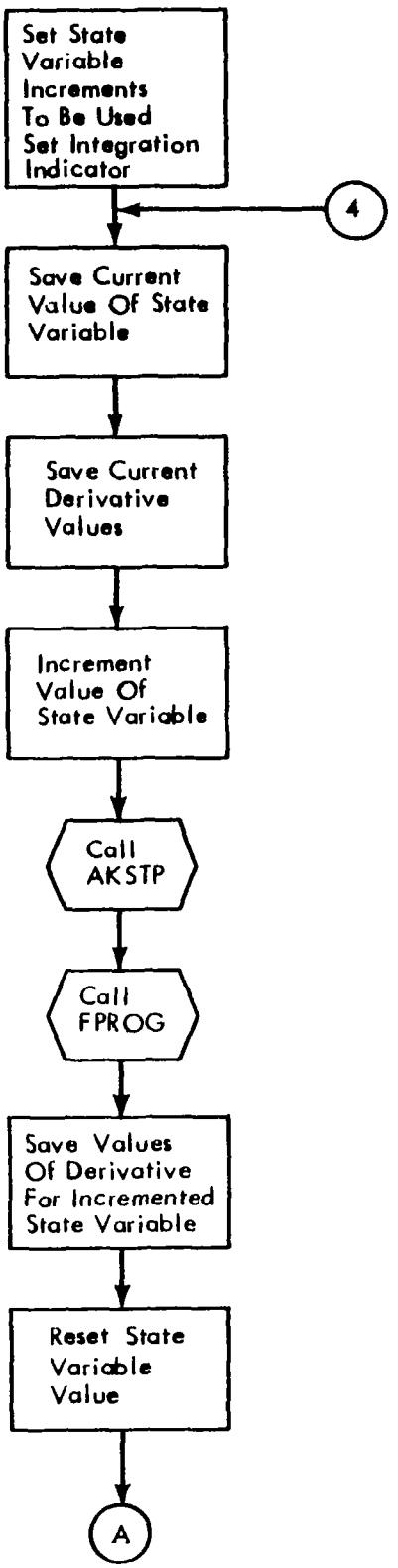
Subroutines Called

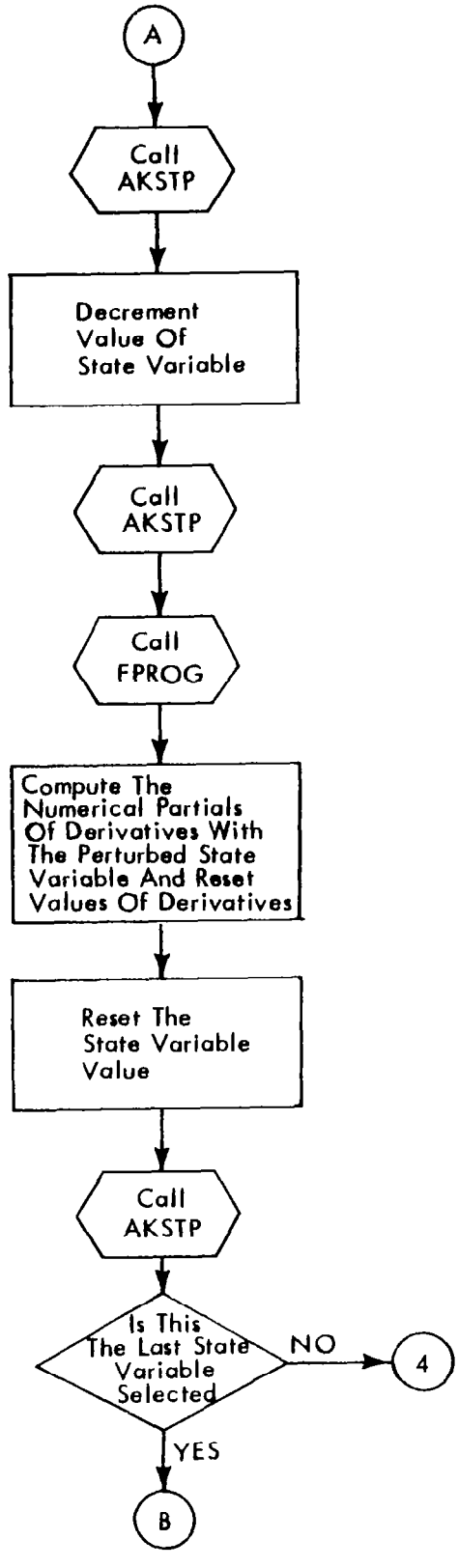
AKSTP FPROG

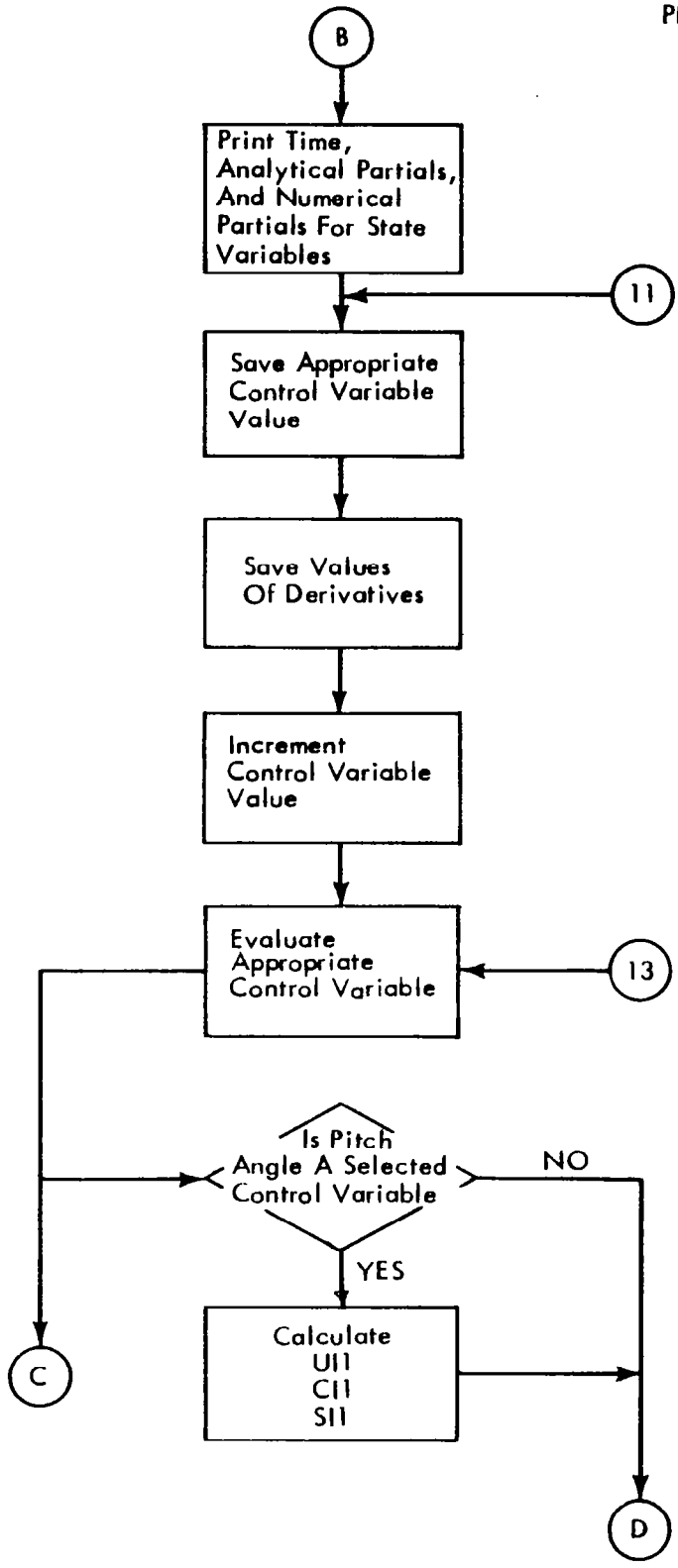
Storage Used

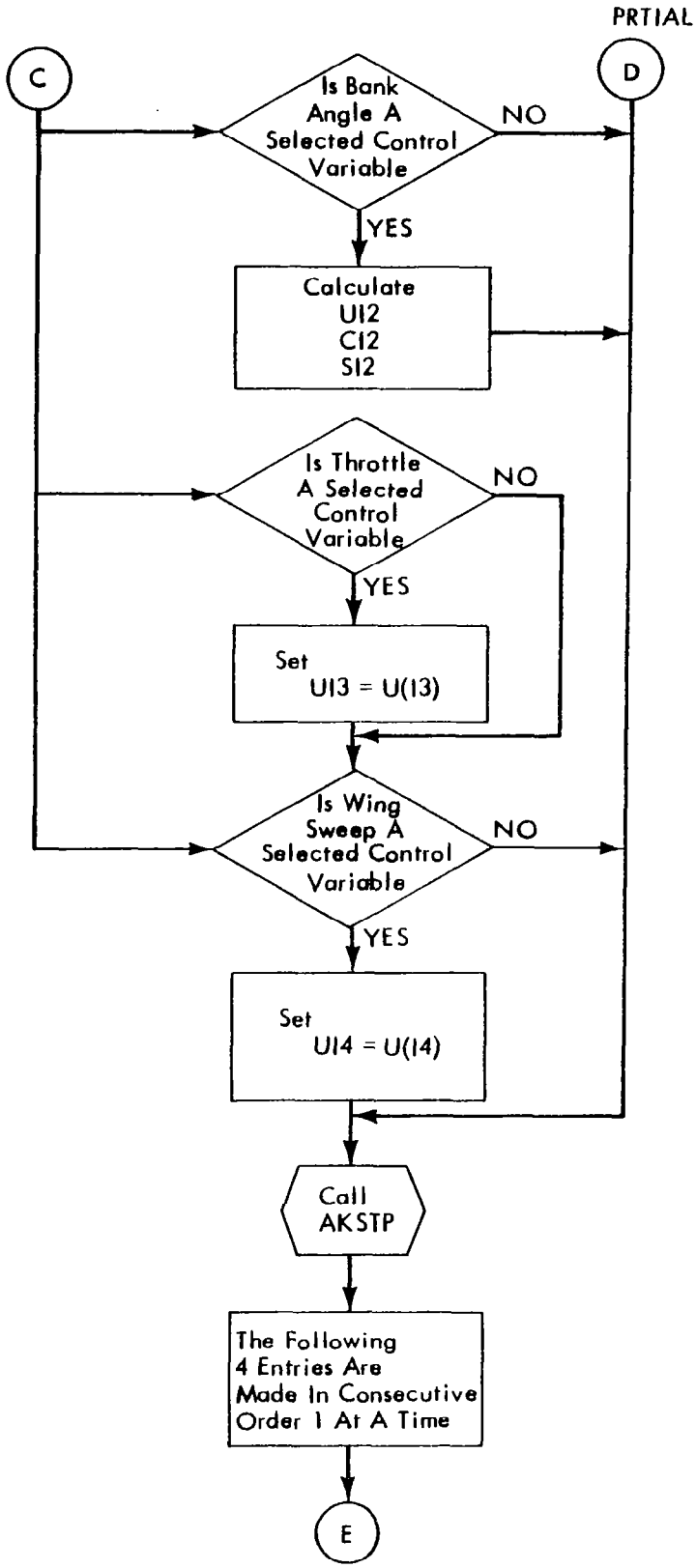
743 cells

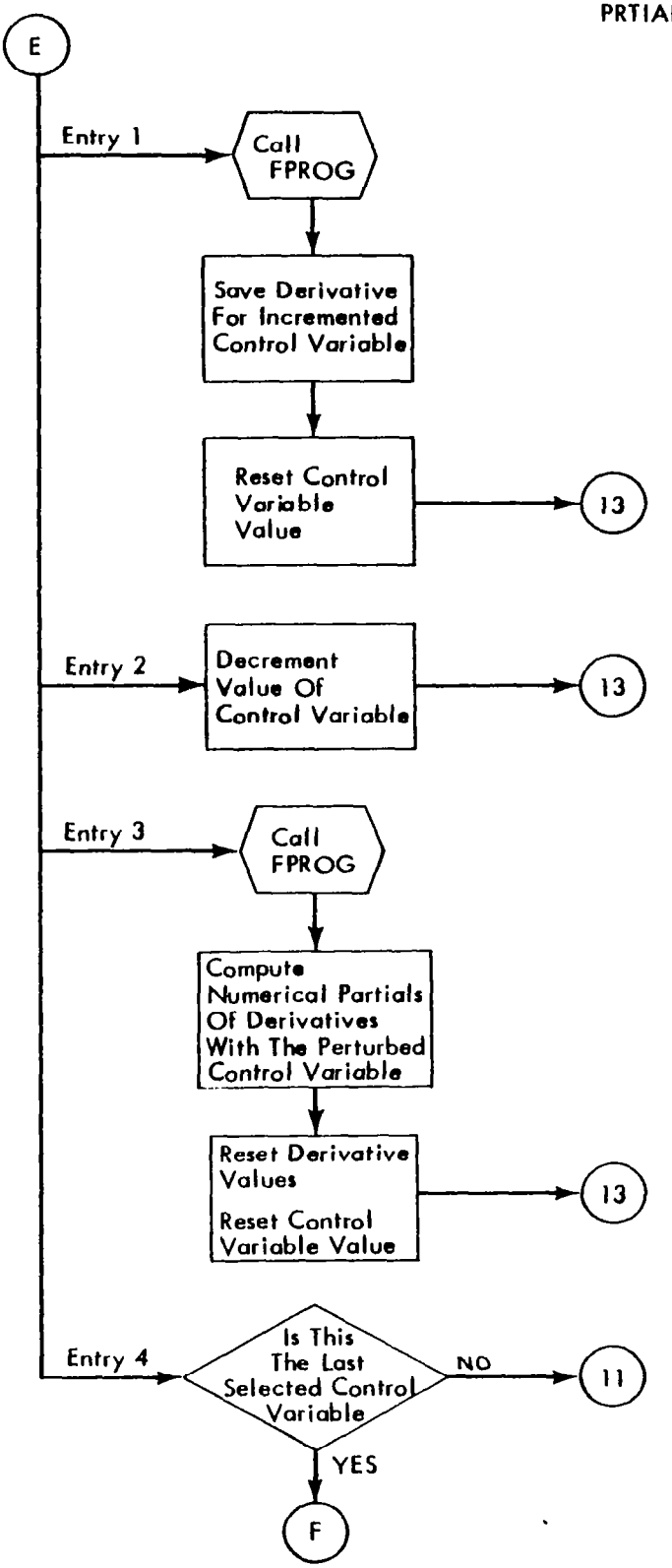
PRTIAL

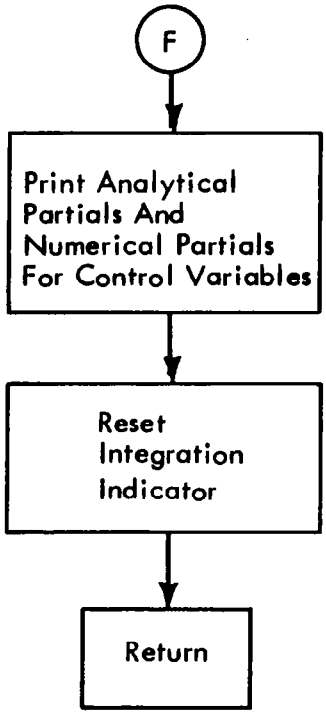

Set State Variable Increments To Be Used Set Integration Indicator
4
Save Current Value Of State Variable
Save Current Derivative Values
Increment Value Of State Variable
Call AKSTP
Call FPROG
Save Values Of Derivative For Incremented State Variable
Reset State Variable Value
A


A
Call AKSTP
Decrement Value Of State Variable
Call AKSTP
Call FPROG
Compute The Numerical Partials Of Derivatives With The Perturbed State Variable And Reset Values Of Derivatives
Reset The State Variable Value
Call AKSTP
Is This The Last State Variable Selected
NO
4
YES
B


B
Print Time, Analytical Partials, And Numerical Partials For State Variables
11
Save Appropriate Control Variable Value
Save Values Of Derivatives
Increment Control Variable Value
Evaluate Appropriate Control Variable
13
Is Pitch Angle A Selected Control Variable
NO
YES
Calculate U11 C11 S11
C
D


C
Is Bank Angle A Selected Control Variable
NO
D
YES
Calculate U12 C12 S12
Is Throttle A Selected Control Variable
NO
YES
Set U13 = U(13)
Is Wing Sweep A Selected Control Variable
NO
YES
Set U14 = U(14)
Call AKSTP
The Following 4 Entries Are Made In Consecutive Order 1 At A Time
E


E
Entry 1
Call FPROG
Save Derivative For Incremented Control Variable
Reset Control Variable Value
13
Entry 2
Decrement Value Of Control Variable
13
Entry 3
Call FPROG
Compute Numerical Partials Of Derivatives With The Perturbed Control Variable
Reset Derivative Values
Reset Control Variable Value
13
Entry 4
Is This The Last Selected Control Variable
NO
11
YES
F


F
Print Analytical Partials And Numerical Partials For Control Variables
Reset Integration Indicator
Return

SKALZ — Plot Scales

Purpose

SKALZ determines the scales for plotting.

Method

Scales are based upon the number of divisions allowed and the variable range covered. The call to SKALZ is made through the calling sequence.

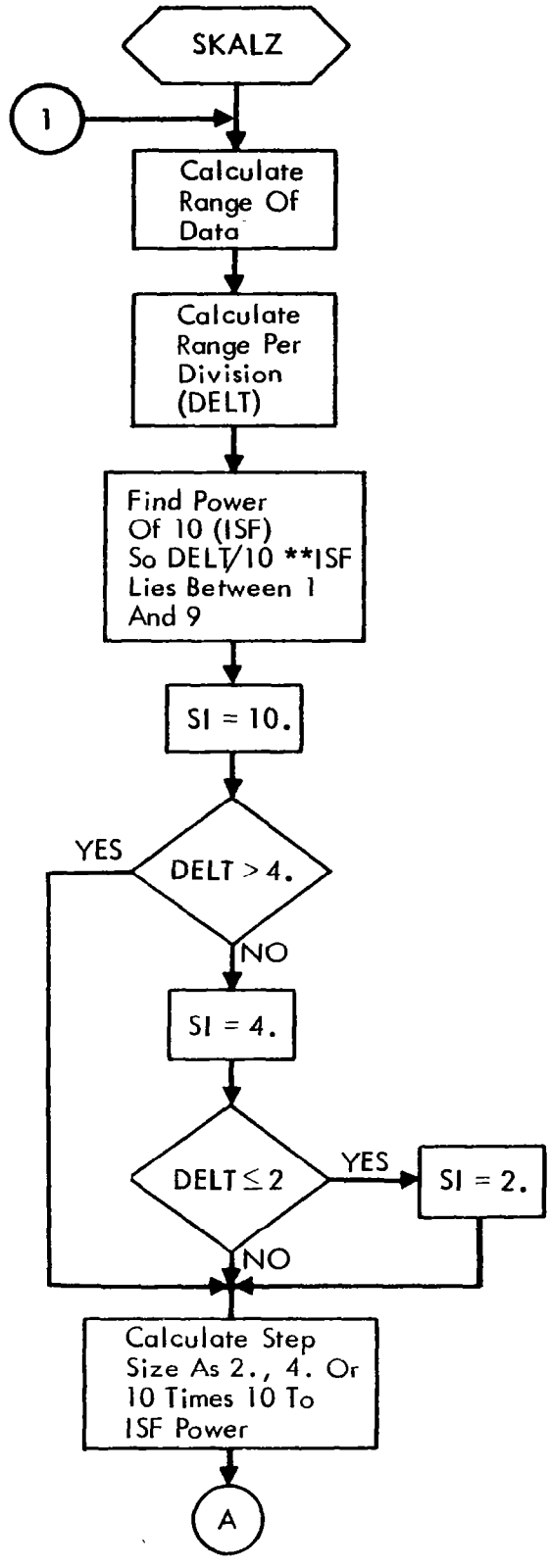
CALL SKALZ (DIV, DMAX, DMIN, SDIV, SMAX, SMIN, STEPP, DIST)

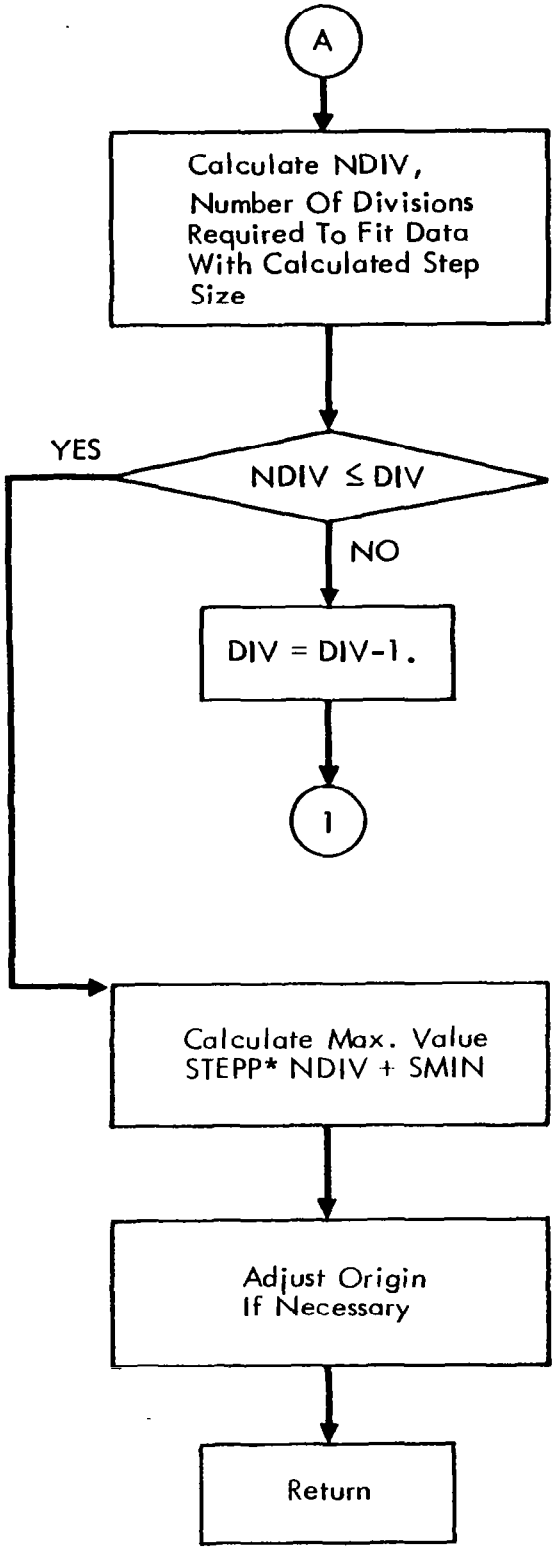
where

DIV	=	maximum number of subdivisions that may be used
DMAX	=	maximum value of independent or dependent variable
DMIN	=	minimum value of independent or dependent variable
SDIV	=	number of divisions used for x or y axis ($<$ DIV)
SMAX	=	maximum value of independent or dependent variable ($\geq$ DMAX)
SMIN	=	minimum value of independent or dependent variable ($\leq$ DMIN)
STEPP	=	step size between successive annotation on an axis (SMAX-SMIN/SDIV)
DIST	=	distance to alter the origin of axis given in one-division increments, (DIV-SDIV)/2.

Storage Used

412 cells


SKALZ
1
Calculate Range Of Data
Calculate Range Per Division (DELT)
Find Power Of 10 (ISF) So DELT/10 **ISF Lies Between 1 And 9
SI = 10.
YES
DELT > 4.
NO
SI = 4.
DELT ≤ 2
YES
SI = 2.
NO
Calculate Step Size As 2., 4. Or 10 Times 10 To ISF Power
A


A
Calculate NDIV,
Number Of Divisions
Required To Fit Data
With Calculated Step
Size
YES
NDIV ≤ DIV
NO
DIV = DIV-1.
1
Calculate Max. Value
STEPP* NDIV + SMIN
Adjust Origin
If Necessary
Return

STEP1 — Integration Of Forward Trajectory

Purpose

STEP1 integrates the equations of motion defining the state variable.

Method

The integration package is a variable-step Runge-Kutta. The Runge-Kutta technique is basically a fourth-order method that is mechanized to vary the integration interval based on two half-step intervals as compared with the full step. This system evaluates the NV derivatives 11 times per integration interval. Only the basic NV equations of motion are integrated using the 11-point method. The other equations of motion are integrated using only four evaluations of the derivatives to save computer time. A complete writeup of the RKVS method is given in reference 8.

Assumptions and Limitations

The minimum step size is specified by the user.

Subroutines Called

AKSTP CONTR FPROG STP1

Storage Used

1179 cells

STEP2 — Adjoint Equation Integration

Purpose

STEP2 integrates the adjoint equations of motion backwards along the trajectory.

Method

Integration is performed using a modification of the Runge-Kutta variable-step technique given in reference 8.

The call to STEP2 is made through the calling statement

```
CALL  STEP2 ( UULAM)
```

where UULAM is a matrix of the weighted impulse response.

Subroutines Called

DVAL2 STP2

Storage Used

2323 cells

STP1 — Store and Print Forward Trajectory Data

Purpose

STP1:

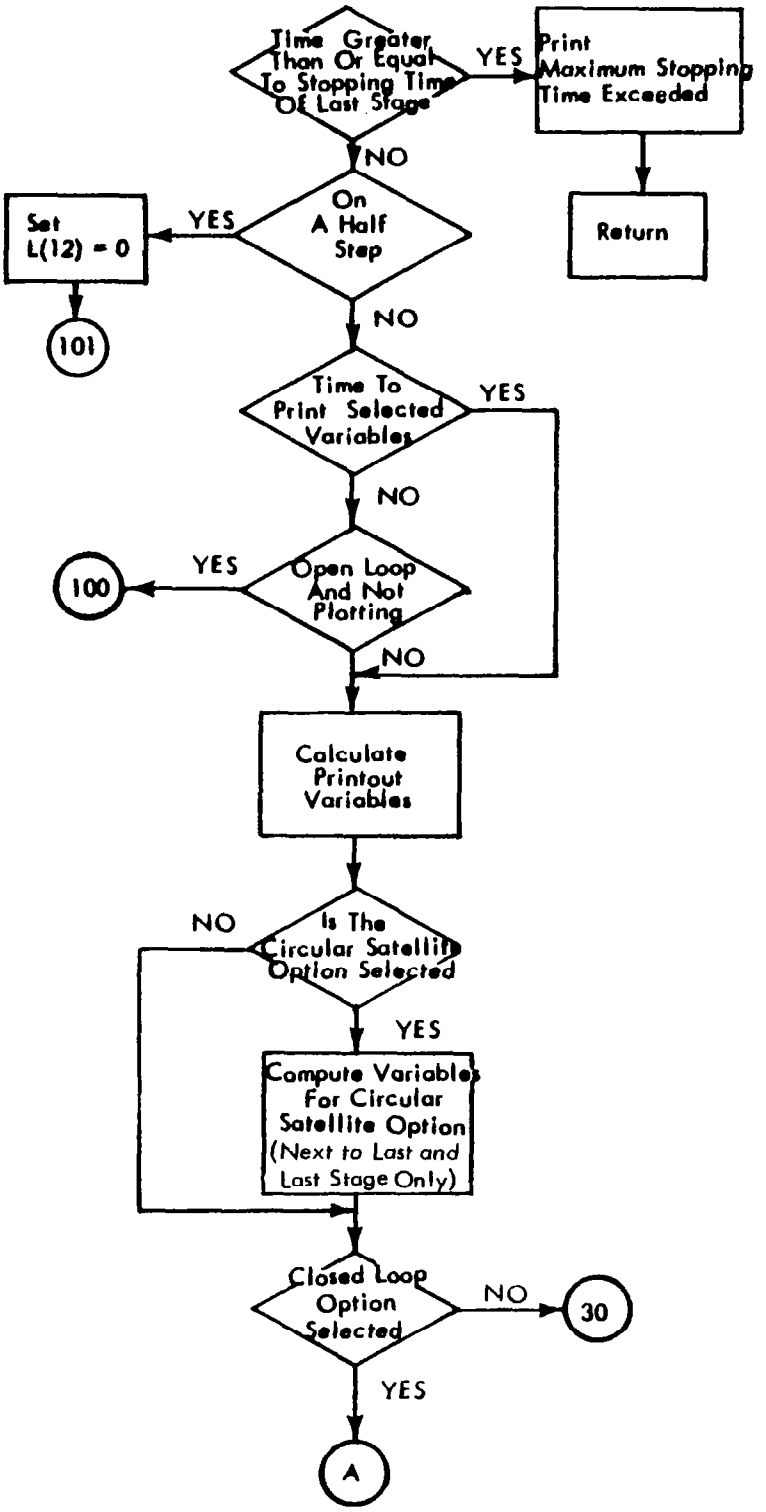
1) Calculates many of the AK variables, including metric conversions and transforms relative state variables to inertial coordinate system;
2) Performs the circular satellite option calculations;
3) Stores partials and plotting data on tape;
4) Tests for maximum stopping time;
5) Controls phases for nominal-guidance modes;
6) Controls printout of forward trajectory.

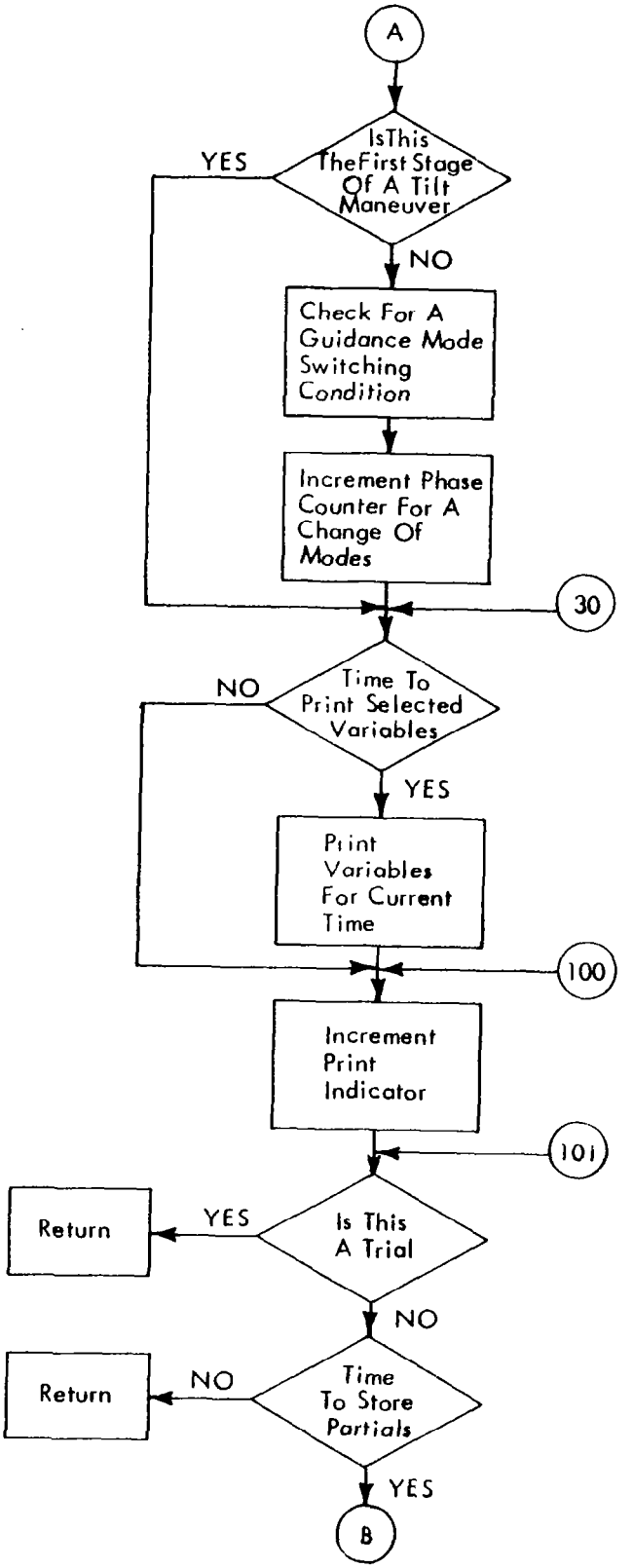
Subroutines Called

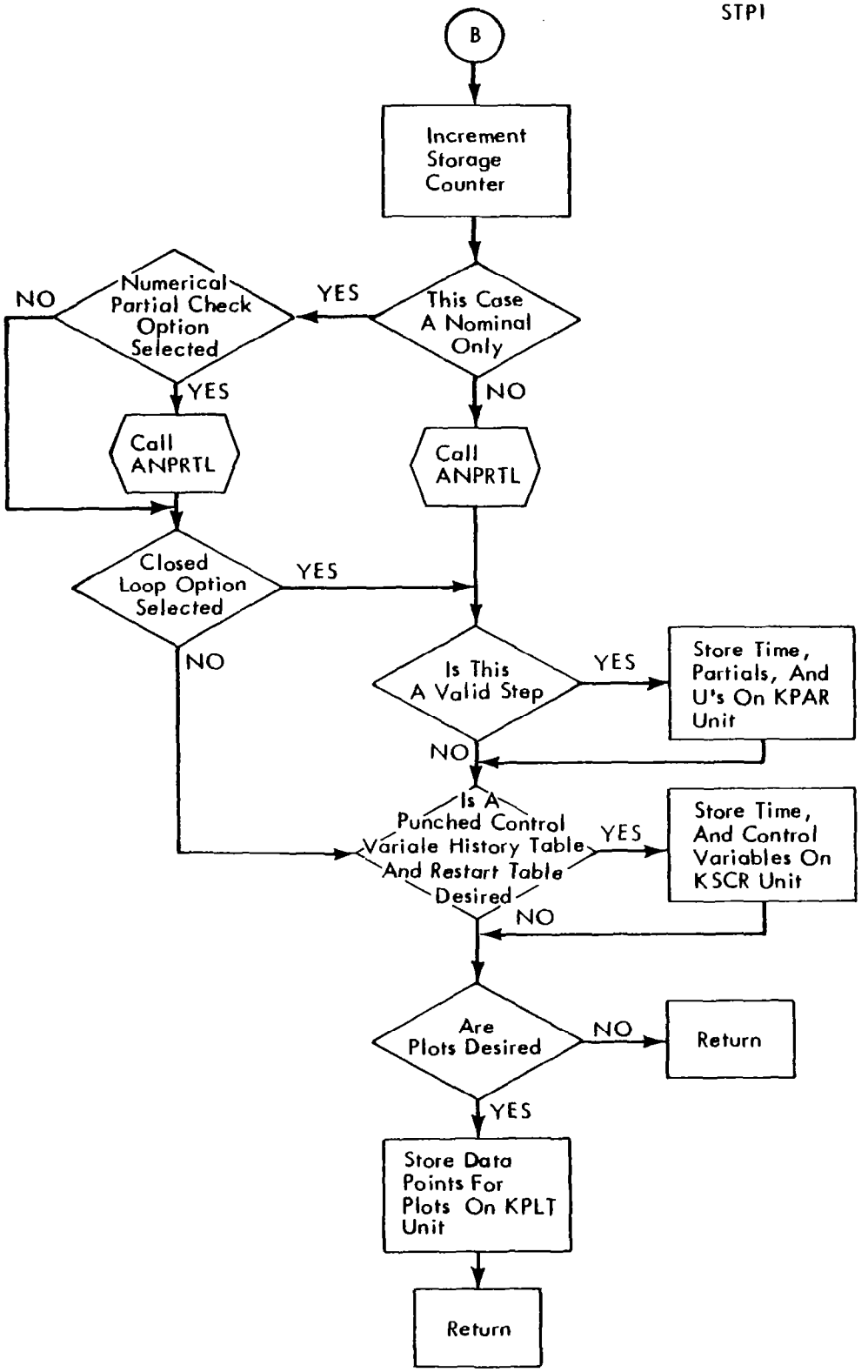
ANPRTL

Storage Used

867 cells


Time Greater Than Or Equal To Stopping Time Of Last Stage
YES
Print Maximum Stopping Time Exceeded
Return
NO
On A Half Step
YES
Set L(12) = 0
101
NO
Time To Print Selected Variables
YES
NO
Open Loop And Not Plotting
YES
100
NO
Calculate Printout Variables
Is The Circular Satellite Option Selected
NO
YES
Compute Variables For Circular Satellite Option (Next to Last and Last Stage Only)
Closed Loop Option Selected
NO
30
YES
A


A
Is This The First Stage Of A Tilt Maneuver
YES
NO
Check For A Guidance Mode Switching Condition
Increment Phase Counter For A Change Of Modes
30
Time To Print Selected Variables
NO
YES
Print Variables For Current Time
100
Increment Print Indicator
101
Is This A Trial
YES
Return
NO
Time To Store Partials
NO
Return
YES
B


B
Increment Storage Counter
This Case A Nominal Only
YES
Numerical Partial Check Option Selected
NO
YES
Call ANPRTL
NO
Call ANPRTL
Closed Loop Option Selected
YES
NO
Is This A Valid Step
YES
Store Time, Partials, And U's On KPAR Unit
NO
Is A Punched Control Variale History Table And Restart Table Desired
YES
Store Time, And Control Variables On KSCR Unit
NO
Are Plots Desired
NO
Return
YES
Store Data Points For Plots On KPLT Unit
Return

STP2 — Reverse Trajectory Store and Print

Purpose

STP2 controls the storage and printing of the influence coefficients during the reverse integration.

Method

The option to print the adjoint variables is made by the user (NC(7) $\neq$ 0). The print interval is selected and printing is done only at the completion of an integration interval. Storage of the influence coefficients for use in determining the new control variable history is performed. The partial derivatives stored during the forward trajectory are called from storage and placed in core for use by DVAL2.

The UULAM matrix which is computed in DVAL2 is transmitted to STP2 through the calling sequence

CALL STP2 (UULAM)

where

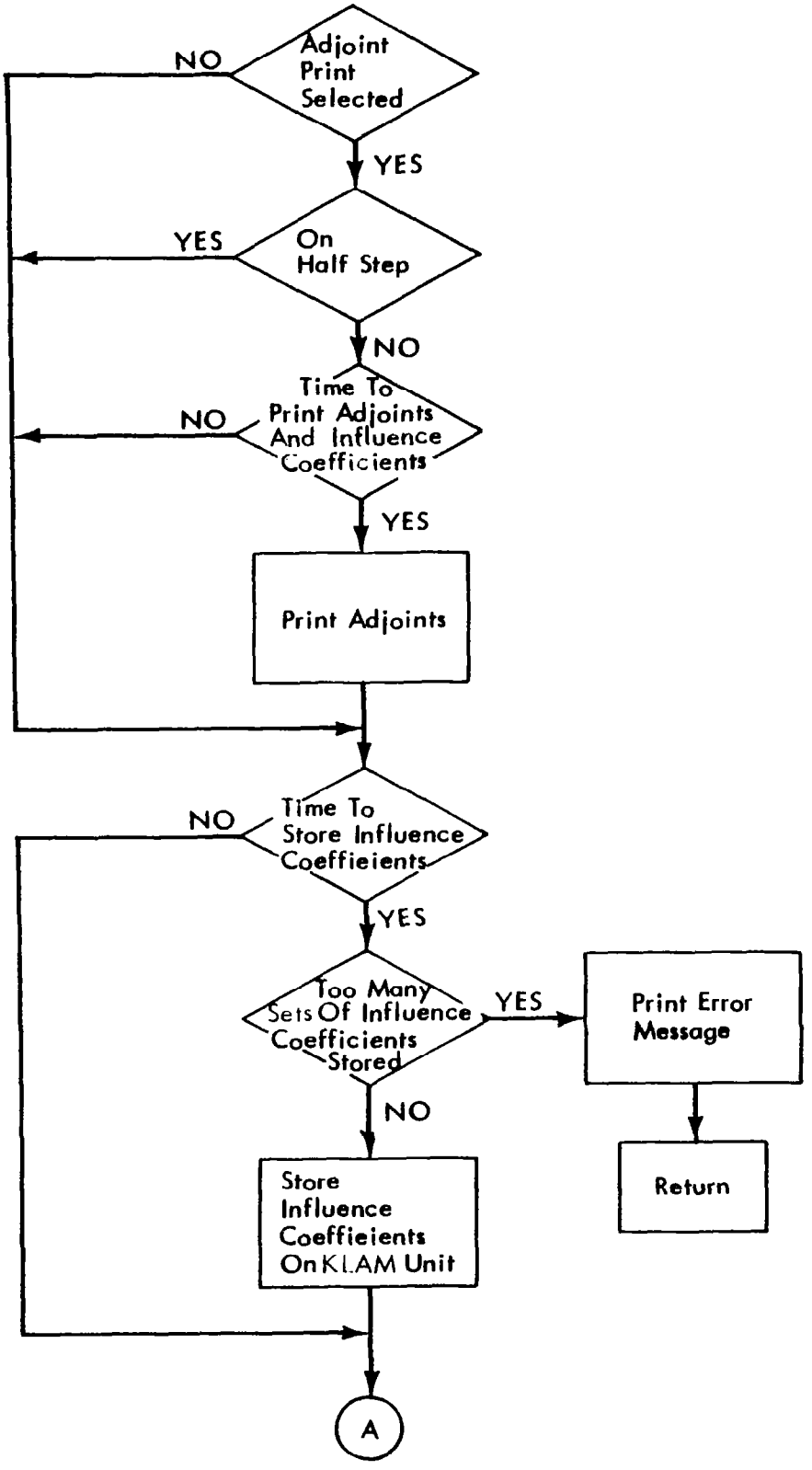
$$UULAM = W^{-1}G' \Lambda$$

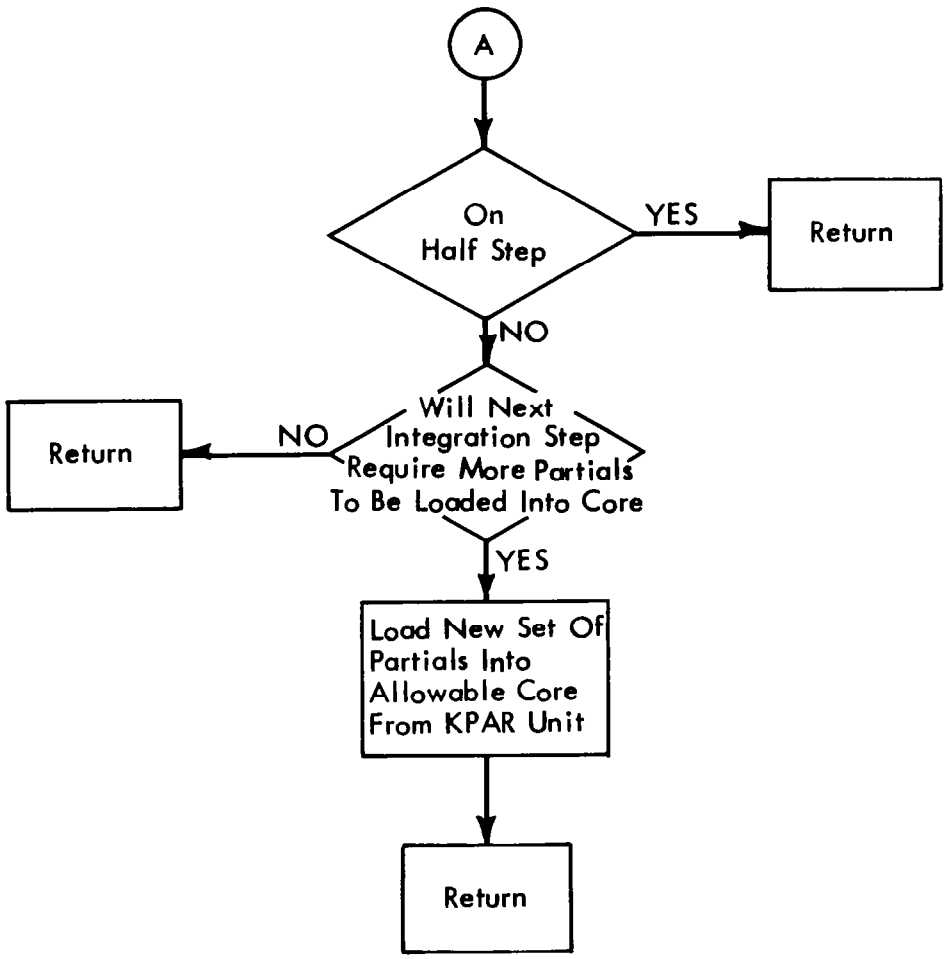
Assumptions and Limitations

The maximum number of points that may be stored is 13590/NUP1, where NUP1 is the number of control variables plus 1.

Storage Used

478 cells


Adjoint Print Selected
NO
YES
On Half Step
YES
NO
Time To Print Adjoints And Influence Coefficients
NO
YES
Print Adjoints
Time To Store Influence Coeffieients
NO
YES
Too Many Sets Of Influence Coefficients Stored
YES
Print Error Message
Return
NO
Store Influence Coeffieients On KLAM Unit
A


A
On Half Step
YES
Return
NO
Will Next Integration Step Require More Partials To Be Loaded Into Core
NO
Return
YES
Load New Set Of Partials Into Allowable Core From KPAR Unit
Return

TITLES — Output Heading Subroutine

Purpose

TITLES is called from INITIAL once per data case and performs the following tasks:

1) Constructs the packed array of headings (DESXUP) from the DESXUK array;

2) Converts the input NC(121) through NC(160) values to internal state variable indicies;

3) Constructs the plot headings for the PLTNAM array;

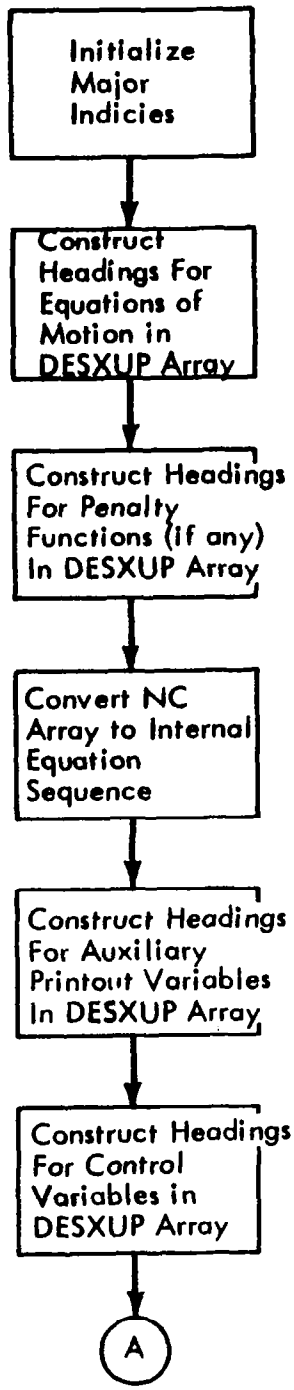
4) Calculates the number of equations of motion (NEOM), placards (NPF), state variables (NX), control variables (NU), and auxiliary printout variables (NK) selected by the user.

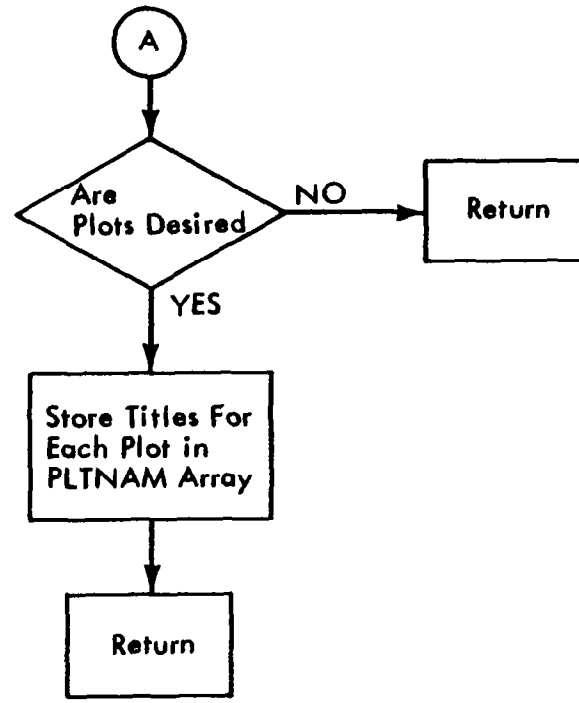
Method

All information required by TITLES is available in the input NC array and the DESXUK array compiled in subroutine BLOCK.

Storage Used

363 cells


Initialize Major Indicies
Construct Headings For Equations of Motion in DESXUP Array
Construct Headings For Penalty Functions (if any) In DESXUP Array
Convert NC Array to Internal Equation Sequence
Construct Headings For Auxiliary Printout Variables In DESXUP Array
Construct Headings For Control Variables in DESXUP Array
A


A
Are Plots Desired
NO
Return
YES
Store Titles For Each Plot in PLTNAM Array
Return

TLD — Thrust, Lift, and Drag Calculations

Purpose

TLD calculates thrust, fuel flow, and aerodynamic coefficients as required by the options. It also calculates some VAR's and partial derivatives as required to perform the above calculations.

Method

TLD uses the NC's input to determine the options to be used for calculating thrust, fuel flow, and aerodynamic coefficients. Body axes data is converted to wind axes for use in the program. Thrust, fuel flow, and aerodynamic coefficients data are obtained by table lookups.

Assumptions and Limitations

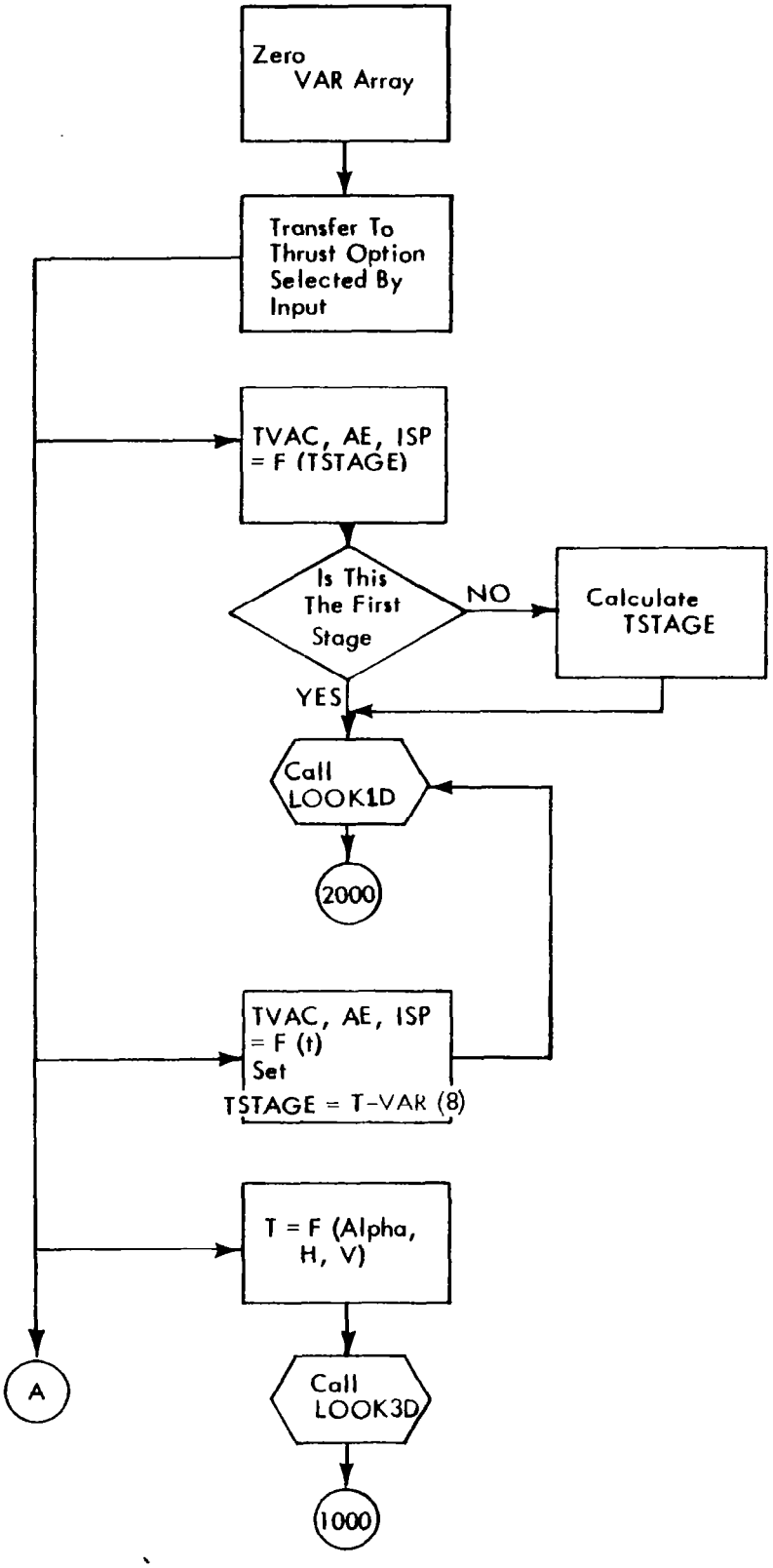
For thrust options 3 to 9, thrust and fuel flow data are tabulated for one engine only. If more than one engine is used, the thrust will be multiplied by the number of engines as input.

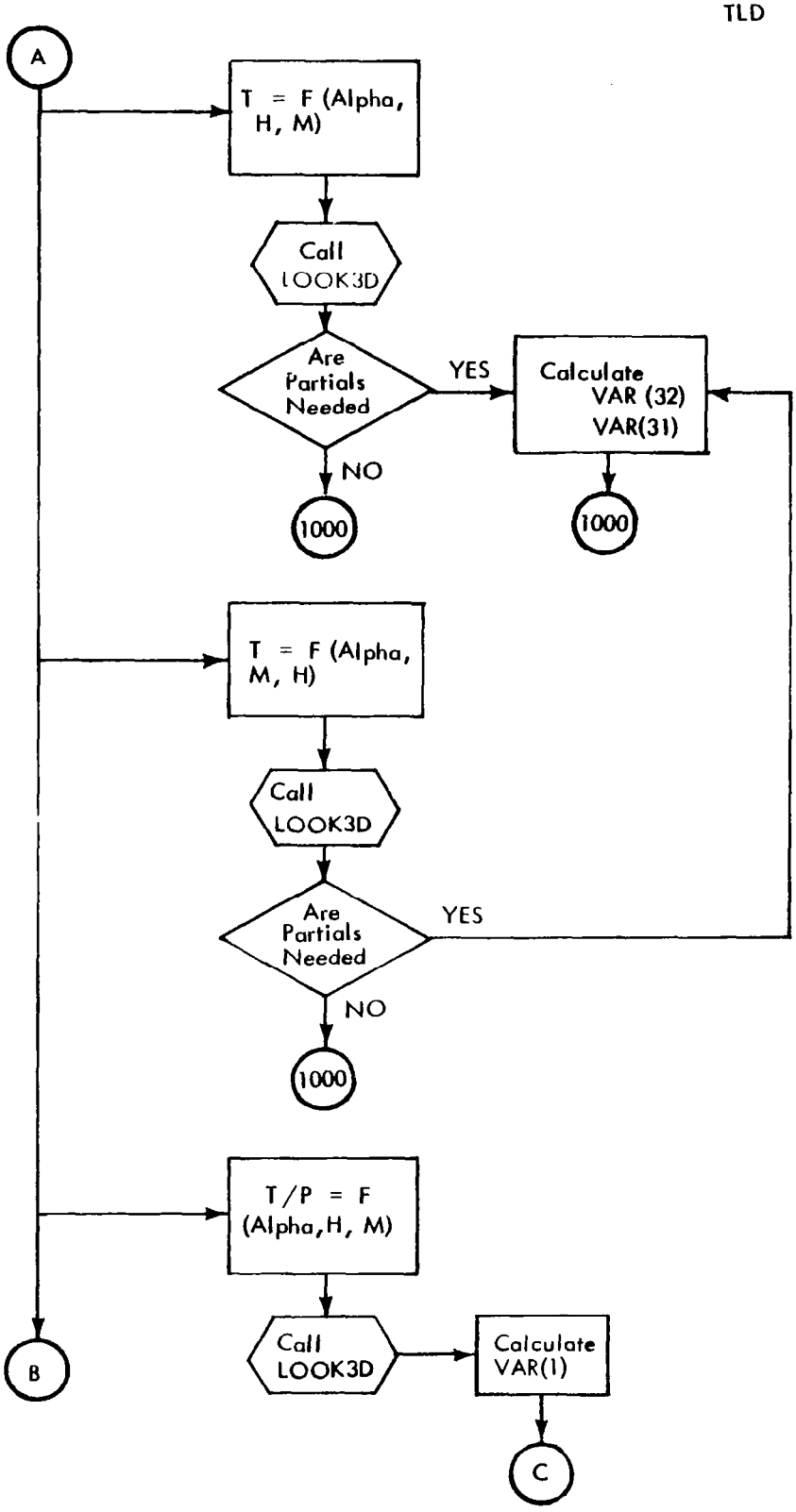
Subroutines Called

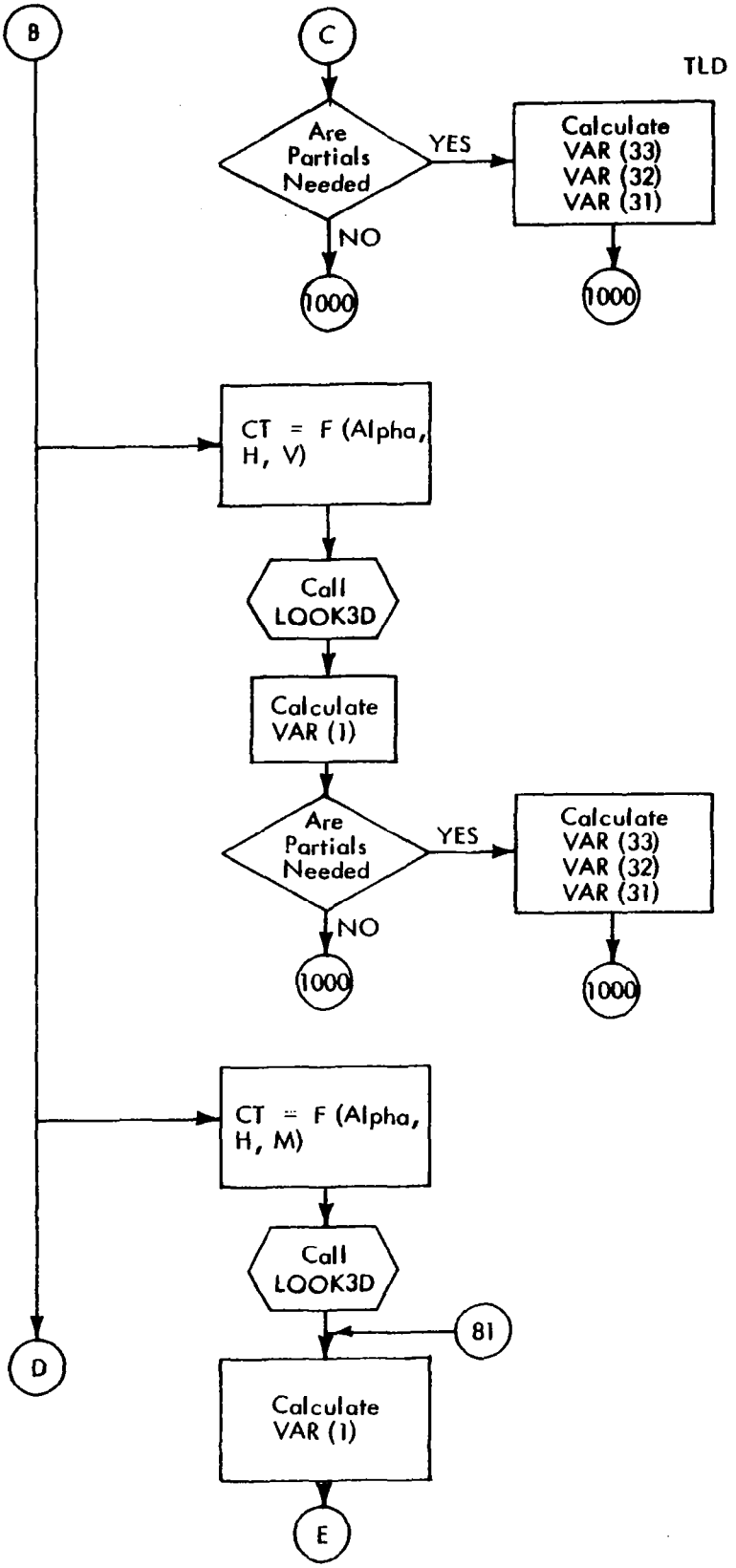
LOOK1D LOOK3D

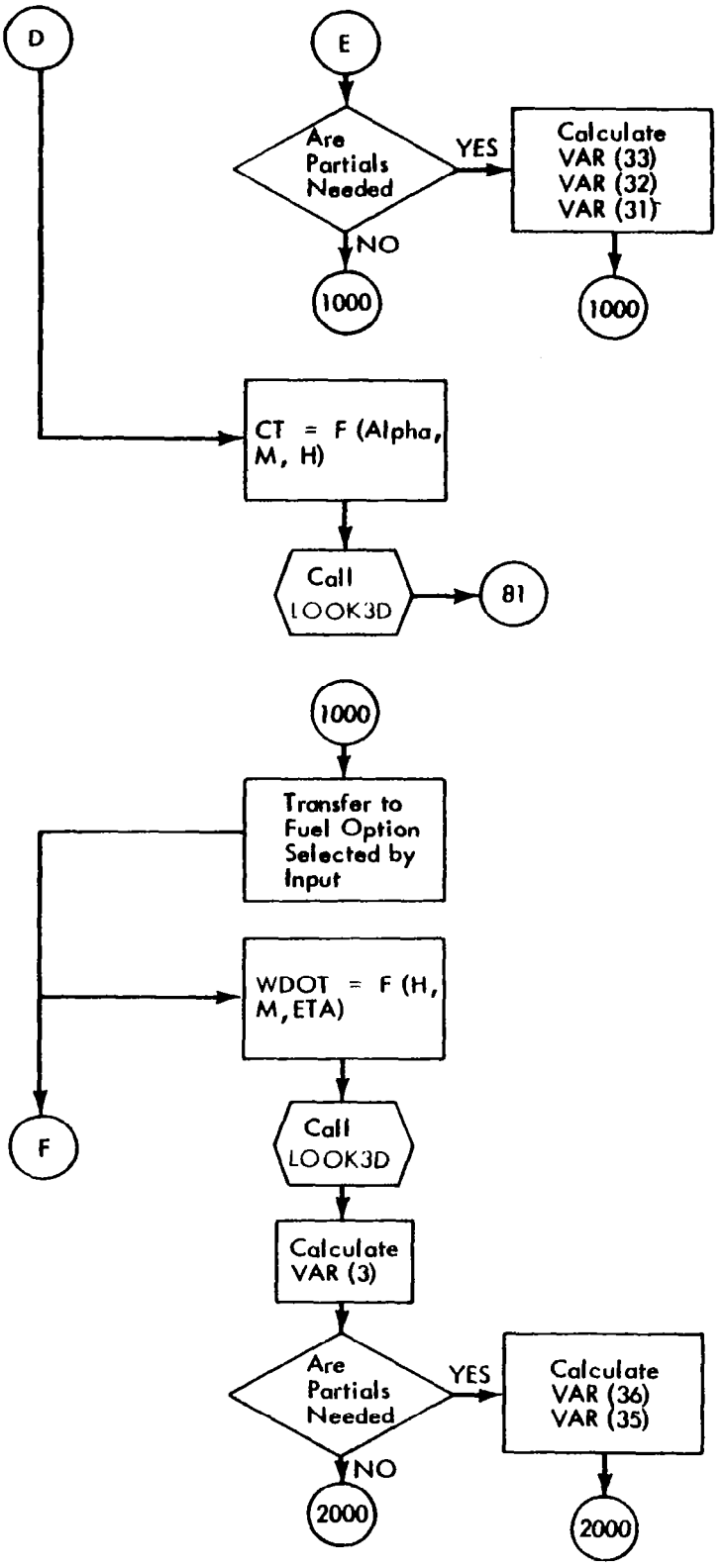
Storage Used

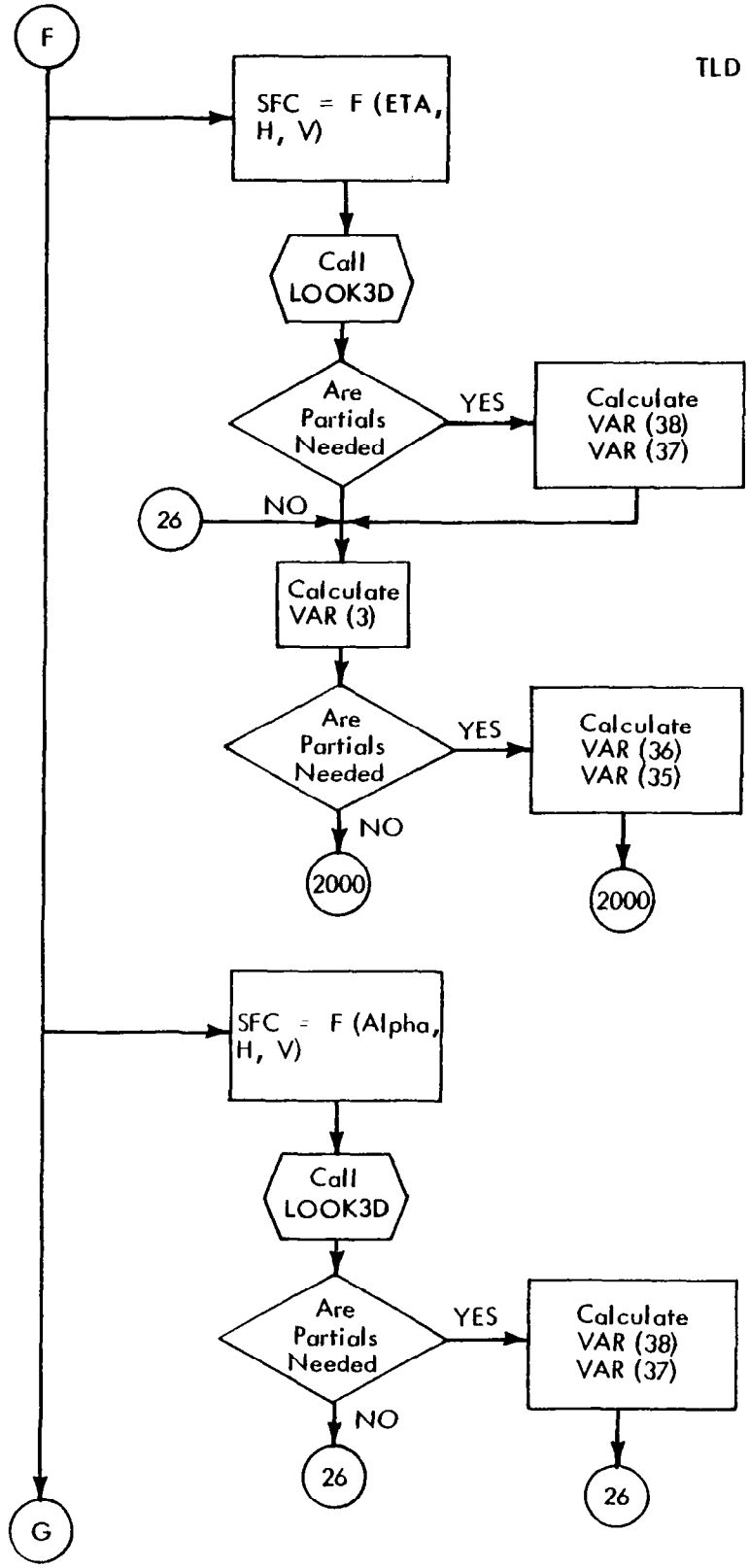
1,419 cells

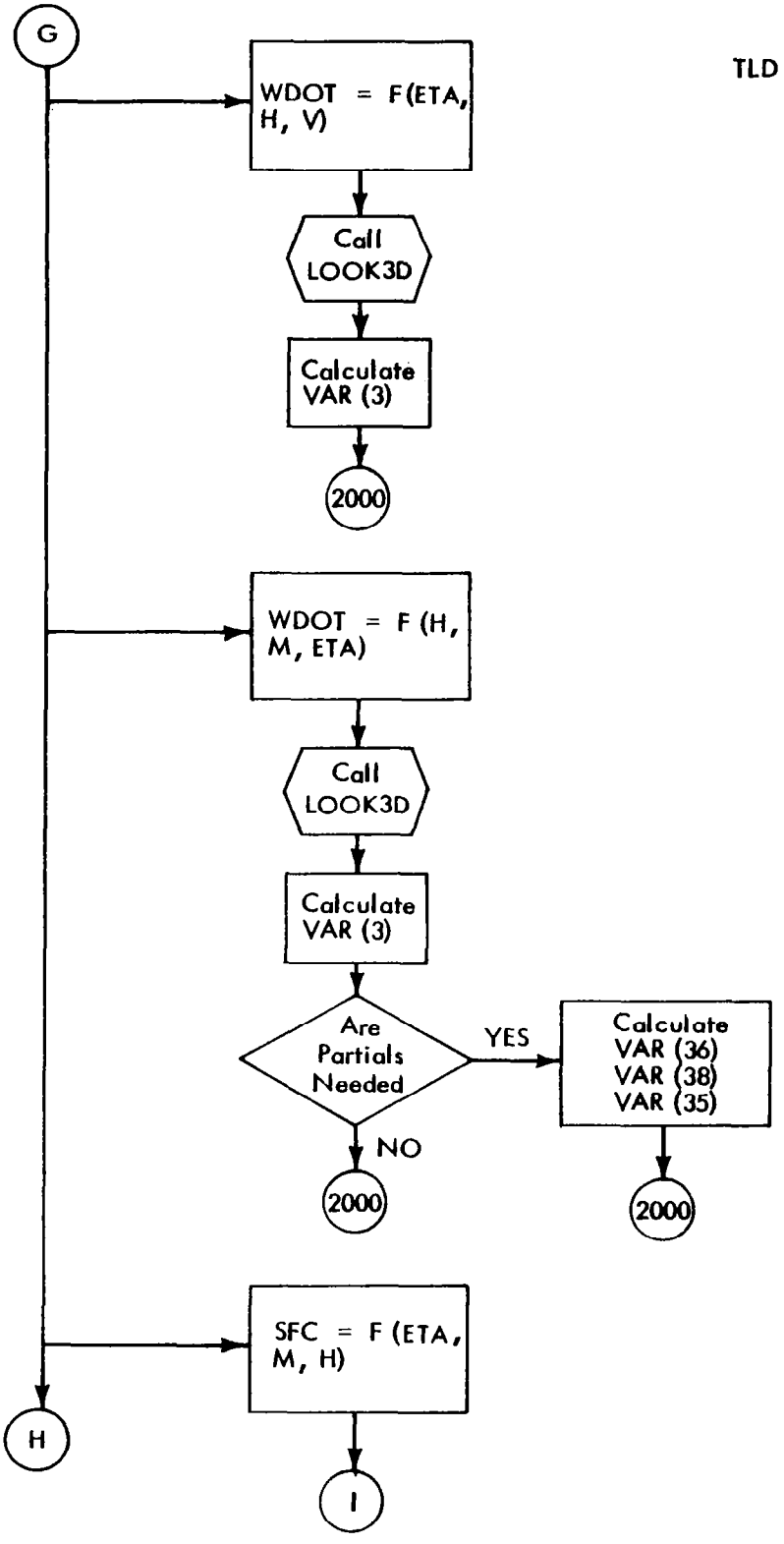

Zero
VAR Array
Transfer To
Thrust Option
Selected By
Input
TVAC, AE, ISP
= F (TSTAGE)
Is This
The First
Stage
NO
Calculate
TSTAGE
YES
Call
LOOK1D
2000
TVAC, AE, ISP
= F (t)
Set
TSTAGE = T-VAR (8)
T = F (Alpha,
H, V)
A
Call
LOOK3D
1000

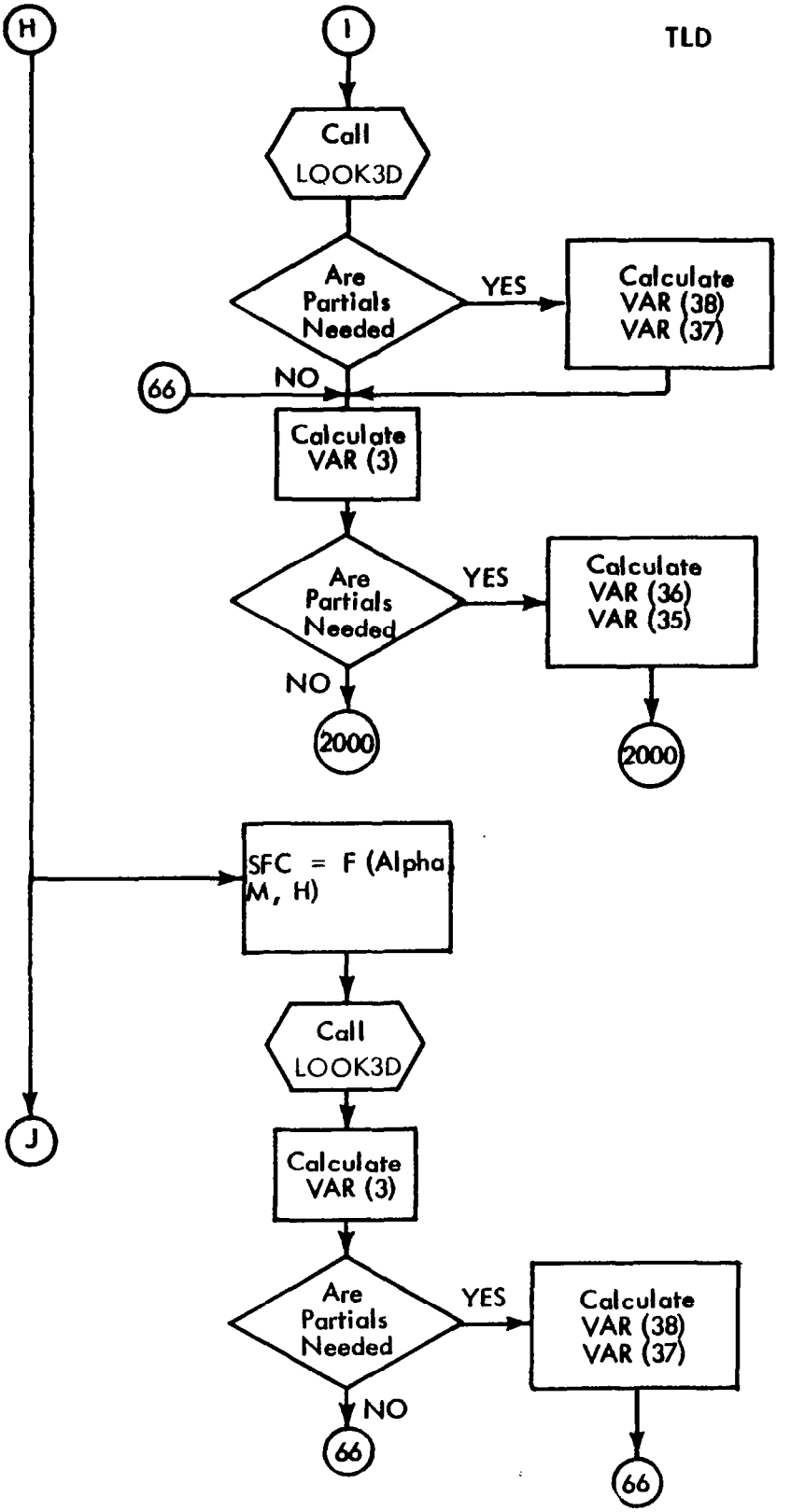

A
T = F (Alpha, H, M)
Call LOOK3D
Are Partials Needed
YES
Calculate VAR (32) VAR(31)
NO
1000
1000
T = F (Alpha, M, H)
Call LOOK3D
Are Partials Needed
YES
NO
1000
T/P = F (Alpha,H, M)
Call LOOK3D
Calculate VAR(1)
B
C

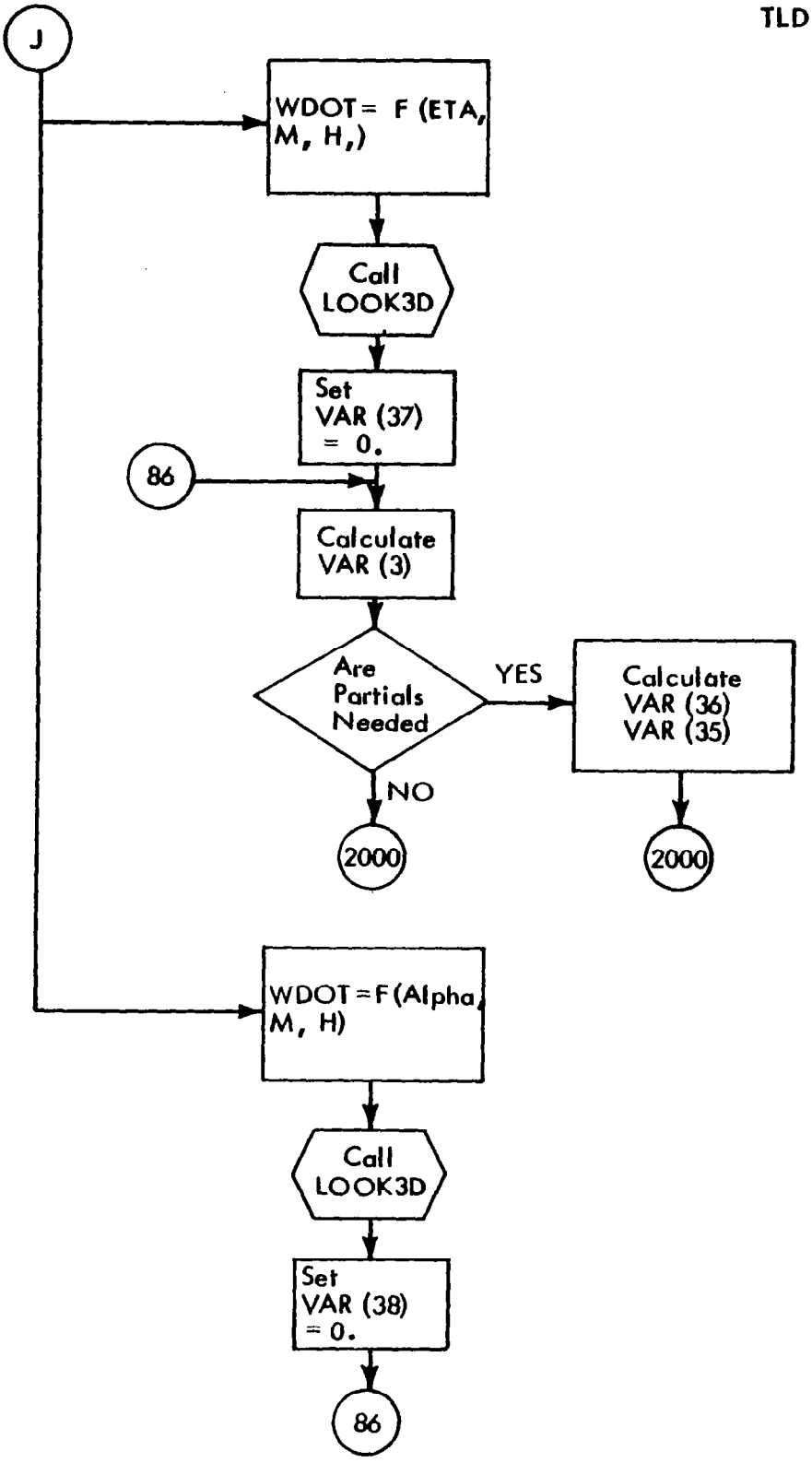

B
C
TLD
Are Partials Needed
YES
NO
Calculate VAR (33) VAR (32) VAR (31)
1000
1000
CT = F (Alpha, H, V)
Call LOOK3D
Calculate VAR (1)
Are Partials Needed
YES
NO
Calculate VAR (33) VAR (32) VAR (31)
1000
1000
CT = F (Alpha, H, M)
Call LOOK3D
81
D
Calculate VAR (1)
E

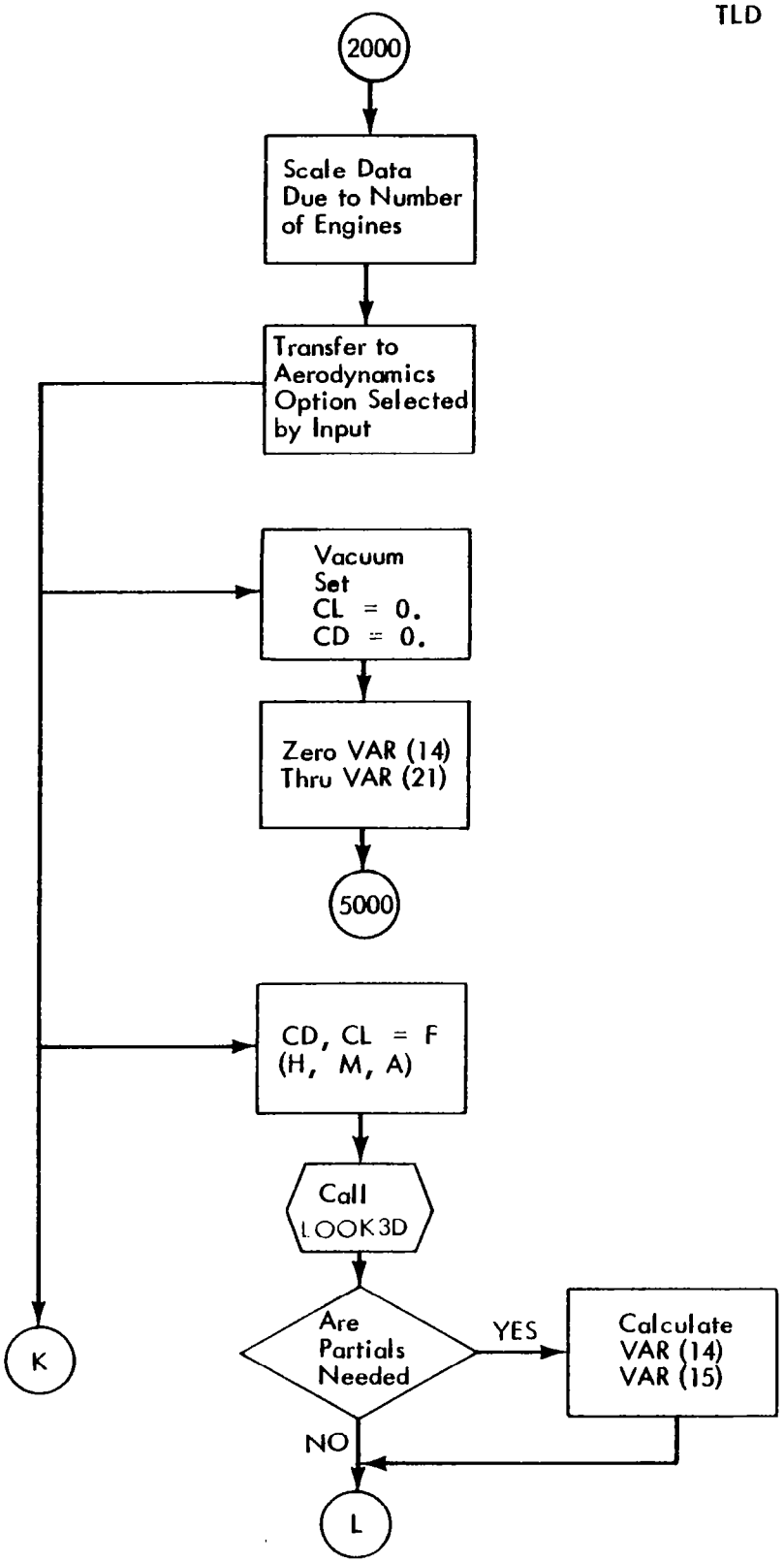

D
E
Are Partials Needed
YES
NO
Calculate VAR (33) VAR (32) VAR (31)
1000
1000
CT = F (Alpha, M, H)
Call LOOK3D
81
1000
Transfer to Fuel Option Selected by Input
WDOT = F (H, M, ETA)
F
Call LOOK3D
Calculate VAR (3)
Are Partials Needed
YES
NO
Calculate VAR (36) VAR (35)
2000
2000

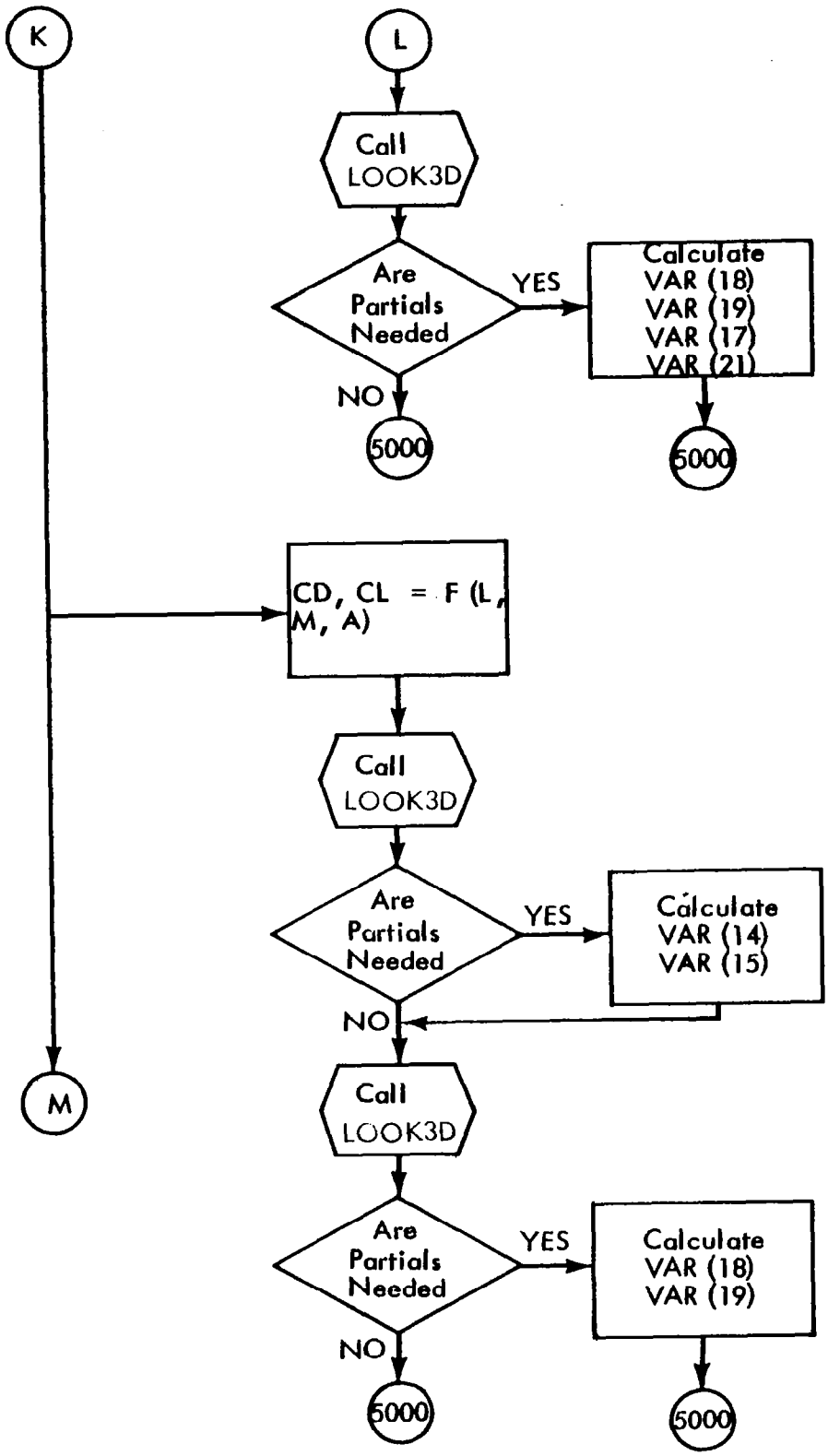

F
TLD
SFC = F (ETA, H, V)
Call LOOK3D
Are Partials Needed
YES
Calculate VAR (38) VAR (37)
26
NO
Calculate VAR (3)
Are Partials Needed
YES
Calculate VAR (36) VAR (35)
NO
2000
2000
SFC = F (Alpha, H, V)
Call LOOK3D
Are Partials Needed
YES
Calculate VAR (38) VAR (37)
NO
26
26
G

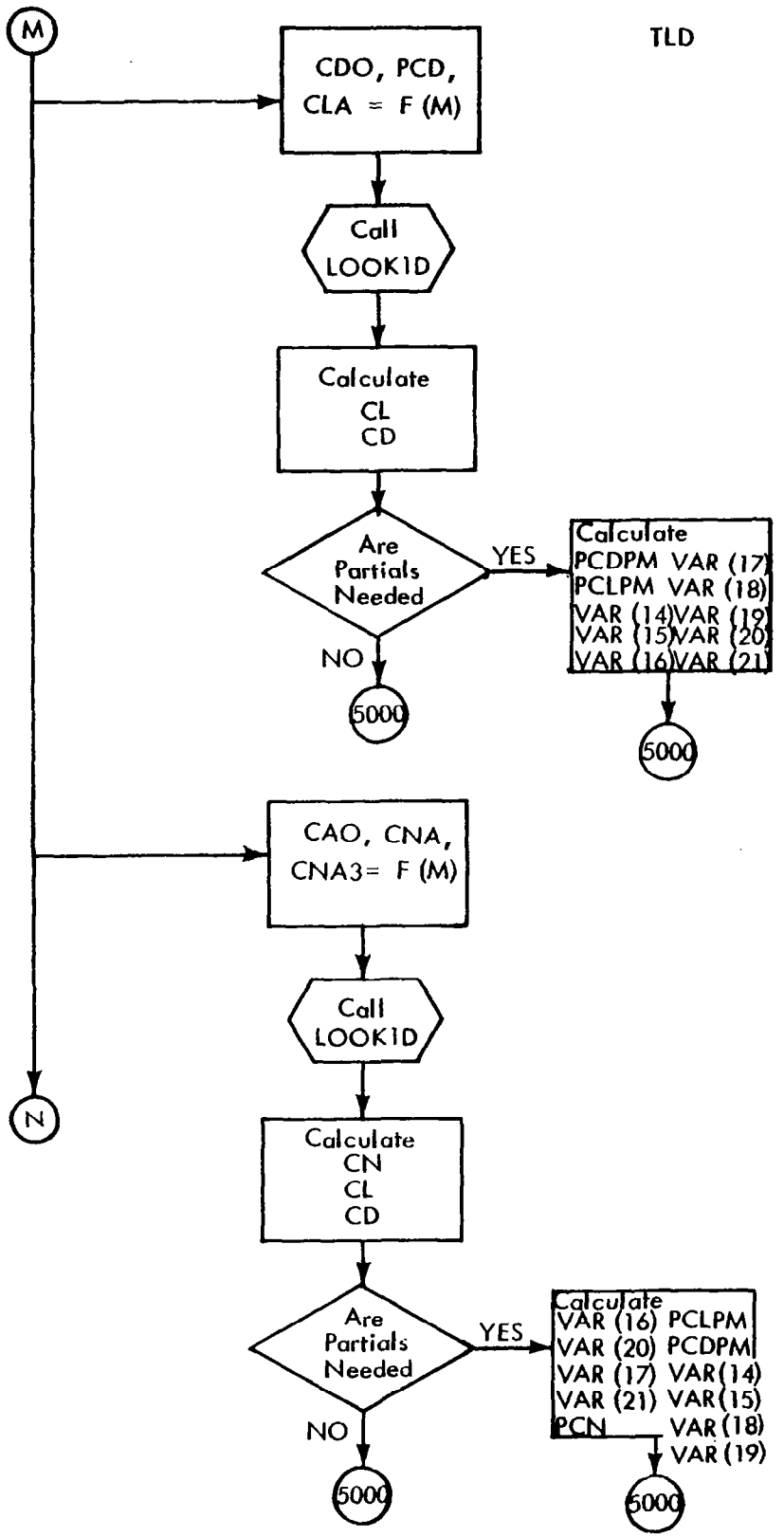

G
WDOT = F(ETA, H, V)
Call LOOK3D
Calculate VAR (3)
2000
WDOT = F (H, M, ETA)
Call LOOK3D
Calculate VAR (3)
Are Partials Needed
YES
Calculate VAR (36) VAR (38) VAR (35)
NO
2000
2000
SFC = F (ETA, M, H)
H
I

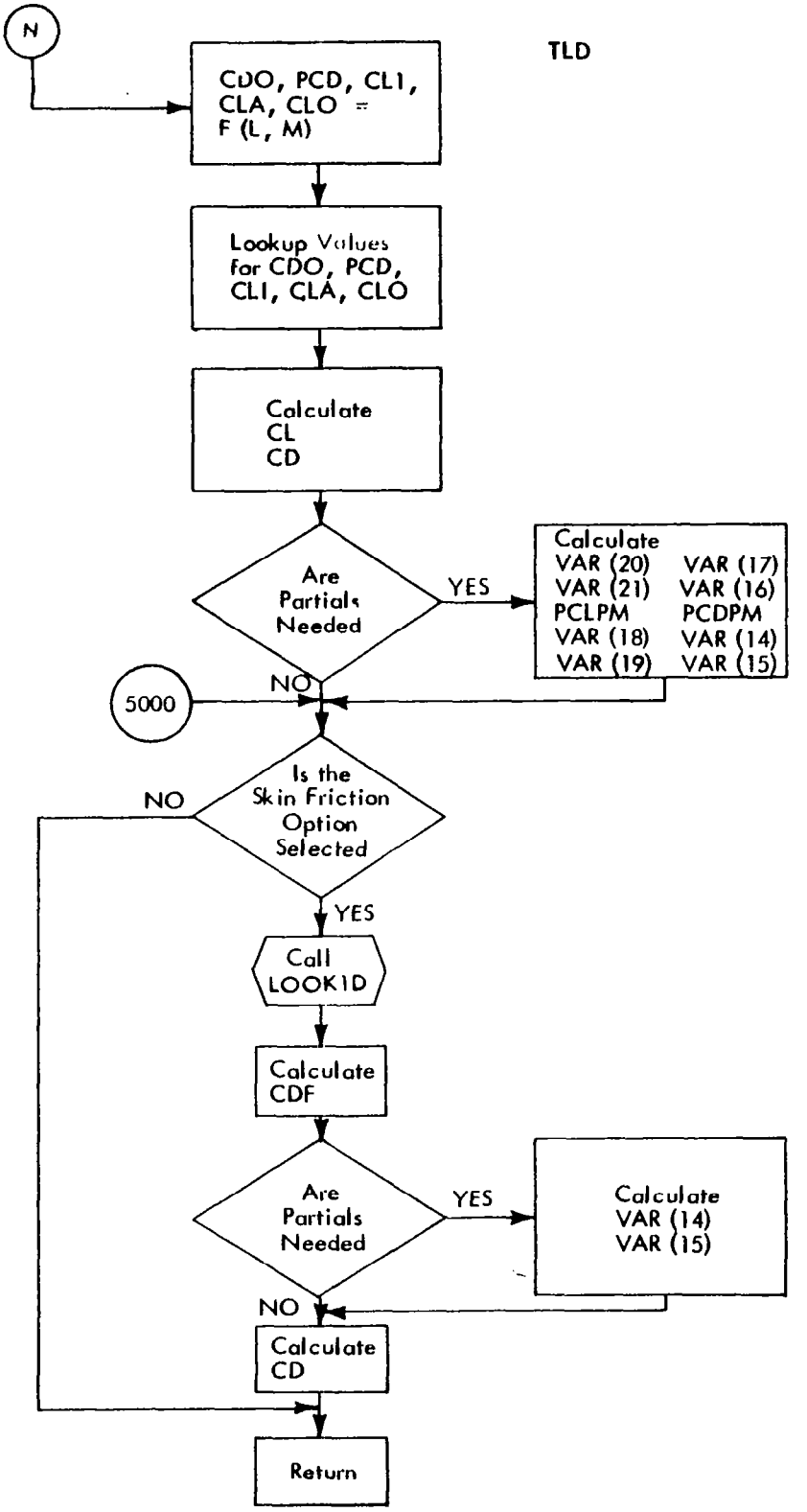

H
I
TLD
Call
LOOK3D
Are
Partials
Needed
YES
Calculate
VAR (38)
VAR (37)
66
NO
Calculate
VAR (3)
Are
Partials
Needed
YES
Calculate
VAR (36)
VAR (35)
NO
2000
2000
SFC = F (Alpha
M, H)
Call
LOOK3D
Calculate
VAR (3)
Are
Partials
Needed
YES
Calculate
VAR (38)
VAR (37)
NO
66
66
J


J
WDOT= F (ETA, M, H,)
Call LOOK3D
Set VAR (37) = 0.
86
Calculate VAR (3)
Are Partials Needed
YES
Calculate VAR (36) VAR (35)
NO
2000
2000
WDOT=F(Alpha, M, H)
Call LOOK3D
Set VAR (38) = 0.
86


2000
Scale Data Due to Number of Engines
Transfer to Aerodynamics Option Selected by Input
Vacuum Set CL = 0. CD = 0.
Zero VAR (14) Thru VAR (21)
5000
CD, CL = F (H, M, A)
Call LOOK3D
Are Partials Needed
YES
Calculate VAR (14) VAR (15)
NO
K
L


K
L
Call LOOK3D
Are Partials Needed
YES
Calculate VAR (18) VAR (19) VAR (17) VAR (21)
NO
5000
5000
CD, CL = F (L, M, A)
Call LOOK3D
Are Partials Needed
YES
Calculate VAR (14) VAR (15)
NO
M
Call LOOK3D
Are Partials Needed
YES
Calculate VAR (18) VAR (19)
NO
5000
5000


TLD
M
CDO, PCD,
CLA = F (M)
Call
LOOK1D
Calculate
CL
CD
Are
Partials
Needed
YES
NO
Calculate
PCDPM VAR (17)
PCLPM VAR (18)
VAR (14) VAR (19)
VAR (15) VAR (20)
VAR (16) VAR (21)
5000
5000
CAO, CNA,
CNA3= F (M)
Call
LOOK1D
N
Calculate
CN
CL
CD
Are
Partials
Needed
YES
NO
Calculate
VAR (16) PCLPM
VAR (20) PCDPM
VAR (17) VAR (14)
VAR (21) VAR (15)
PCN VAR (18)
VAR (19)
5000
5000


TLD
N
CDO, PCD, CL1, CLA, CLO = F (L, M)
Lookup Values for CDO, PCD, CL1, CLA, CLO
Calculate CL CD
Are Partials Needed
YES
Calculate
VAR (20) VAR (17)
VAR (21) VAR (16)
PCLPM PCDPM
VAR (18) VAR (14)
VAR (19) VAR (15)
NO
5000
Is the Skin Friction Option Selected
NO
YES
Call LOOK1D
Calculate CDF
Are Partials Needed
YES
Calculate
VAR (14)
VAR (15)
NO
Calculate CD
Return

UCALC — Control Variable History Update

Purpose

UCALC is assigned the task of computing the control variable history for the next trajectory.

Method

The control variable history used to generate the last valid trajectory and the impulse response history computed during the last backward integration is recovered from the KLAM unit. The Lagrange multipliers are used to compute a new control history by

$$\bar{u}_{NEW} = \bar{u}_{OLD} + \left[UULAM\right]\left[\frac{\mu_s}{\gamma_s}\right]$$

and $\bar{u}_{NEW}$ is stored on the KPAR unit. Next the first part of $\bar{u}_{NEW}$ that will fit in core is stored in TIMEU and the remainder is written on the KTAN unit. STP1 reads KTAN to accomplish the control history data overlay in the TIMEU array.

The Lagrange multipliers, computed in MATRIX, are transmitted to UCALC by

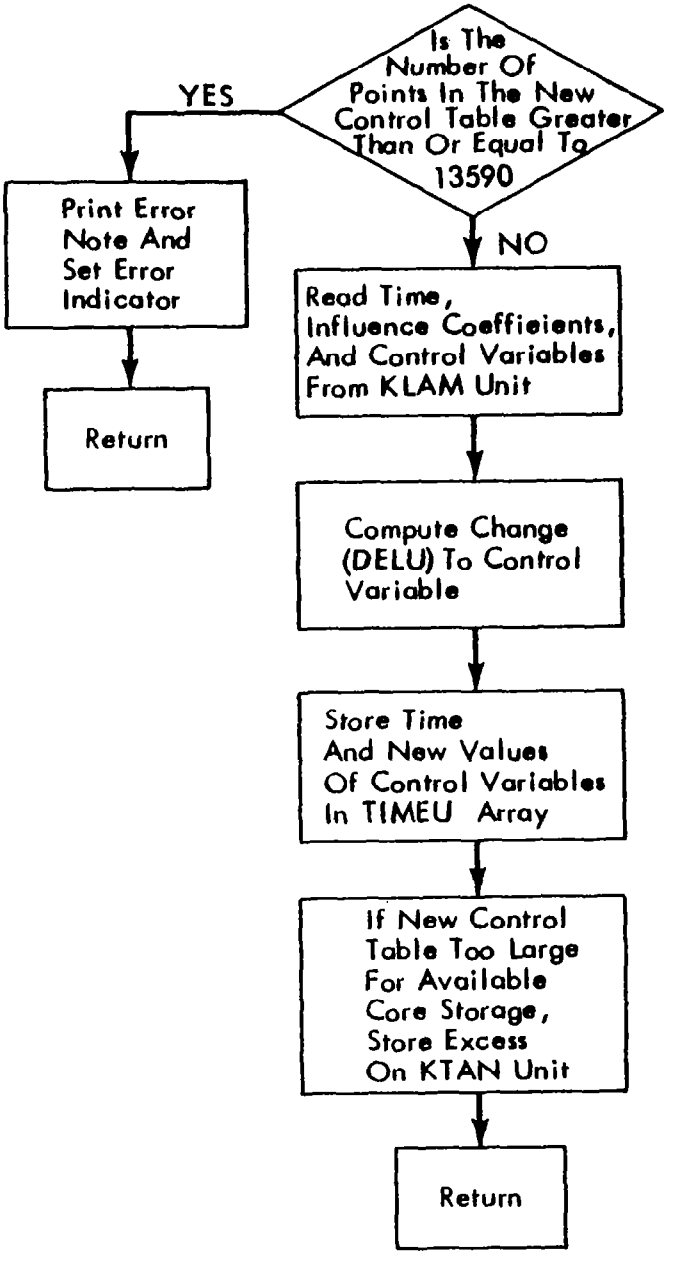
CALL UCALC (XMU)

where

XMU = the array of Lagrange multipliers

Storage Used

532 cells


Is The Number Of Points In The New Control Table Greater Than Or Equal To 13590
YES
NO
Print Error Note And Set Error Indicator
Return
Read Time, Influence Coeffieients, And Control Variables From KLAM Unit
Compute Change (DELU) To Control Variable
Store Time And New Values Of Control Variables In TIMEU Array
If New Control Table Too Large For Available Core Storage, Store Excess On KTAN Unit
Return

VALID — Valid Step Monitor

Purpose

VALID is called from MATRIX at the end of every valid step and is assigned the following tasks:

1) Decides whether to accept or reject the valid step.
2) Update free initial condition state variables.
3) Tighten temporary tolerance bands.
4) Updates INDSIC array.
5) Updates CPSI array.

Method

A detailed description of the logic used by VALID is given in Appendix C. If VALID decides to accept the valid step, switches are triggered to allow a reverse integration. If the valid step is rejected, the KSCR and KPLT units are backspaced to the start of the previous iteration. The new valid trajectory then will replace the rejected one.

Elements not in COMMON are transmitted between MATRIX and VALID through the calling sequence

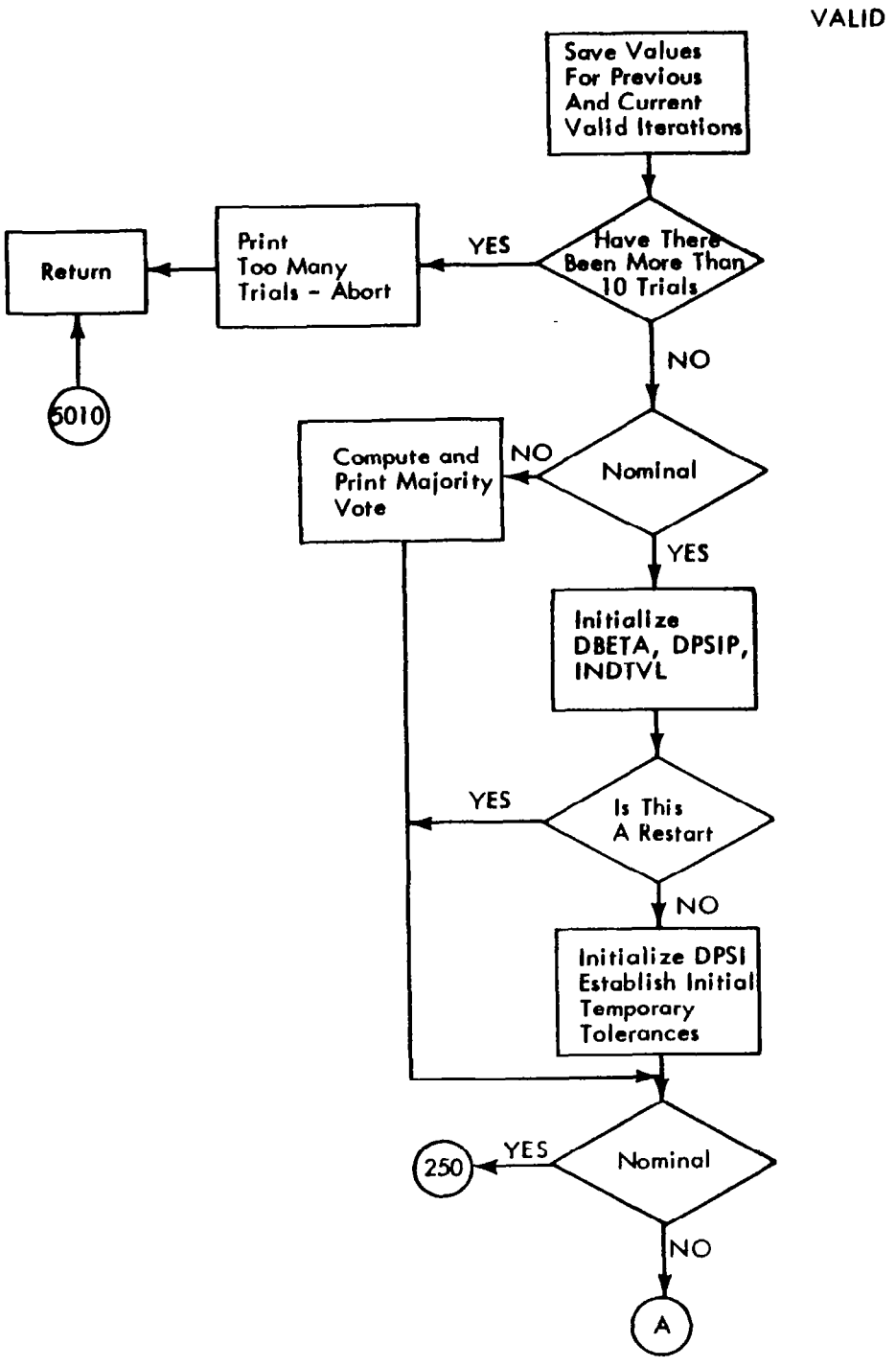
CALL VALID (DBETA, DPSI, DPSIP, INDTVL, XMU, XLAMDX, XX)

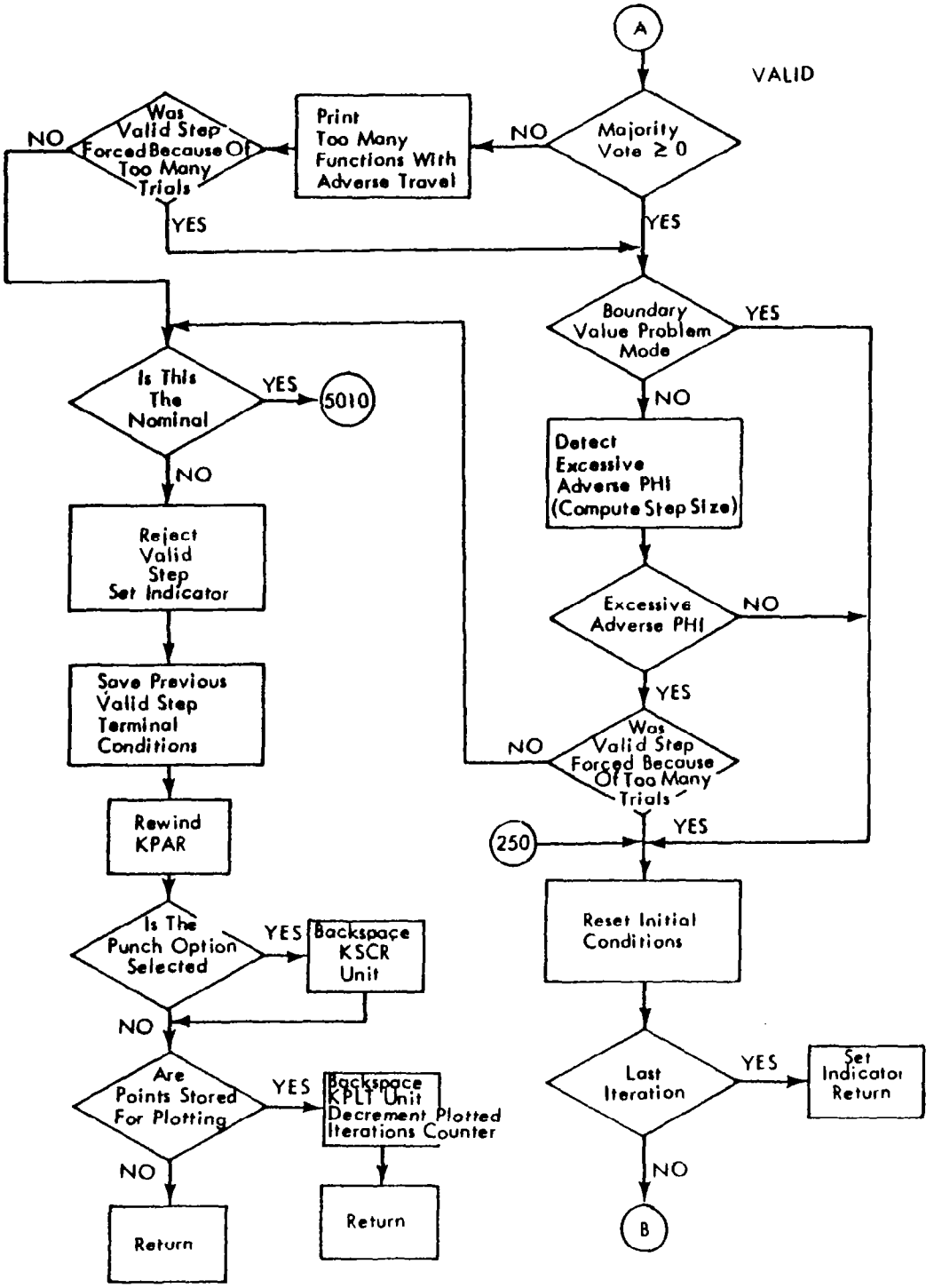
where

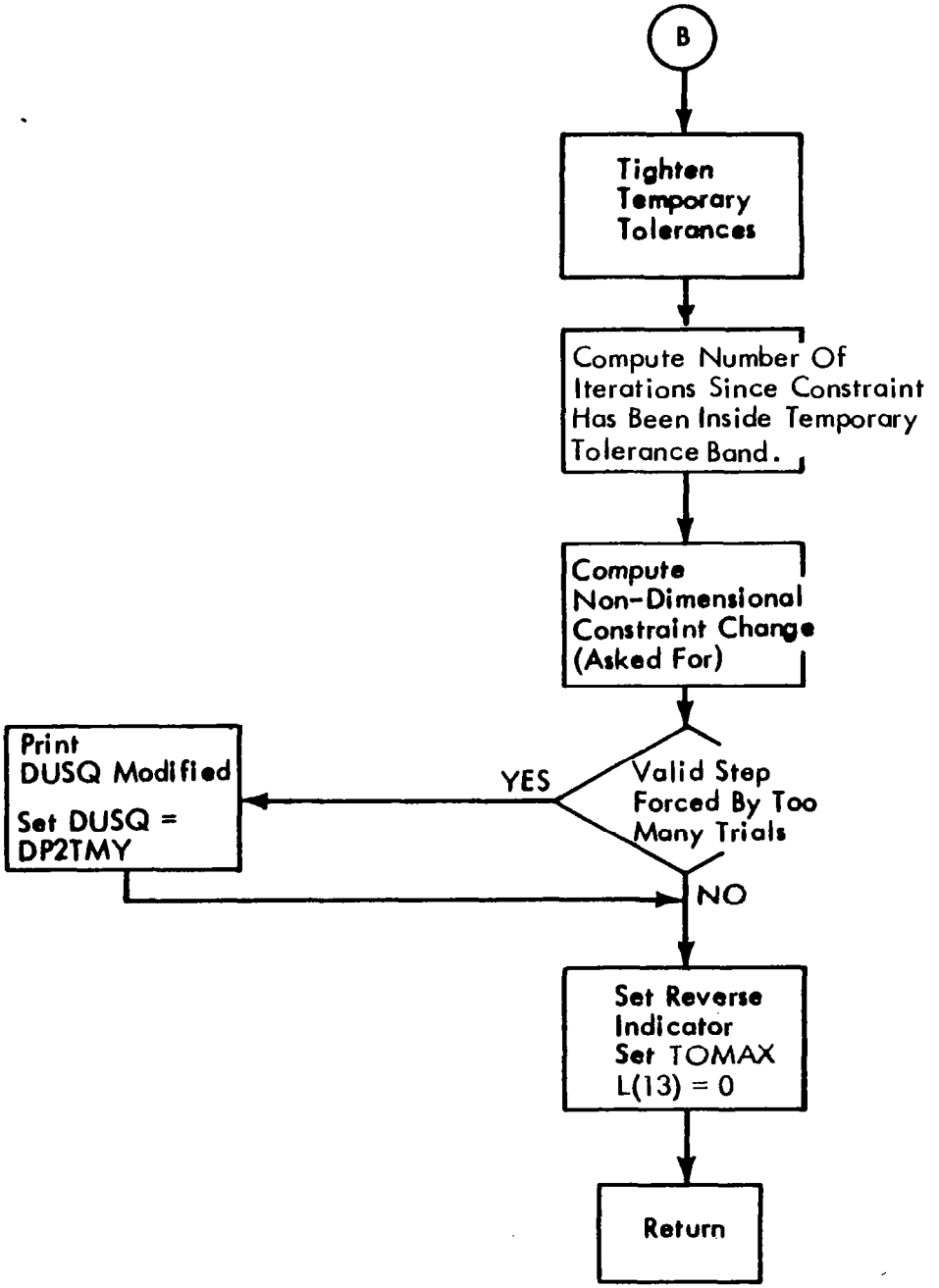
DBETA	= the change in constraints modified by the end point change due to the variation of initial conditions
DPSI	= actual change in constraints measured from a previous valid step
DPSIP	= predicted change in the constraint end points
INDTVL	= constraint travel indicator
XMU	= array of Lagrange multipliers, $\left[\bar{\mu}_s, \bar{\nu}_s\right]$
XLAMDX	= predicted change in the constraints due to variable initial conditions
XX	= array of constraint end points

Storage Used

735 cells


VALID
Save Values For Previous And Current Valid Iterations
Have There Been More Than 10 Trials
YES
Print Too Many Trials - Abort
Return
5010
NO
Nominal
NO
Compute and Print Majority Vote
YES
Initialize DBETA, DPSIP, INDTVL
Is This A Restart
YES
NO
Initialize DPSI Establish Initial Temporary Tolerances
Nominal
YES
250
NO
A


A
VALID
Majority Vote ≥ 0
NO
Print Too Many Functions With Adverse Travel
Was Valid Step Forced Because Of Too Many Trials
NO
YES
YES
Boundary Value Problem Mode
YES
NO
Detect Excessive Adverse PHI (Compute Step Size)
Excessive Adverse PHI
NO
YES
Was Valid Step Forced Because Of Too Many Trials
NO
YES
250
Reset Initial Conditions
Last Iteration
YES
Set Indicator Return
NO
B
Is This The Nominal
YES
5010
NO
Reject Valid Step Set Indicator
Save Previous Valid Step Terminal Conditions
Rewind KPAR
Is The Punch Option Selected
YES
Backspace KSCR Unit
NO
Are Points Stored For Plotting
YES
Backspace KPLT Unit Decrement Plotted Iterations Counter
NO
Return
Return


B
Tighten Temporary Tolerances
Compute Number Of Iterations Since Constraint Has Been Inside Temporary Tolerance Band.
Compute Non-Dimensional Constraint Change (Asked For)
Valid Step Forced By Too Many Trials
YES
Print DUSQ Modified
Set DUSQ = DP2TMY
NO
Set Reverse Indicator
Set TOMAX
L(13) = 0
Return

VARIC — Variable Initial Conditions

Purpose

VARIC computes the direction and magnitude to perturb the free initial conditions to improve performance and also controls the maximum payload option. This subroutine is called after every reverse integration and results in an update of the DELX array.

Method

The method used in VARIC is discussed in detail in the analytical development section.

Data not in COMMON and computed in MATRIX are transmitted to VARIC by

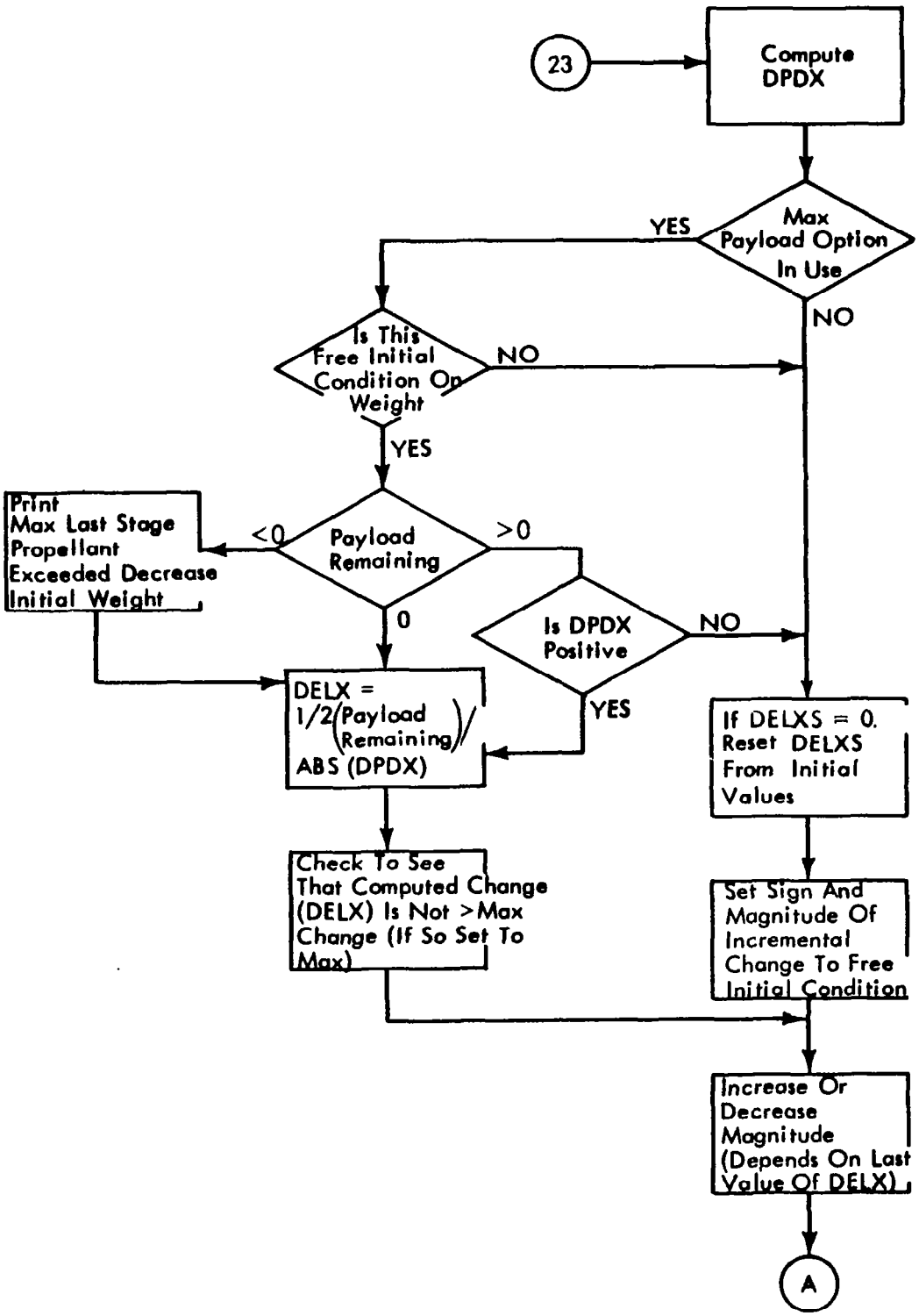
CALL VARIC (SFISSD)

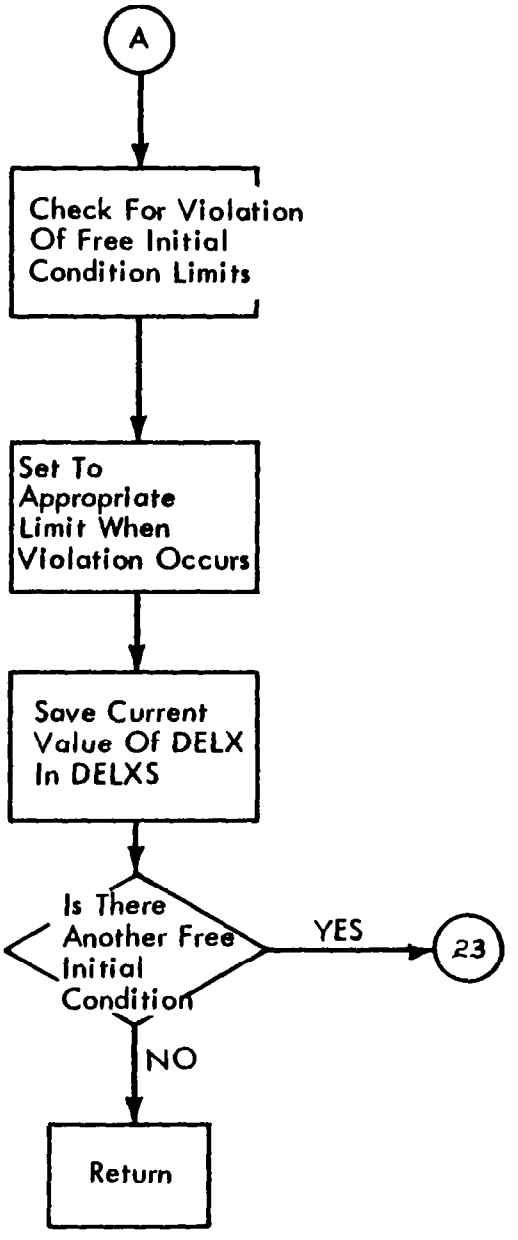
where

SFISSD = $d\bar{\psi}' I_{\psi\psi}^{-1}$

Storage Used

284 cells


23
Compute DPDX
Max Payload Option In Use
YES
NO
Is This Free Initial Condition On Weight
NO
YES
Print Max Last Stage Propellant Exceeded Decrease Initial Weight
<0
Payload Remaining
>0
0
Is DPDX Positive
NO
YES
DELX = 1/2(Payload Remaining)/ ABS (DPDX)
If DELXS = 0. Reset DELXS From Initial Values
Check To See That Computed Change (DELX) Is Not >Max Change (If So Set To Max)
Set Sign And Magnitude Of Incremental Change To Free Initial Condition
Increase Or Decrease Magnitude (Depends On Last Value Of DELX)
A


A
Check For Violation Of Free Initial Condition Limits
Set To Appropriate Limit When Violation Occurs
Save Current Value Of DELX In DELXS
Is There Another Free Initial Condition
YES
23
NO
Return

Program Flexibility. — One of the main features of STOP is its flexibility. The program has been modularized on both the macro and micro levels. On the macro level, for instance, the steepest-ascent block of the program is linked to the equations of motion by only one subroutine, ANPRTL. Thus, the entire equations of motion block may be altered or even replaced without requiring any changes in the steepest-ascent block. Every attempt has been made to assign one task to each subroutine, and to make each subroutine as independent as possible from the others. Finally, the flow has been organized so that there is one subroutine that is called once per data case, INITAL; another called once per stage, INITCO; one called once per forward trajectory, EXEC; etc.

On the micro level, the equations of motion and partial derivatives are modularized. This feature allows the selection of any subset of the 40 available equations for a given problem. Also, equations may be added or deleted easily. A few examples of the program flexibility are given below.

Example 1: Additional printout variable example: Suppose it is desired to print, Beta = $\sqrt{|M^2 - 1|}$, which is not currently available in the AK array. AK(39), not in current use, is available for growth.

Step 1. In subroutine STP1 add the FORTRAN statement
AK(39) = SQRT (ABS (AK(5)**2-1.))

Step 2. In subroutine BLOCK put the words (////BE) and (TA////)
in DESXUK (159) and DESXUK(160), respectively.

Step 3. Set NC(119) = 1 in the input data deck. This selects
AK(39) to be printed in the forward trajectory output.

No other changes are required.

Example 2: Additional placard example: Suppose it is desired to constrain the weight flow, which is not currently an available placard. X(K37) is an available state variable for placard additions.

Step 1. In subroutine FPROG, after the two FORTRAN statements:

```
37 CONTINUE
ITAB = ITAB + 2
```

add a call to subroutine PLAC. If the $\dot{W}$ limit is a function of time, the call would be,

```
CALL PLAC(II, ITAB, 1, T, 0., 0., AK(7))
```

Step 2. In subroutine ANPARP, after the statement:

```
37 CONTINUE
```

add the partials of F(K37) with respect to each state and control variable. Note that the call to PLAC generates F(K37) = $(\dot{W} - \dot{W}_{limit})^2$ for the portion of the trajectory where the fuel flow exceeds the limit value(s). Some typical partials are:

$$\frac{\partial F(K37)}{\partial H} = 2\,(\dot{W} - \dot{W}_{limit})\,\frac{\partial \dot{W}}{\partial H}$$

$$\frac{\partial F(K37)}{\partial V} = 2\,(\dot{W} - \dot{W}_{limit})\,\frac{\partial \dot{W}}{\partial V}$$

$$\frac{\partial F(K37)}{\partial \theta} = 2\,(\dot{W} - \dot{W}_{limit})\,\frac{\partial W}{\partial \alpha}\,\frac{\partial \alpha}{\partial \theta}$$

$$\frac{\partial F(K37)}{\partial \eta} = 2\,(\dot{W} - \dot{W}_{limit})\,\frac{\partial \dot{W}}{\partial \eta}$$

or in FORTRAN

```
TERM = 2 * (AK(7) - S(I, 2))

PFX(K37, K2) = TERM * VAR(35)

PFX(K37, K4) = TERM * VAR(36)

PFU(K37, I1) = TERM * VAR(37) * VAR(41)

PFU(K37, I3) = TERM * VAR(38)
```

Step 3. In subroutine BLOCK the heading for $\dot{W}$ penalty function, which consists of the words (/W/DØ) and (T/PF//), is placed in DESXUK(75) and DESXUK(76), respectively.

Step 4. Set NC(157) = -1, 1, or 2 in the input data deck depending upon the type of limit desired, i.e., minimum, maximum, or corridor respectively.

Step 5. Add a placard table to the data deck for the fuel flow limit.

Example 3: Equation replacement example: It is a relatively straightforward procedure to replace the complete equation of motion block. This capability is demonstrated in the example below.

Consider a cat standing on a free-wheeling disk at point A as shown in figure 22. The cat wishes to get to point B on the disk in minimum time. The cat is constrained to the disk for the entire mission. The properties of the cat are

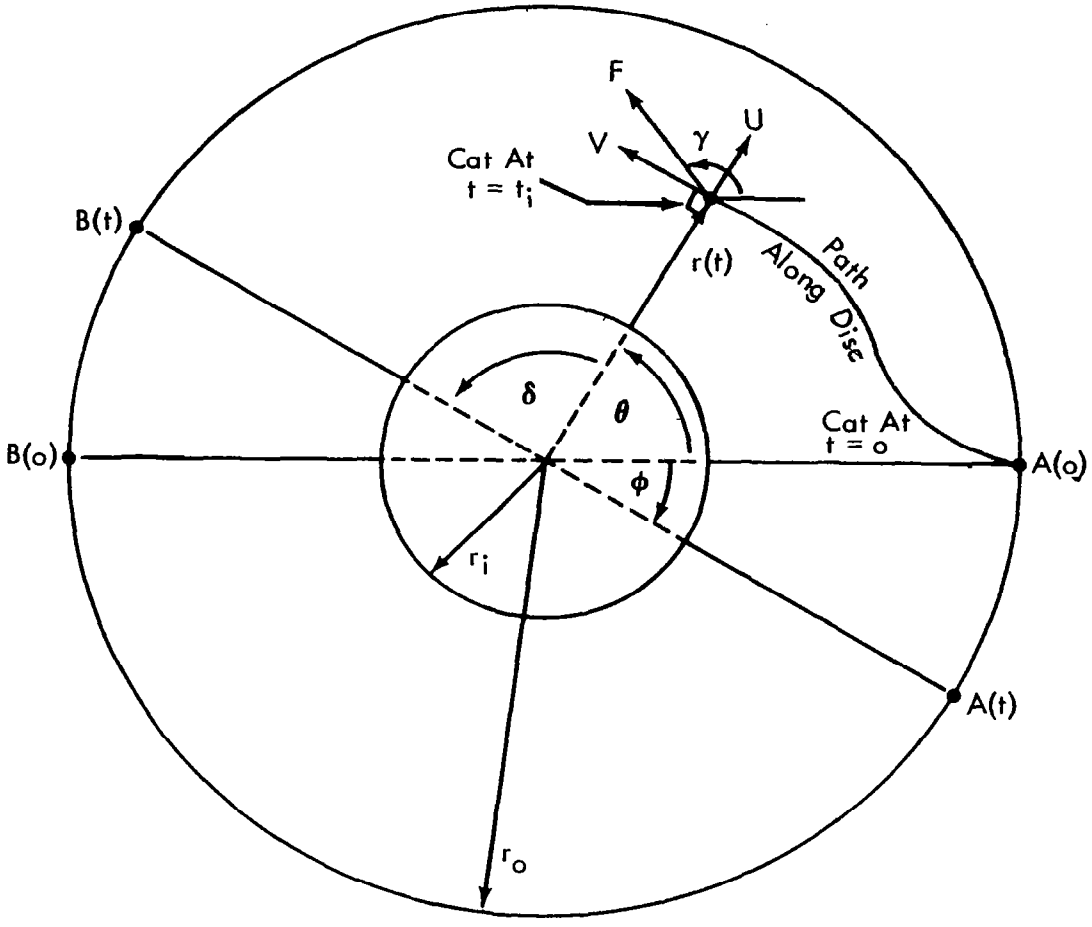


Figure 22. CAT ON A DISC PROBLEM

$m = 0.5$ slug

$F = 1$ lb

and the properties of the disk are

$r_o = 10$ ft

$r_i = 7$ ft

$I = 70$ slug ft^2

The pertinent equations of motion are:

$$\dot{r} = U$$
$$\theta = V/r \text{ DEG}$$
$$\dot{U} = V^2/r + F/m \text{ Cos}(\gamma - \theta)$$
$$\dot{V} = -UV/r + F/m \text{ Sin}(\gamma - \theta)$$
$$\dot{\phi} = \omega$$
$$\dot{\omega} = F/I \text{ r Sin}(\gamma - \theta) \text{ DEG}$$
$$\dot{s} = -V/r \text{ . DEG} - \omega$$
$$\dot{t} = 1.$$
$$\dot{PF} = (r - r_{limit})^2$$

Now generalizing to the STOP formulation, the state variables are:

X(K1) = r (ft)

X(K2) = θ (deg)

X(K3) = U (fps)

X(K4) = V (fps)

X(K5) = ϕ (deg)

X(K6) = ω (deg/sec)

X(K7) = δ (deg)

X(K8) = t (sec)

X(K21) = PF (ft^2/sec^2)

The control variable is:

U(I1) = γ (deg)

The initial conditions are:

$X_o(K1) = 10$

$X_o(K2) = 0$

$X_o(K3) = 0$

$X_o(K4) = 0$

$X_o(K5) = 0$

$X_o(K6) = 0$

$X_o(K7) = 0$

$X_o(K21) = 0$

The problem statement is:

$\Omega = X(K7) = 0$ stopping quantity

$\phi = t$ (minimize) performance

$\psi_1 = r = 10$

$\psi_2 = R\ PF = 0$

The partials required are:

$\partial \dot{r}/\partial u = 1.$

$\partial \theta/\partial r = -V/r^2$ DEG

$\partial \theta/\partial V = 1/r$ DEG

$\partial \dot{U}/\partial r = -V^2/r^2$

$\partial \dot{U}/\partial V = 2V/r$

$\partial \dot{U}/\partial \theta = F/m\ \mathrm{Sin}(\gamma-\theta)$ RAD

$\partial \dot{U}/\partial \theta = -F/m\ \mathrm{Sin}(\gamma-\theta)$ RAD

$\partial \dot{V}/\partial r = +UV/r^2$

$\partial \dot{V}/\partial U = -V/r$

$\partial \dot{V}/\partial V = -U/r$

$\partial \dot{V}/\partial \theta = -F/m\ \mathrm{Cos}(\gamma-\theta)$ RAD

$\partial \dot{V}/\partial \gamma = F/m\ \mathrm{Cos}(\gamma-\theta)$ RAD

$\partial \dot{\phi}/\partial \omega = 1.$

$\partial \dot{\omega}/\partial r = +F/I\ \mathrm{Sin}(\gamma-\theta)$ DEG

$$\partial\dot{\omega}/\partial\theta = -F/I\ r\ \mathrm{Cos}\ (\gamma-\theta)$$

$$\partial\dot{\omega}/\partial\gamma = +F/I\ r\ \mathrm{Cos}\ (\gamma-\theta)$$

$$\partial\dot{\delta}/\partial r = -\partial\dot{\theta}/\partial r$$

$$\partial\dot{\delta}/\partial V = -\partial\dot{\theta}/\partial V$$

$$\partial\dot{\delta}/\partial\omega = -1.$$

$$\partial\dot{P}F/\partial r = 2\,(r - r_{limit})$$

The subroutines requiring changes are FPROG, ANPRTL, and ANPARP. In addition, a dummy subroutine AKSTP may be used and part of STP1 may be deleted to save computer time.

As a matter of interest, the solution to the above problem is shown in figure 23. The problem solution required approximately 4 hours of engineering time and 10 minutes of computer time.

Program and data overlay. — Due to the overall size of STOP, it is not possible to fit all the subroutines and data into the IBM 7094 available core as is discussed in "Operating Information."

It was necessary to minimize core requirements while maintaining the program's capability, flexibility, and execution speed in that order of importance. To this end, program overlay, data overlay, and equivalence are used.

Program overlay: The STOP program can be broken into five major functional groups of subroutines: data initialization, forward integration, backward integration, convergence control, and plotting control. Each of these groups could operate separately if a main program could control their execution and interlinking data.

Using the IBSYS overlay feature, the program was divided into the above mentioned groups or links and three additional links. Two of these additional links were added to minimize the storage required for the forward integration subroutines, and the last link controls the final output for a data case. These nine links are described below.

LINK 0 - Controls the program flow or execution from link to link and remains in core all the time.

LINK 1 - Forward integration link
LINK 2 - Numerical partial check link
LINK 3 - Nominal guidance link

} Links 1, 2, & 3 comprise the forward integration

LINK 4 - Convergence control link

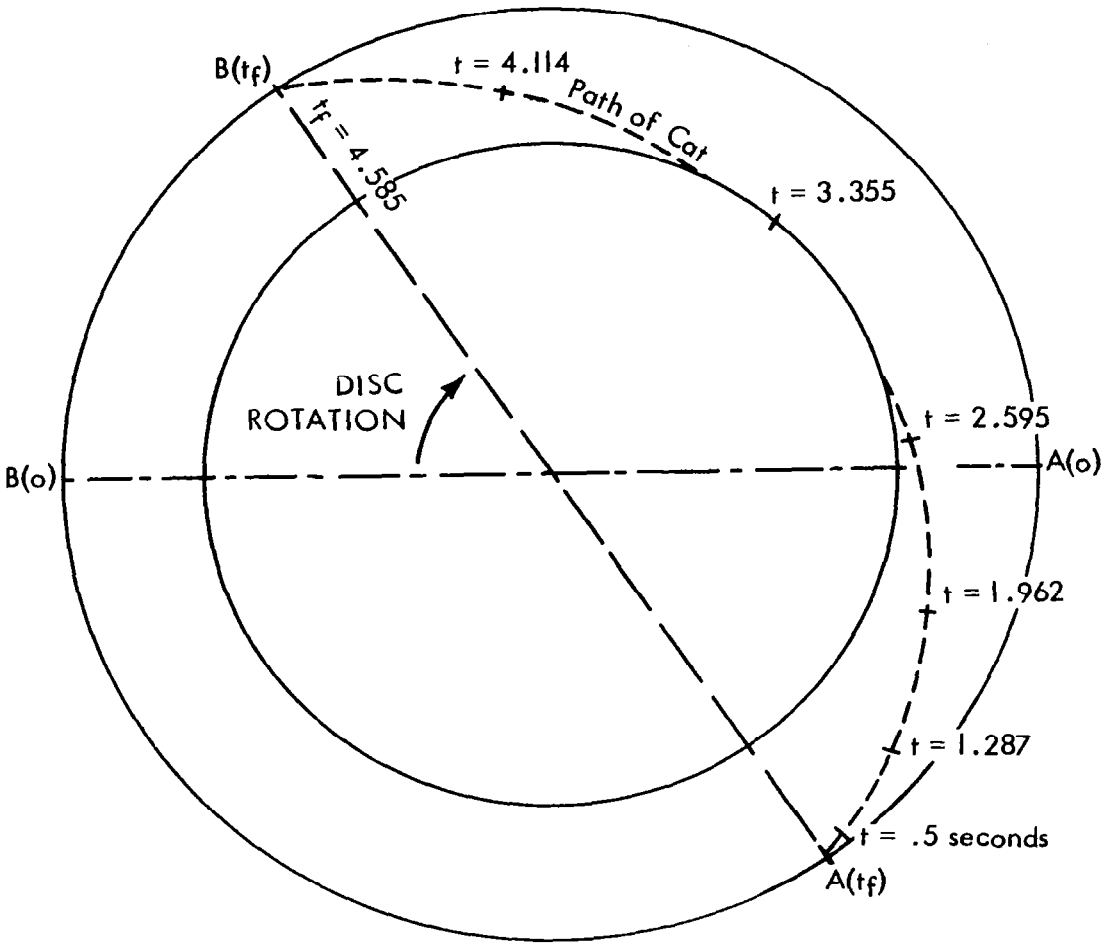


Figure 23. PATH OF CAT ON DISC

LINK 5 - Backward integration link

LINK 6 - Initialization link

LINK 7 - Final output link

LINK 8 - Plotting link

Under the IBSYS overlay feature, unless otherwise specified, all links will be loaded onto the I/O unit assigned as SYSUT2. In an effort to speed up the loading of links, the two most frequently used links (forward integration and convergence control) are loaded on separate I/O units by themselves, and these I/O units are rewound after each loading.

See "Operating Information" for a breakdown of links by subroutine. Figure 17 illustrates the program overlay and gives approximate core requirements for each link.

Data overlay: Since all of the input data for large cases will not fit in core, some of the data is written on scratch tapes and overlayed during execution.

Control table overlay: The input control variable table is normally stored in the TIMEU array, dimensioned to hold 1000 data points. If the input table exceeds 1000 points, that part of the table which won't fit in TIMEU is written in binary format on the I/O unit referenced as KTAN. After each forward integration step, the EXEC subroutine tests the current time against the highest time currently stored in the TIMEU array to see if more data should be read from KTAN into TIMEU.

A similar procedure is used in subroutine UCALC, where the updated control table is calculated. In this subroutine, the I/O unit referenced as KPAR is used for temporary storage as the updated control table is built. Again, a maximum of 1000 words of the control table are stored in TIMEU, and the remainder written on KTAN.

If the total number of data points in the updated control table exceeds 13,590, the data case is aborted and a comment printed.

Nonstage and stage-dependent data overlay: Except for the control and restart tables, all other tables, both nonstage and stage dependent, are initially read into the Z array, dimensioned 7000.

The nonstage dependent data is read first and written in binary format on the I/O unit referenced as KDAT. The stage-dependent data is then read, one stage at a time and written on KDAT. If the sum of the nonstage-dependent and stage-dependent data for any stage exeeds 7000 points, a message is printed and the data case aborted.

During the forward integration, the Z array contains the nonstage-dependent and stage-dependent data for the current stage. At the beginning of each new stage, the data for that stage is read from KDAT into Z, replacing or overlaying the data for the preceding stage. After the last stage has been integrated, for each valid forward trajectory, the end points of that trajectory are written on KDAT. These end points at the end of a data case are read from KDAT and printed as the trajectory summary.

At the start of each forward trajectory, trial or valid, KDAT is rewound. The nonstage-dependent tables are read and the stage-dependent tables for the current stage are read back into the Z array. Reading the nonstage-dependent tables into the Z array at the start of each iteration is necessary to allow other links to use the Z array for storage while they are executing.

Equivalence. — The equivalence feature is used liberally throughout the program to make maximum use of available core in each link. Another purpose of the equivalence is to allow use of parts of the NC array (or its equivalent) for subscripting. The two major arrays to which other variables are equivalenced are NC and Z.

Z array equivalencing: Figure 24a illustrates the manner in which the Z array is used. The first column, represented by the shaded area, presents the usage of the Z array during forward integration. Following the weighting table, which is always in the Z array and never equivalenced, the nonstage-dependent data is stored starting at Z(201). Then the stage-dependent data for any one stage is stored. Both the nonstage-dependent data and the stage-dependent data are destroyed by equivalence during the backward integration and then reloaded into the Z array from KDAT in subroutine EXEC.

The Z array, from Z(201) through Z(2872), is used during the backward integration (link 5) for temporary storage of the state variable values, derivatives, integration errors, and other values needed by RKVS (Runge-Kutta variable step) integration package.

The Z array from Z(2900) through Z(7000) and the TIMEU array, (1-1000), are used during the backward integration for storage of the partial derivatives read from KPAR.

NC array equivalencing: The NC array is equivalenced in subroutines throughout the program to make use of subscripted subscripts; i.e. use the NC as a subscript. Figure 24b shows the manner of NC equivalencing. From the figure it can be seen that the control variable indicators, NC(77) to NC(80), are equivalenced with the control variable indices I1, I2, I3, and I4. The state variable and placard indicators, NC(121) through NC(160), are equivalenced with state and placard variable indices K1 through K40.

Trouble Shooting.— The purpose of this section is to provide a quick reference to aid in correcting typical problems encountered during the operation of

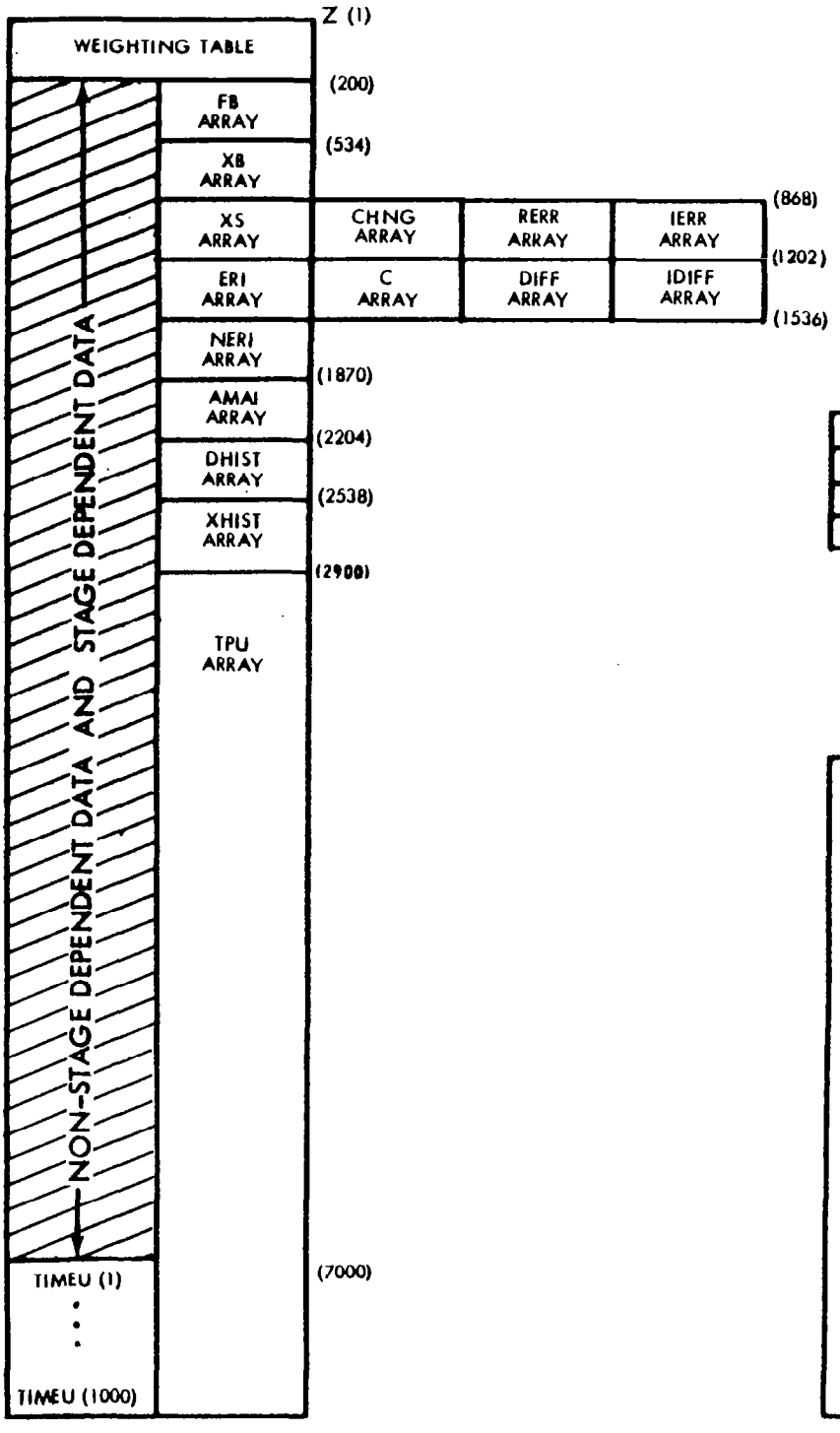


Figure 24a.
Z ARRAY EQUIVALENCING

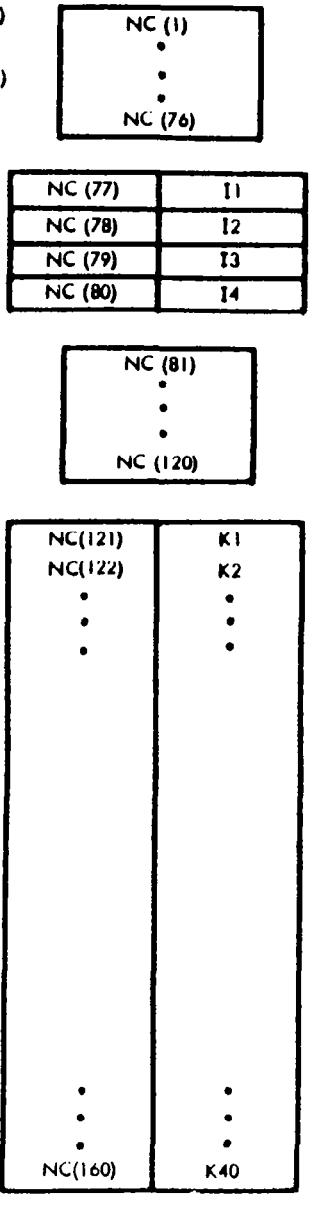


Figure 24b.
NC ARRAY EQUIVALENCING

STOP. The most common problems, as with any large, flexible computer program, are input errors. Another common problem is slow convergence and/or excessive run time. Also, STOP may abort during a run, either because of a problem it has detected or an error detected by the computing system.

It is always possible to restart STOP from the last valid iteration, whether termination was a successful exit, a STOP abort, a system abort, or a system failure. Problems that execute more than one hour have a reasonable probability of experiencing a system failure, which makes restart capability a necessary requirement.

An input checklist and a trouble-shooting guide are provided below to assist the user in obtaining satisfactory results from STOP.

Input checklist: The following checklist will aid in data deck preparation. It includes the most common input errors encountered by users not familiar with STOP. The checklist assumes that all keypunch errors have been removed and that all data are placed correctly in their fields. In particular, check all integer data to make sure it is right adjusted. The checklist is in the same order as the input instruction guide.

Card set 1 (title card)

No problem

Card set 2 (controls)

Check MSTAGE. Remember that if a tilt maneuver is used, it counts as a stage.

Card set 3 (NC controls)

1. Check last card for blank field in columns 61-65. If last card is full, add a blank card.
2. Check NC (4). It should equal the number of enroute and terminal constraints plus payoff and stopping variables.
3. Check NC (16) and NC (17). NC (16) must be 1 if the input control table was punched by STOP. NC (17) must be 1 if the restart table is included.
4. Check NC (18) through NC (32). Only state variables selected for integration may be used as the stopping condition. Three basic requirements must be examined to be sure that the stopping condition is attainable. First, XSTP must be reached before the stopping time (TSTP) of the final stage. Second, the stopping condition must be compatible with the constraints selected (e.g. do not select altitude if the constraint on flight path angle is zero degrees). Finally, select a stopping condition that will always be monotonic in the last stage of flight and has a large terminal time derivative. Note that the stopping condition is only examined during the final stage.

Performance and constraint functions must be state variables selected for integration. Be sure that the sign of the performance variable index is plus for maximizing and minus for minimizing.

Constraint-dependent parameters and constraint tables must be input in the same order as they are in NC (20) to NC (32).

5. Check NC (121) — NC (160). Do not select state variables that are not needed, such as path range or the losses. Do not use latitude and heading for non-rotating-earth problems. Instead, formulate the problem in terms of equatorial flight. Also, do not select placards unless they are — or are in danger of — being violated. Note that run time is almost directly proportional to the number of selected state variables.

Card set 4 (initial condition data)

1. Check DUSQ if a restart table is not included. Values on the order of 100 are reasonable for airplane-type problems. For other vehicles, use an initial value of about one tenth of the controllable trajectory time.

2. Check FINNER and BINNER. Values of 18 are typical. The run time may be decreased by using 15's, but convergence may be slower. For rocket problems, values of 18 seem to work well, while for SST problems 15's are better. In general, difficult problems require settings that will result in at least 500 integration steps.

3. Check XO ( )'s to verify they are in the same order as the state variables selected. Also, if more than seven state variables are selected, there must be two or three cards of XO ( )'s. The XO's are a packed array of initial state variable values. Note, for example, that XO (7) does not necessarily equal X (K7) at the starting time.

Card set 5 (stage dependent parameters)

1. Check to be sure eight cards are present if MSTAGE is less than eight, otherwise 16 cards.

2. Check TSTP. The last stage requires a stopping time that will not be reached under usual conditions. It should be set to allow an abort if the stopping condition is not met. For VTO systems the first stage contains the tilt maneuver; TSTP for this stage represents the termination of the tilt maneuver.

3. Check DTMIN. Too large a value will slow convergence. SST problems behave badly if DTMIN is much larger than 0.1 second.

4. Check K_T and K_M. The first run should be a nominal only so that weight, thrust, and other data may be checked. K_M and K_T may be used to adjust errors in total propellant consumption and total impulse.

Card set 6 (constraint dependent parameters)

1. Check to be sure that one card is present for each enroute and terminal constraint and that they are in the same order as the constraints selected by the NC's.
2. Check the DPSI values. Very small values may result in slow convergence. Values of about 0.1 on angles and 100 on altitude are reasonable.

Card set 7 (free initial conditions)

1. Check to be sure there are NC (5) cards.
2. If a tilt maneuver is selected, $\Delta\gamma_o$ will be applied at end of tilt (start of stage 2). A large maximum value is required.
3. To restart with variable initial conditions:
   a. If weight is a variable initial condition, all positive-stage weights must be updated to be compatible with the weights from the valid step from which restart is desired. (card set 5, card subset 1). Zero and negative values are not updated.
   b. If a variable initial condition is selected on gamma and a tilt maneuver is selected, table 2 must be updated with the value of gamma at the beginning of stage 2 from the valid step from which restart is desired.
   c. All other state variables chosen as variable initial conditions are updated in card set 4, card 3. These values are also updated from the valid step from which restart is desired.

Card set 8 (nominal guidance data)

1. Check to be sure NC (13) cards are present.
2. Check the sign of NSW (see instructions).

Card set 9 (tables)

1. Check table 0 if constructed by the user. This table will not be extrapolated and, therefore, a sufficiently large time must be included as the last point. Multiple time points may be used at staging time. This table must always be present regardless of whether or not a nominal guidance is used.
2. Check the numbers of the placard tables (see notes at the end of the input instructions).
3. All tables after table 0 may be extrapolated (linearly) in any direction.

	SYMPTOM		DIAGNOSIS		REMEDY
1.	System abort during data input	a)	Table card count in error	a)	Check card count in each table
		b)	Deck or data setup error	b)	See appropriate sections of input instructions
		c)	Error in count of state variables, terminal constraints, free initial conditions, or guidance modes	c)	Make sure card set 4 contains at least four cards if more than seven state variables were selected (NC(121) - NC(160)). Make sure card set 5 contains 16 cards if MSTAGE is greater than seven. Make sure card set 6 contains one card for each selected constraint (terminal and enroute) (NC(4) - 2 cards required). Card set 7 must contain NC(5) cards and card set 8 must have NC(13) cards.
		d)	STG or END card missing or misplaced	d)	Check data deck
		e)	NC input error	e)	Check card set 3. Make sure last card has a blank field in columns 61-65.
		f)	Restart error	f)	Restart table required if NC(17) = 1. See output description.
2.	Trajectory fails to meet stopping condition or prints note, MAXIMUM STOPPING TIME EXCEEDED	a)	Stopping conditions not attainable on last stage	a)	Correct table 0 or use closed-loop options. See suggestions for NC(18) in checklist.
		b)	TSTP of last stage too small	b)	Increase TSTP of last stage. Trajectory will always abort if this time is reached.
3.	STOP abort with note, MAXIMUM ALLOWABLE TABLE STORAGE EXCEEDED		Input data tables exceed allowable storage for a stage (tables 1 through 30 plus tables 31 through 34 for a stage)		Reduce the amount of data or divide the stage-dependent tables into more stages

	SYMPTOM		DIAGNOSIS		REMEDY
4.	STOP abort with note, NUMBER OF POINTS STORED IN BACK-WARD INTEGRA-TION EXCEEDS MAXIMUM ALLOWABLE	a)	Backward integration too tight	a)	Lower BINNER by 3
		b)	Storage too frequent	b)	Increase ISTOR2 by 1 (Recommended solution)
5.	One or more constraints not met after 20 iterations	a)	Constraints not attainable	a)	Change problem statement
		b)	Solution nominal dependent	b)	Start from different nominal
		c)	Partial derivatives in error	c)	Check partial derivatives (NC(12) option)
		d)	Input weighting matrix problem	d)	Increase weighting of applicable control variable
		e)	Placard too nonlinear	e)	Nonlinearities are often caused by spikes in the placard table. Try redesigning placard so that the placard has a more severe violation in the spike region.
		f)	Placard corners or dis-continuities	f)	Avoid corners or discontinuities in placard tables.
6.	Excessive run time (Note: Run time is very prob-lem dependent. Simple problems may run several iterations per minute, while very difficult problems may require 1 hour per iteration)	a)	Unnecessary state vari-ables being integrated	a)	During the iterative procedure, integrate only the minimum number of equations (e. g. if the problem can be phrased in terms of flight around the equator, and bank control is not required, latitude and heading equations are not required). Run time is nearly proportional to the number of state variables.
		b)	Unnecessary constraints selected.	b)	Constrain only the state variables required.

	SYMPTOM		DIAGNOSIS		REMEDY
7.	Excessive end-point oscillations		Variable initial conditions allow too large a variation per iteration		Decrease initial increment per iteration (card set 7) for the free initial conditions.
8.	Slow convergence	a)	Weighting matrix problems	a)	Try decreasing the input weighting matrix elements on all control variables during the initial part of the trajectory. Also, decrease weighting on control variables that tend to be nonlinear (such as sweep)
		b)	Integration too loose	b)	Decrease DTMIN and increase BINNER and FINNER. The integration should never reach minimum step size.
		c)	Error in partials	c)	Check partials (NC(12) option)
		d)	Poor stopping condition	d)	Select a stopping variable that is monotonic and has a large time derivative at XSTP. Problems that stop on time usually behave better.
		e)	Constraint tolerances too tight	e)	Increase DPSI's (card set 6)
		f)	EPSLN too large	f)	EPSLN (card set 4) must be small enough so that end-point motion within the EPSLN band is small compared with the end-point motion predicted by the steepest ascent logic.
9.	STOP abort with note, CONVERGENCE FAILED		All constraints have moved less than their respective final tolerance bands and the change in performance is less than or equal to 10^{-6} times the value of performance on the previous valid step.		See remedy for symptom 8.

	SYMPTOM	DIAGNOSIS	REMEDY
10.	STOP termination with note, THE LAST TRAJECTORY APPEARS TO BE AN EXTREMAL	All constraints are inside their respective final tolerance bands, the change in performance is less than or equal to 10^{-6} times the value of performance on the previous valid step, and the denominator of the first Lagrange multiplier is less than or equal to one ten thousandth of $I_{\phi\phi}$.	Congratulations
11.	STOP abort with note, GRADIENT OF PHI TOO NEGATIVE TO CONTINUE	The denominator of the first Lagrange multiplier plus 10 percent of $I_{\phi\phi}$ is negative.	See remedy for symptom 8.
12.	STOP abort with note, OVERFLOW OCCURRED IN MATRIX INVERSION - ABORT	Overflow in inversion of $I_{\psi\psi}$ matrix.	Check problem statement and table 1.
13.	STOP abort with note, SINGULAR MATRIX - ABORT	Singular $I_{\psi\psi}$ matrix.	Check problem statement and table 1.

	SYMPTOM	DIAGNOSIS	REMEDY
14.	STOP abort with note, NUMBER OF TRIES PLUS REJECTED VALID STEPS FOR CURRENT ITERATION EXCEEDS 10 — ABORT	An excessive number of rejected valid steps in combination with an excessive number of trials have been run in an attempt to run an accepted valid step.	See remedy for symptom 8.

Plotting Information

The program will plot any state, auxiliary printout, or control variable versus time or any other state, auxiliary printout, or control variable. Each curve represents a particular iteration. Plotting is performed on either the Orthomat drafting machine or the Stromberg-Carlson (SC) 4020 microfilm recorder. Orthomat plots can be made on one of two pregridded forms: 11- by 17-inch mm vellum, or 8-1/2- by 11-inch mm vellum. SC 4020 plots may be made on 9- by 9-inch vellum (hard copy) or 16-mm microfilm.

Limitations. — The following are plotting limitations:

1) 7 plots maximum;

2) 6 curves per plot maximum

3) Multiple-data cases cannot be plotted;

4) X and Y axes annotation occurs every 2 cm.;

5) Plot titling consists of the following

LINE 1 — centered, 0.2 inch below the X axis
DEP. VAR. NAME VS INDEP. VAR. NAME

LINE 2 — centered, 0.2 inch below LINE 1
The first 72 characters taken from the title card on the data deck.

Input controls required for plotting. — The following values should be examined in the input when plotting is desired:

NC(1) ORTHOMAT

= 0, output will be on 11- by 17-inch pregridded mm vellum

= 1, output will be on 8-1/2- by 11-inch pregridded mm vellum

SC 4020

= 0, output will be 9- by 9-inch vellum (hard copy)

= 1, output will be 16 mm microfilm

= 2, output will be 9- by 9-inch vellum (hard copy)

= 3, output will be both hard copy and microfilm

NC(3) ORTHOMAT

= 0, multiple curves will be defined with plot symbols (see multiple curve identification)

= 1, multiple curves will be drawn with new colors (see multiple curve identification).

SC 4020

Not Used

NC(11) ORTHOMAT & SC 4020

= 0, No plots wanted

= +N, SC 4020 plots wanted

= -N, Orthomat plots wanted

Note: The nominal, or first trajectory, the last trajectory, and every |N| valid trajectories will be plotted, until NITC is reached or six trajectories plotted. Example: NITC = 10, NC(11) = -2. The Orthomat will plot the nominal and iterations 2, 4, 6, 8, and 10.

Each plot to be made is defined by a pair of integers that are input in the NC array, starting with NC(61).

Figure 11 gives the plot index for all variables.

NC(61) Dependent variable for plot 1

NC(62) Independent variable for plot 1

NC(63) Dependent variable for plot 2

NC(64) Independent variable for plot 2

⋮

NC(73) Dependent variable for plot 7

NC(74) Independent variable for plot 7

Note: A comment card must be included with the control cards stating that unit C1 will contain plot output.

Multiple-curve identification. —When plotting on the Orthomat, multiple curves can be identified by either plot symbols or colored ink. (Note: Colors other than black or red do not reproduce very well).

SC 4020 multiple curves are always identified with plot symbols:

Curve No.	Plot Symbol	Ink Color
1	None	Black
2	⊙	Blue
3	▽	Green
4	◇	Purple
5	△	Red
6	⊡	Black

REFERENCES

1. NASA Request for Proposal L-5347 — A Research Study for Supersonic Transport Trajectory Optimization. NASA, Langley Research Center.

2. Bryson, A. E.; and Denham, W. F.: A Steepest-Ascent Method for Solving Optimum Programming Problems. J. Appl. Mech., June 1962.

3. Hague, D. S.: Three-Degree-of-Freedom Problem, Optimization Formulation. Wright Field FDL-TDR-64-1 Part I, Volume 3, October 1964.

4. Leitmann, G.: Optimization Techniques. Academic Press, New York, N.Y., 1962.

5. Bryson, A. E.; Denham, W. F.; and Dreyfus, S. E.: Optimal Programming, Problem with Inequality Constraints I: Necessary Conditions for Extremal Solutions. AIAA Journal, November 1963.

6. Denham, W. F.; and Bryson, A. E.: Optimal Programming Problems with Inequality Constraints II: Solution by Steepest-Ascent. AIAA Journal, January 1964.

7. Phase II-A Comprehensive Report — Sonic Boom and Noise. Boeing Report D6-8686-7, November 1964.

8. Anderson, S. L.: Runge-Kutta Methods of Numerical Integration for Flight Simulation Programs. Boeing Report D2-139527-1, September 1966.

9. Finke, G. F.: Standard Atmospheric Properties (1962). Boeing Applied Mathematics Report AS 1772.

APPENDIX A

CONTROL VARIABLE CHOICE FOR POINT MASS EQUATIONS OF MOTION

The point mass equations of motion used for optimization of trajectories using angle of attack as a control variable are unstable for some types of vehicles with low thrust/weight, high lift/weight airbreathing engines. For rocket-type vehicles or high thrust/weight airbreathing vehicles, stability problems are generally not encountered. Problems arise when flying open loop with an angle-of-attack history defined. Analysis of the linearized point mass equations of motion with angle of attack as the control variable is given herein showing that a lightly damped oscillatory motion results. The use of pitch angle as the control variable is anlyzed; it is shown that the equations of motion are stable and that no oscillation occurs.

The linearized point mass equations of motion, assuming constant weight and flat earth, are:

$$\left.\begin{aligned}
\dot{\bar{v}} - \frac{\left(T_{\bar{v}} - D_{\bar{v}}\right)\bar{v}}{m\,V_o} - \frac{\left(T_{\bar{h}} - D_{\bar{h}}\right)\bar{h}}{m\,V_o} + \frac{g}{V_o}\,\gamma &= -\frac{D_\alpha}{m\,V_o}\,\alpha \\
\gamma - \frac{\left(L_{\bar{v}} - L_o + W\right)\bar{v}}{m\,V_o} - \frac{L_{\bar{h}}}{m\,V_o}\,\bar{h} &= \frac{\left(T_o + L_\alpha\right)}{m\,V_o}\,\alpha \\
\dot{\bar{h}} - \frac{V_o}{h_o}\,\gamma &= 0
\end{aligned}\right\} \qquad \text{(A1)}$$

where

γ = perturbed flight path angle
v = perturbed velocity
h = perturbed altitude
$\bar{v} = v/V_o$
$\bar{h} = h/h_o$
$T_{\bar{v}} = V_o\,\partial T/\partial v$
$D_{\bar{v}} = V_o\,\partial D/\partial v$
$T_{\bar{h}} = h_o\,\partial T/\partial h$
$D_{\bar{h}} = h_o\,\partial D/\partial h$
$D_\alpha = \partial D/\partial\alpha$
$L_{\bar{h}} = h_o\,\partial L/\partial h$
$L_{\bar{v}} = V_o\,\partial L/\partial V$
$L_\alpha = \partial L/\partial\alpha$

All of the state variables listed are perturbation values and $(\quad)_o$ are evaluated at the initial condition. At the initial time the conditions are assumed to be

$$\dot{\bar{V}}_o = \frac{T_o - D_o}{m} - g\gamma_o$$

$$\dot{\gamma}_o = \frac{T_o \alpha_o + L_o}{m V_o} - \frac{W}{m V_o} = 0 \qquad \text{(A2)}$$

$$\dot{\bar{h}}_o = \frac{V_o}{h_o}\gamma_o$$

Taking the Laplace transform of equations A1 and placing in matrix form gives:

$$\begin{bmatrix} S - \frac{T_{\bar{V}} - D_{\bar{V}}}{m V_o} & \frac{g}{V_o} & -\frac{T_{\bar{h}} - D_{\bar{h}}}{m V_o} \\ -\frac{L_{\bar{V}} + W - L_o}{m V_o} & S & -\frac{L_{\bar{h}}}{m V_o} \\ 0 & -\frac{V_o}{h_o} & S \end{bmatrix} \begin{Bmatrix} \bar{V} \\ \bar{\gamma} \\ \bar{h} \end{Bmatrix} = \bar{\alpha} \begin{Bmatrix} \frac{D_\alpha}{m V_o} \\ \frac{T_o + L_\alpha}{m V_o} \\ 0 \end{Bmatrix} \qquad \text{(A3)}$$

The characteristic equation of the system of equations is obtained by equating the determinant of the three by three matrix to zero giving:

$$S^3 - S^2\left(\frac{T_{\bar{V}} - D_{\bar{V}}}{m V_o}\right) - S\left[\frac{V_o}{h_o}\frac{L_{\bar{h}}}{m V_o} - \frac{g}{V_o}\left(\frac{T_{\bar{V}} + W - L_o}{m V_o}\right)\right]$$

$$- \frac{V_o}{h_o}\left(\frac{T_{\bar{h}} - D_{\bar{h}}}{m V_o}\right)\left(\frac{L_{\bar{V}} + W - L_o}{m V_o}\right) + \frac{V_o}{h_o}\frac{L_{\bar{h}}}{m V_o}\left(\frac{T_{\bar{V}} - D_{\bar{V}}}{m V_o}\right) = 0 \qquad \text{(A4)}$$

The characteristic equation evaluated for a typical SST in a cruise condition, assuming $\gamma_o = 0$, $\dot{\bar{V}}_o = 0$, $h_o = 50{,}000$ feet, and $V_o = 3322$ fps

$$T_o = D_o = 96{,}300 \text{ lb}$$

$$W = 346{,}000 \text{ lb}$$

$$L_o = 344{,}000 \text{ lb}$$

$$\alpha_o = 0.0144 \text{ radians}$$

$$L_\alpha = 2.39 \times 10^7 \text{ lb/rad}$$

$$\frac{\partial L}{\partial h} = \frac{L_o}{\rho} \frac{\partial \rho}{\partial h} = - 18.98 \text{ lb/ft}$$

$$\frac{\partial L}{\partial v} = 2 \frac{L_o}{V_o} = 207 \text{ lb - sec/ft}$$

$$\frac{\partial D}{\partial h} = \frac{D_o}{\rho} \frac{\partial \rho}{\partial h} = - 5.32 \text{ lb/ft}$$

$$\frac{\partial D}{\partial v} = 2 \frac{D_o}{V_o} = 57.9 \text{ lb - sec/ft}$$

$$\frac{\partial T}{\partial h} = - 3.18 \text{ lb/ft}$$

$$\frac{\partial T}{\partial u} = 5.3 \text{ lb - sec/ft}$$

gives

$$S^3 + 4.89 \times 10^{-3} S^2 + 1.96 \times 10^{-3} S + 4.79 \times 10^{-6} = 0$$

which has the roots

$$S = - 2.44 \times 10^{-3}$$

$$S = - 1.22 \times 10^{-3} \pm j \, 4.43 \times 10^{-2}$$

These indicate that all roots are stable, but the damping ratio of the oscillatory pair, 0.028, indicates that they represent near neutral stability. The period is 141.6 seconds. Figure A1 shows the results of a digital computer run for similar conditions using the complete nonlinear point mass equation angle-of-attack constant and perturbing the initial value of the flight path angle. The motion is slightly divergent with a period of about 148 seconds. The nonlinearity of the point mass simulation could account for the slight differences as compared to the linear approximation.

The dominant term in the natural frequency for the oscillating mode is $\partial L/\partial h$, or more specifically, the $\partial \rho/\partial h$ (atmospheric density gradient). This term is large for winged vehicles. The important terms in damping are: (1) $\partial T/\partial v$, strictly an airbreather term which is destabilizing; and (2) $\partial D/\partial v$, a stabilizing term — but the desire is to have vehicles with low drag, which makes this term relatively small. For accelerating vehicles, a high thrust/weight helps keep damping large, as can be seen by writing

$$\frac{\partial D}{\partial v} = 2 \frac{D_o}{V_o} = 2 \frac{T_o}{V_o} \left(1 - \frac{\dot{V}_o/g}{T_o/W} \right)$$

A low thrust/weight reduces the damping of the system.

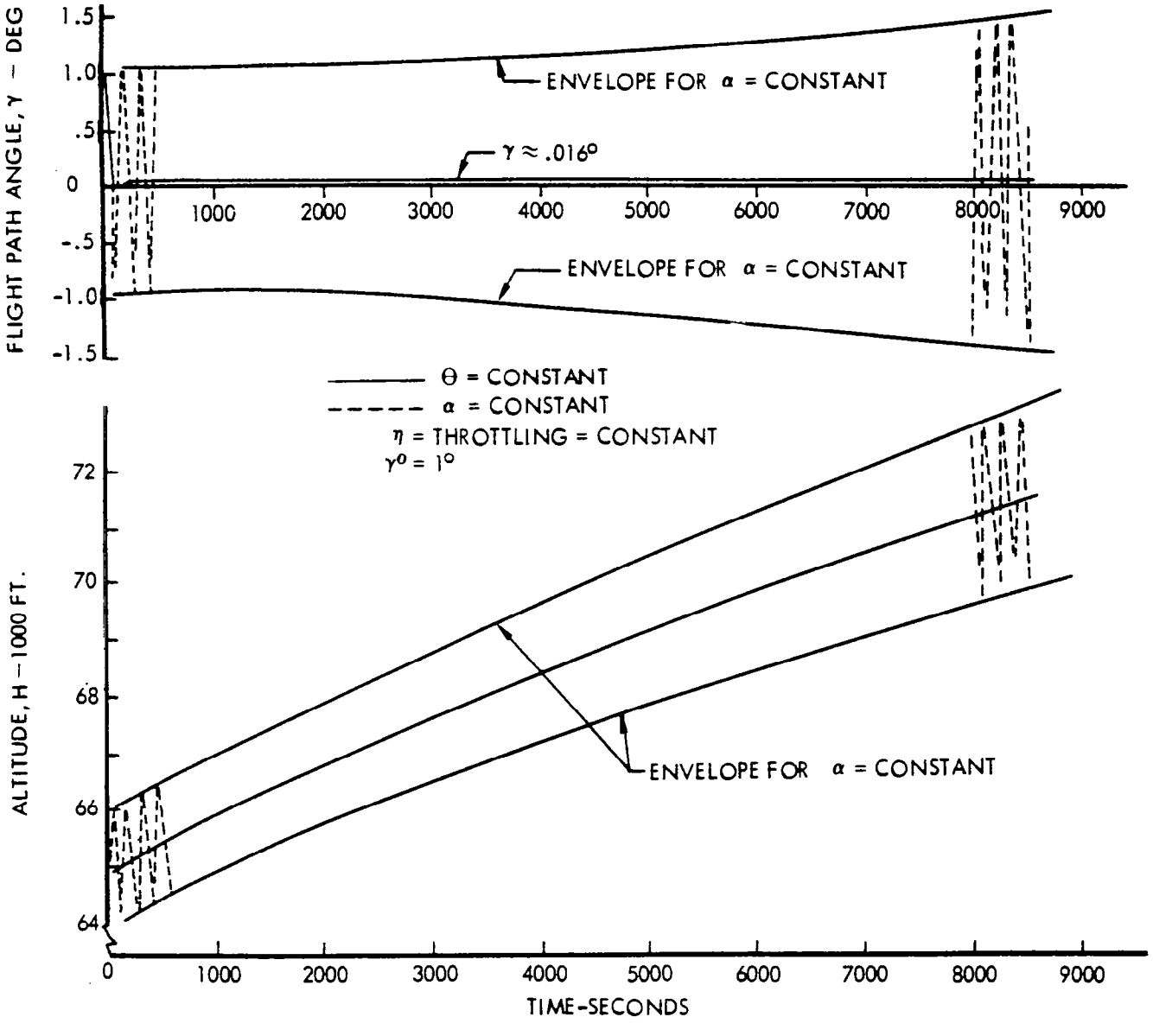


Figure A1 SUPERSONIC CRUISE WITH Θ AND α CONTROL

Introducing θ as the control variable, equations A3 become (upon letting $\bar{\alpha} = \bar{\theta} - \bar{\gamma}$)

$$\begin{bmatrix} S - \frac{T_{\bar{v}} - D_{\bar{v}}}{m V_o} & \frac{g}{V_o} + \frac{D_\alpha}{m V_o} & - \frac{T_{\bar{h}} - D_{\bar{h}}}{m V_o} \\ - \frac{L_{\bar{v}} + W - L_o}{m V_o} & S + \frac{T_o + L_o}{m V_o} & - \frac{L_{\bar{h}}}{m V_o} \\ 0 & - \frac{V_o}{h_o} & S \end{bmatrix} \begin{Bmatrix} \bar{v} \\ \bar{\gamma} \\ \bar{h} \end{Bmatrix} = \begin{Bmatrix} \frac{D_\alpha}{m V_o} \\ \frac{T_o + L_\alpha}{m V_o} \\ 0 \end{Bmatrix} \bar{\theta} \qquad \text{(A5)}$$

which has the characteristic equation

$$S^3 + S^2 \left[- \left(\frac{T_{\bar{v}} - D_{\bar{v}}}{m V_o} + \frac{T_o + L_\alpha}{m V_o} \right) \right] + S \left[- \frac{T_{\bar{v}} - D_{\bar{v}}}{m V_o} \left(\frac{T_o + L_\alpha}{m V_o} \right) - \frac{V_o}{h_o} \frac{L_{\bar{h}}}{m V_o} + \frac{L_{\bar{v}} + W - L_o}{m V_o} \left(\frac{g}{V_o} + \frac{D}{m V_o} \right) \right]$$

$$+ \left[\frac{T_{\bar{v}} - D_{\bar{v}}}{m V_o} + \frac{V_o}{h_o} \frac{L_{\bar{h}}}{m V_o} - \frac{L_{\bar{v}} + W - L_o}{m V_o} \left(\frac{V_o}{h_o} \frac{T_{\bar{h}} - D_{\bar{h}}}{m V_o} \right) \right] = 0 \qquad \text{(A6)}$$

When evaluated for the same flight condition as for the α controller, it is

$$S^3 + 0.45\, S^2 + 4.0 \times 10^{-3}\, S + 4.79 \times 10^{-6} = 0 \qquad \text{(A7)}$$

where the roots are approximately

$$S = 1.2 \times 10^{-3}$$

$$S = - 0.45$$

$$S = - 7.65 \times 10^{-3}$$

Thus each root is real and negative. Figure A1 shows the effect of the θ type control (i.e., θ held constant and the flight path angle perturbed). The motion is overdamped and no oscillation occurs. The damping due to the flight path angle ($T_o + L_\alpha / m V_o$) dominates the dynamics and eliminates the unrealistic oscillation from the cruise flight path.

The use of the θ controller essentially introduces a fourth degree of freedom with the assumptions of neutral stability, no short-period dynamics, and a position feedback autopilot.

APPENDIX B

PROGRAM EQUATIONS, VARIABLES, AND CONSTANTS DEFINED

The purpose of this appendix is to define the names placed in COMMON which describe basic and frequently used equations, variables, and constants.

Equations of Motion

The equations of motion defining the state variables, auxiliary state variables, and placards are defined as F( ). The argument of the functions identify the state variables as given in figure 1.

$$F(K1) = dW/dt = -WDOT$$

$$F(K2) = dH/dt = V_R \sin \gamma_R$$

$$F(K3) = d\gamma_R/dt = \Big\{\Big[(T \sin \alpha + L) \cos \varphi \, G_o/W - G \cos \gamma_R + \omega^2 R \cos \beta \, (\cos \beta \cos \gamma_R + \sin \beta \sin \gamma_R \sin \psi_R)\Big]/V_R + V_R \cos \gamma_R/R + 2 \omega \cos \beta \cos \psi_R \Big\} DEG$$

$$F(K4) = d V_R/dt = (T \cos \alpha - D) G_o/W - G \sin \gamma_R + \omega^2 R \cos \beta \, (\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R)$$

$$F(K5) = d\beta/dt = V_R \cos \gamma_R \sin \psi_R \, DEG/R$$

$$F(K6) = d \psi_R/dt = -\Big[(T \sin \alpha + L) \sin \varphi \, G_o/(W V_R \cos \gamma_R) + V_R \cos \gamma_R \sin \beta \cos \psi_R/(R \cos \beta) + 2 \omega (\sin \beta - \sin \gamma_R \cos \beta \sin \psi_R/\cos \gamma_R) + \omega^2 R \sin \beta \cos \beta \cos \psi_R/(V_R \cos \gamma_R)\Big] DEG$$

$$F(K7) = d\lambda/dt = V_R \cos \gamma_R \cos \psi_R \, DEG/(R \cos \beta)$$

$$F(K8) = dTD/dt = 1$$

$$F(K9) = dRNG/dt = V_R \cos \gamma_R \, R_o/(6076.103R)$$

$$F(K10) = d AHI/dt = Q V_R$$

$$F(K11) = d\Delta V/dt = TG_o/W + \omega^2 R \cos \beta \, (\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R)$$

$$F(K12) = d GL/dt = G \sin \gamma_R$$

$$F(K13) = d DL/dt = DG_o/W$$

$$F(K14) = d T VL/dt = T(1 - \cos \alpha) \, G_o/W$$

F(K15) = $dER/dt = dH/dt + V_R/dt/Go$

F(K16)
F(K17)
F(K18)
F(K19)
F(K20)
} Not defined; available for growth

F(K21) = $d\theta^*/dt = \left(\theta - \theta_{lim}(t)\right)^2$

F(K22) = $d\varphi^*/dt = \left(\varphi - \varphi_{lim}(t)\right)^2$

F(K23) = $d\eta^*/dt = \left((\eta - \eta_{lim}(H, M)\right)^2$

F(K24) = $d\Lambda^*/dt = \left(\Lambda - \Lambda_{lim}(H, M)\right)^2$

F(K25) = $d\alpha^*/dt = \left(\alpha - \alpha_{lim}(H, M)\right)^2$

F(K26) = Not defined, available for growth

F(K27) = $dHD^*/dt = \left(dH/dt - dH/dt_{lim}(H)\right)^2$

F(K28) = $dQt^*/dt = \left(Q - Q_{lim}(t)\right)^2$

F(K29) = $dQM^*/dt = \left(Q - Q_{lim}(M)\right)^2$

F(K30) = $dQ\alpha^*/dt = \left(Q\alpha - Q\alpha_{lim}(M)\right)^2$

F(K31) = $dTEMP^*/dt = \left(TEMP - TEMP_{lim}(t)\right)^2$

F(K32) = $dN^*/dt = \left(N - N_{lim}(H, M)\right)^2$

F(K33) = $dRPA^*/dt = \left(RPA - RPA_{lim}(t)\right)^2$

F(K34) = $dH^*/dt = \left(H - H_{lim}(M)\right)^2$

F(K35) = $d\Delta P^*/dt = \left(\Delta P - \Delta P_{lim}(\lambda, \beta)\right)^2$

F(K36) = $dM^*/dt = \left(M - M_{lim}(H)\right)^2$

F(K37) Not defined; available for growth

F(K38) = $d\gamma^*/dt = \left(\gamma - \gamma_{lim}(H, M)\right)^2$

F(K39)
F(K40)
} Not defined; available for growth

Partial Derivatives

The equations defining the partial derivatives of the equations of motion with respect to the state variables are given as PFX(A,B) where the first argument, A, identifies the equation of motion and the second, B, identifies the derivative state variables. The partial derivative with respect to the control variables are

given as PFU(A,B), where again the first argument identifies the equation of motion and the second identifies the derivative control variable.

$$\text{PFX(K1, K2)} = -\partial\dot{W}/\partial H$$

$$\text{PFX(K1, K3)} = -\partial W/\partial\alpha \;.\; \partial\alpha/\partial\gamma_R$$

$$\text{PFX(K1, K4)} = -\partial\dot{W}/\partial V_R$$

$$\text{PFU(K1, I1)} = -\partial W/\partial\alpha \;.\; \partial\alpha/\partial\theta$$

$$\text{PFU(K1, I3)} = -\partial\dot{W}/\partial\eta$$

$$\text{PFX(K2, K3)} = V_R \cos\gamma_R \cdot \text{RAD}$$

$$\text{PFX(K2, K4)} = \sin\gamma_R$$

$$\text{PFX(K3, K1)} = -(T \sin\alpha + L)\cos\varphi\; G_o\, \text{DEG}/(W^2 V_R)$$

$$\text{PFX(K3, K2)} = \Big\{\Big[(\partial T/\partial H \;\sin\alpha + \partial L/\partial H)\cos\varphi\; G_o/W - \partial G/\partial H \cos\gamma_R + \omega^2 \cos\beta(\cos\beta\cos\gamma_R + \sin\beta\sin\gamma_R \sin\psi_R)\Big]/V_R - V_R\cos\gamma_R/R^2\Big\}\,\text{DEG}$$

$$\text{PFX(K3, K3)} = \Big[(T\cos\alpha + (\partial T/\partial\alpha \;.\; \sin\alpha + \partial L/\partial\alpha)\,\text{DEG}).\; \partial\alpha/\partial\gamma_R \cdot \cos\varphi\; G_o/W + G\sin\gamma_R + \omega^2 R\cos\beta(-\cos\beta\sin\gamma_R + \sin\beta\cos\gamma_R\sin\psi_R)\Big]/V_R - V_R\sin\gamma_R/R$$

$$\text{PFX(K3, K4)} = \Big\{-\Big[(T\sin\alpha + L)\cos\varphi\; G_o/W - G\cos\gamma_R + \omega^2 R\cos\beta(\cos\beta\cos\gamma_R + \sin\beta\sin\gamma_R\sin\psi_R)\Big]/V_R^2 + (\partial T/\partial V_R \sin\alpha + \partial L/\partial V_R)\cos\varphi\; G_o/(W V_R) + \cos\gamma_R/R\Big\}\,\text{DEG}$$

$$\text{PFX(K3, K5)} = \omega^2 R\Big[\cos\beta(-\sin\beta\cos\gamma_R + \cos\beta\sin\gamma_R\sin\psi_R) - \sin\beta(\cos\beta\cos\gamma_R + \sin\beta\sin\gamma_R\sin\psi_R)/V_R - 2\omega\sin\beta\cos\psi_R$$

$$\text{PFX(K3, K6)} = \omega^2 R\cos\beta\sin\beta\sin\gamma_R\cos\psi_R/V_R - 2\omega\cos\beta\sin\psi_R$$

$$\text{PFU(K3, I1)} = (T\cos\alpha + (\partial T/\partial\alpha\sin\alpha + \partial L/\partial\alpha)\,\text{DEG})\cos\varphi \;.\; \partial\alpha/\partial\varphi \cdot G_o/(W V_R)$$

$$\text{PFU(K3, I2)} = -(T\sin\alpha + L)\sin\varphi\; G_o/(W V_R)$$

$$\text{PFU(K3, I3)} = \partial T/\partial\eta\sin\alpha\cos\varphi\; G_o \;.\; \text{DEG}/(W V_R)$$

$$\text{PFU(K3, I4)} = \partial L/\partial\Lambda \;.\; \cos\varphi\; G_o\,\text{DEG}/(W V_R)$$

$$PFX(K4, K1) = -(T\cos\alpha - D)\, G_o/W^2$$

$$PFX(K4, K2) = (\partial T/\partial H \cos\alpha - \partial D/\partial H)\, G_o/W - \partial G/\partial H \cdot \sin\gamma_R + \omega^2 \cos\beta\, (\cos\beta \sin\gamma_R - \sin\beta \cos\gamma_R \sin\psi_R)$$

$$PFX(K4, K3) = (-T \sin\alpha\, RAD + \partial T/\partial\alpha \cos\alpha - \partial D/\partial\alpha)\, \partial\alpha/\partial\gamma_R \cdot G_o/W + \left[-G\cos\gamma_R + \omega^2 R \cos\beta\, (\cos\beta\cos\gamma_R + \sin\beta \sin\gamma_R \sin\psi_R)\right] \cdot RAD$$

$$PFX(K4, K4) = (\partial T/\partial V_R \cos\alpha - \partial D/\partial V_R)\, G_o/W$$

$$PFX(K4, K5) = \omega^2 R\left[\cos\beta\, (-\sin\beta \sin\gamma_R - \cos\beta\cos\gamma_R \sin\psi_R) - \sin\beta\, (\cos\beta \sin\gamma_R - \sin\beta\cos\gamma_R \sin\psi_R)\right] RAD$$

$$PFX(K4, K6) = -\omega^2 R \cos\beta \sin\beta \cos\gamma_R \cos\psi_R \cdot RAD$$

$$PFU(K4, I1) = (\partial T/\partial\alpha \cos\alpha - \partial D/\partial\alpha - T\sin\alpha\, RAD)\, \partial\alpha/\partial\theta \cdot G_o/W$$

$$PFU(K4, I3) = \partial T/\partial\eta \cos\alpha\, G_o/W$$

$$PFU(K4, I4) = -\partial D/\partial\Lambda \cdot G_o/W$$

$$PFX(K5, K2) = -V_R \cos\gamma_R \sin\psi_R \cdot DEG/R^2$$

$$PFX(K5, K3) = -V_R \sin\gamma_R \sin\psi_R/R$$

$$PFX(K5, K4) = \cos\gamma_R \sin\psi_R/R$$

$$PFX(K5, K6) = V_R \cos\gamma_R \cos\psi_R/R$$

$$PFX(K6, K1) = (T\sin\alpha + L) \sin\varphi\, G_o\, DEG/(W^2 V_R \cos\gamma_R)$$

$$PFX(K6, K2) = \left[-(\partial T/\partial H \sin\alpha + \partial L/\partial H) \sin\varphi\, G_o/(W V_R \cos\gamma_R) + V_R \cos\gamma_R \cos\psi_R \sin\beta/(R^2\cos\beta) - \omega^2 \sin\beta\cos\beta\cos\psi_R/(V_R\cos\gamma_R)\right] \cdot DEG$$

$$PFX(K6, K3) = \left\{\left[-(\partial T/\partial\alpha \cdot \sin\alpha + \partial L/\partial\alpha)\, DEG - T\cos\alpha\right] \cdot \partial\alpha/\partial\gamma_R - (T\sin\alpha + L)\sin\gamma_R/\cos\gamma_R\right\} \sin\varphi\, G_o/(W V_R \cos\gamma_R) + (2\sin\psi_R - \omega R \sin\beta\cos\psi_R \sin\gamma_R/V_R) \cdot \omega\cos\beta/\cos^2\gamma_R + V_R \sin\beta \sin\gamma_R \cos\psi_R/(R\cos\beta)$$

$$PFX(K6, K4) = \left\{\left\{\left[T\sin\alpha + V_R(-\partial T/\partial V_R \cdot \sin\alpha - \partial L/\partial V_R)\right] \cdot \sin\varphi\, G_o/W + \omega^2 R \sin\beta\cos\beta\cos\psi_R\right\}/(V_R^2\cos\gamma_R) - \cos\gamma_R\cos\psi_R \sin\beta/(R\cos\beta)\right\} DEG$$

$$PFX(K6, K5) = -V_R\cos\gamma_R\cos\psi_R/(R\cos^2\beta) - 2\omega(\cos\beta + \sin\gamma_R \sin\beta \cdot \sin\psi_R/\cos\gamma_R) - \omega^2 R(\cos^2\beta - \sin^2\beta)\cos\psi_R/(V_R\cos\gamma_R)$$

PFX(K6, K6) $= V_R \cos \gamma_R \sin \psi_R \sin \beta/(R \cos \beta) + 2\ \omega \sin \gamma_R \cdot \cos \beta \cos \psi_R/\cos \gamma_R + \omega^2\ R \sin \beta \cos \beta \sin \psi_R/(V_R \cos \gamma_R)$

PFU(K6, I1) $= \left[- (T \cos \alpha\ RAD + \partial T/\partial \alpha \sin \alpha + \partial L/\partial \alpha)\ \partial \alpha/\partial \theta \cdot \sin \varphi\ G_o/W\ V_R \cos \gamma_R) \right] DEG$

PFU(K6, I2) $= - (T \sin \alpha + L) \cos \varphi\ G_o/(W\ V_R \cos \gamma_R)$

PFU(K6, I3) $= - \partial T/\partial \eta \sin \alpha \sin \psi_R\ G_o\ DEG/(W\ V_R \cos \gamma_R)$

PFU(K6, I4) $= - \partial L/\partial \Lambda \sin \varphi\ G_o\ DEG/(W\ V_R \cos \gamma_R)$

PFX(K7, K2) $= - V_R \cos \gamma_R \cos \psi_R\ DEG/(R^2 \cos \beta)$

PFX(K7, K3) $= - V_R \sin \gamma_R \cos \psi_R\ DEG/(R \cos \beta)$

PFX(K7, K4) $= \cos \gamma_R \cos \psi_R\ DEG/(R \cos \beta)$

PFX(K7, K5) $= V_R \cos \gamma_R \cos \psi_R \sin \beta/(R \cos^2 \beta)$

PFX(K7, K6) $= - V_R \cos \gamma_R \sin \psi_R/(R \cos \beta)$

PFX(K9, K2) $= - V_R \cos \gamma_R\ R_o/(6076.103\ R^2)$

PFX(K9, K3) $= - V_R \sin \gamma_R\ R_o/(6076.103\ R)$

PFX(K9, K4) $= \cos \gamma_R\ R_o/(6076.103\ R)$

PFX(K10, K2) $= V_R\ \partial Q/\partial H$

PFX(K10, K4) $= 3\ Q$

PFX(K11, K1) $= - T\ G_o/W^2$

PFX(K11, K2) $= \partial T/\partial H\ G_o/W + \omega^2 \cos \beta\ (\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R)$

PFX(K11, K3) $= \partial T/\partial \alpha \cdot \partial \alpha/\partial \gamma_R\ G_o/W + \omega^2 R \cos \beta\ (\cos \beta \cos \gamma_R + \sin \beta \sin \gamma_R \sin \psi_R)\ RAD$

PFX(K11, K4) $= \partial T/\partial V_R\ G_o/W$

PFX(K11, K5) $= \omega^2\ R \left[\cos \beta\ (- \sin \beta \sin \gamma_R - \cos \beta \cos \gamma_R \sin \psi_R) - \sin \beta\ (\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R) \right] RAD$

PFX(K11, K6) $= - \omega^2\ R \cos \beta \sin \beta \cos \gamma_R \cos \psi_R\ RAD$

PFU(K11, I1) $= \partial T/\partial \alpha \cdot \partial \alpha/\partial \theta \cdot G_o/W$

PFU(K11, I3) $= \partial T/\partial \eta\ G_o/W$

PFX(K12, K2) $= \partial G/\partial H \sin \gamma_R$

PFX(K12, K3) $= G \cos \gamma_R\ RAD$

$PFX(K13, K1) = - D\ G_o/W^2$

$PFX(K13, K2) = \partial D/\partial H\ G_o/W$

$PFX(K13, K3) = \partial D/\partial \alpha\ \partial \alpha/\partial \gamma_R\ G_o/W$

$PFX(K13, K4) = \partial D/\partial V_R\ G_o/W$

$PFU(K13, I1) = \partial D/\partial \alpha\ \partial \alpha/\partial \theta\ G_o/W$

$PFU(K13, I4) = \partial D/\partial \Lambda\ G_o/W$

$PFX(K14, K1) = - T\ (1 - \cos \alpha)\ G_o/W^2$

$PFX(K14, K2) = \partial T/\partial H\ (1 - \cos \alpha)\ G_o/W$

$PFX(K14, K3) = (\partial T/\partial \alpha\ (1 - \cos \alpha) + T \sin \alpha \text{ RAD})\ \partial \alpha/\partial \gamma_R\ G_o/W$

$PFX(K14, K4) = \partial T/\partial V_R\ (1 - \cos \alpha)\ G_o/W$

$PFU(K14, I1) = \partial T/\partial \alpha\ (1 - \cos \alpha) + T \sin \alpha \text{ RAD})\ \partial \alpha/\partial \theta\ G_o/W$

$PFU(K14, I3) = \partial T/\partial \eta\ (1 - \cos \alpha)\ G_o/W$

$PFX(K15, K1) = V_R G_o PFX(K4, K1)$

$PFX(K15, K2) = V_R G_o PFX(K4, K2)$

$PFX(K15, K3) = PFX(K2, K3) + (V_R G_o PFX(K4, K3)$

$PFX(K15, K4) = PFX(K2, K4) + V_R G_o PFX(K4, K4) + dV_R/dt/Go$

$PFX(K15, K5) = V_R G_o PFX(K4, K5)$

$PFX(K15, K6) = V_R G_o PFX(K4, K6)$

$PFU(K15, I1) = V_R G_o PFU(K4, I1)$

$PFU(K15, I2) = V_R G_o PFU(K4, I2)$

$PFU(K15, I3) = V_R G_o PFU(K4, I3)$

$PFU(K15, I4) = V_R G_o PFU(K4, I4)$

$PFU(K21, I1) = 2(\theta - \theta_{lim})$

$PFU(K22, I2) = 2(\varphi - \varphi_{lim})$

$PFX(K23, K2) = - 2(\eta - \eta_{lim})(\partial \eta_{lim}/\partial H + \partial M/\partial H \cdot \partial \eta_{lim}/\partial M)$

$PFX(K23, K4) = - 2(\eta - \eta_{lim})(\partial \eta_{lim}/\partial M \cdot \partial M/\partial V)$

$PFU(K23, I3) = 2(\eta - \eta_{lim})$

$PFX(K24, K2) = - 2\ (\Lambda - \Lambda_{lim})\ (\partial \Lambda_{lim}/\partial H + \partial \Lambda_{lim}/\partial M\ \partial M/\partial H)$

$PFX(K24, K4) = - 2\ (\Lambda - \Lambda_{lim}) \partial \Lambda_{lim}/\partial M \cdot \partial M/\partial V$

$PFU(K24, I4) = 2(\Lambda - \Lambda_{lim})$

$PFX(K25, K2) = - 2\ (\alpha - \alpha_{lim})\ (\partial \alpha_{lim}/\partial H + \partial M/\partial H \cdot \partial \alpha_{lim}/\partial M)$

$PFX(K25, K3) = 2(\alpha - \alpha_{lim})\ \partial \alpha/\partial \gamma_R$

$$PFX(K25, K4) = -2(\alpha - \alpha_{lim})\,\partial\alpha_{lim}/\partial M \cdot \partial M/\partial V$$

$$PFU(K25, I1) = 2(\alpha - \alpha_{lim})\,\partial\alpha/\partial\theta$$

$$PFX(K27, K2) = -\left[2\,F(K2) - HD_{lim}\right]\partial HD_{lim}/\partial H$$

$$PFX(K27, K3) = 2\left[F(K2) - HD_{lim}\right]PFX(K2, K3)$$

$$PFX(K27, K4) = 2\left[F(K2) - HD_{lim}\right]PFX(K2, K4)$$

$$PFX(K28, K2) = 2(Q - Q_{lim})\,\partial Q/\partial H$$

$$PFX(K28, K4) = 2(Q - Q_{lim})\,\partial Q/\partial V_R$$

$$PFX(K29, K2) = 2(Q - Q_{lim})(\partial Q/\partial H - \partial M/\partial H \cdot \partial Q_{lim}/\partial M)$$

$$PFX(K29, K4) = 2(Q - Q_{lim})(\partial Q/\partial V_R - \partial M/\partial V_R \cdot \partial Q_{lim}/\partial M)$$

$$PFX(K30, K2) = 2(Q\alpha - Q\alpha_{lim})(\alpha\,\partial Q/\partial H - \partial M/\partial H \cdot \partial Q\alpha_{lim}/\partial M)$$

$$PFX(K30, K3) = 2(Q\alpha - Q\alpha_{lim})(Q \cdot \partial\alpha/\partial\gamma_R$$

$$PFX(K30, K4) = 2(Q\alpha - Q\alpha_{lim})(\alpha\,\partial Q/\partial V_R - \partial M/\partial V_R \cdot \partial Q\alpha_{lim}/\partial M)$$

$$PFU(K30, I1) = 2(Q\alpha - Q\alpha_{lim})\,Q\,\partial\alpha/\partial\theta$$

$$PFX(K31, K2) = 2(TEMP - TEMP_{lim})\left[(1 + 0.2\,M^2)\,\partial TEM/\partial H + 0.4\,M \cdot TEM \cdot \partial M/\partial H\right]$$

$$PFX(K31, K4) = 2(TEMP - TEMP_{lim})(0.4\,M \cdot TEM \cdot \partial M/\partial V_R)$$

$$PFX(K32, K1) = -2(N - N_{lim})\,N/W$$

$$PFX(K32, K2) = 2(N - N_{lim})\left[(\partial L/\partial H \cos\alpha + \partial D/\partial H \cdot \sin\alpha)/W - (\partial N_{lim}/\partial H + \partial M/\partial H \cdot \partial N_{lim}/\partial M)\right]$$

$$PFX(K32, K3) = 2(N - N_{lim})\left[(-L\sin\alpha + D\cos\alpha)\,RAD + \partial L/\partial\alpha\cos\alpha + \partial D/\partial\alpha\sin\alpha\right](\partial\alpha/\partial\gamma_R)/W$$

$$PFX(K32, K4) = 2(N - N_{lim})\left[(\partial L/\partial V_R\cos\alpha + \partial D/\partial V_R\sin\alpha)/W - \partial M/\partial V_R \cdot \partial N_{lim}/\partial M\right]$$

$$PFU(K32, I1) = 2(N - N_{lim})\left[(-L\sin\alpha + D\cos\alpha)\,RAD + \partial L/\partial\alpha\cos\alpha + \partial D/\partial\alpha\sin\alpha\right](\partial\alpha/\partial\theta)/W$$

$$PFU(K32, I4) = 2(N - N_{lim})(\partial L/\partial\Lambda\cos\alpha + \partial D/\partial\Lambda\sin\alpha)/W$$

$$PFX(K33, K1) = -2(RPA) - RPA_{lim})\,RPA/W$$

$$PFX(K33, K2) = 2(RPA - RPA_{lim})\,G_o^2/(W^2\,RPA)\left[(L + T\sin\alpha) \cdot (\partial L/\partial H + \partial T/\partial H \cdot \sin\alpha) + (D - T\cos\alpha)(\partial D/\partial H - \partial T/\partial H \cdot \cos\alpha)\right]$$

$$PFX(K33, K3) = 2 (RPA - RPA_{lim}) G_o^2/(W^2 RPA)\Big[(L + T \sin \alpha) (\partial L/\partial \alpha + \partial T/\partial \alpha \sin \alpha + T \cos \alpha \cdot RAD) + (D - T \cos \alpha) (\partial D/\partial \alpha - \partial T/\partial \alpha \cos \alpha + T \sin \alpha \cdot RAD)\Big] \partial \alpha/\partial \gamma_R$$

$$PFX(K33, K4) = 2 (RPA - RPA_{lim}) G_o^2/(W^2 RPA)\Big[(L + T \sin \alpha) (\partial L/\partial V_R + \partial T/\partial V_R \cdot \sin \alpha) + (D - T \cos \alpha) (\partial D/\partial V_R - \partial T/ V_R \cdot \cos \alpha)\Big]$$

$$PFU(K33, I1) = 2 (RPA - RPA_{lim}) G_o^2/(W^2 RPA)\Big[(L + T \sin \alpha) (\partial L/\partial \alpha + \partial T/\partial \alpha \sin \alpha + T \cos \alpha \cdot RAD) + (D - T \cos \alpha) (\partial D/\partial \alpha - \partial T/\partial \alpha \cos \alpha + T \sin \alpha \cdot RAD)\Big] \partial \alpha/\partial \theta$$

$$PFU(K33, I3) = 2 (RPA - RPA_{lim}) G_o^2/(W^2 RPA)\Big[(L + T \sin \alpha) \partial T/\partial \eta \cdot \sin \alpha - (D - T \cos \alpha) \partial T/\partial \eta \cdot \cos \alpha\Big]$$

$$PFU(K33, I4) = 2 (RPA_{lim}) G_o^2/(W^2 RPA)\Big[(L + T \sin \alpha) \cdot \partial L/\partial \Lambda + (D - T \cos \alpha) \partial D/\partial \Lambda\Big]$$

$$PFX(K34, K2) = 2 (H - H_{lim}) (1 - \partial M/\partial H \cdot \partial H_{lim}/\partial M)$$

$$PFX(K34, K4) = 2 (H - H_{lim}) (\partial M/\partial V_R \cdot \partial H_{lim}/\partial M)$$

$$PFX(K35, K2) = \Big[F(K35)_{H + 100} - F(K35)_H\Big]/100$$

$$PFX(K35, K3) = \Big[F(K35)_{\alpha + 0.1} - F(K35)_{\alpha}\Big](\partial \alpha/\partial \gamma_R)/0.1$$

$$PFX(K35, K4) = \Big[F(K35)_{V_R + 10} - F(K35)_{V_R}\Big]/10$$

$$PFX(K35, K5) = \Big[F(K35)_{\beta + 0.1} - F(K35)_{\beta}\Big]/0.1$$

$$PFX(K35, K6) = \Big[F(K35)_{\psi + 0.1} - F(K35)_{\psi}\Big]/0.1$$

$$PFX(K35, K7) = \Big[F(K35)_{\lambda + 0.1} - F(K35)_{\lambda}\Big]/0.1$$

$$PFU(K35, I1) = \Big[F(K35)_{\alpha + 0.1} - F(K35)_{\alpha}\Big] (\partial \alpha/\partial \theta)/0.1$$

$$PFU(K35, I4) = \Big[F(K35)_{\Lambda + 0.1} - F(K35)_{\Lambda}\Big]/0.1$$

$$PFX(K36, K2) = 2 (M - M_{lim}) (\partial M/\partial H - \partial M_{lim}/\partial H)$$

$$PFX(K36, K4) = 2 (M - M_{lim}) \partial M/\partial V_R$$

$$PFX(K38, K2) = -2 (\gamma_R - \gamma_{R_{lim}}) (\partial \gamma_{R_{lim}}/\partial H + \partial \gamma_{R_{lim}}/\partial M \, \partial M/\partial H)$$

$$PFX(K38, K3) = 2 (\gamma_R - \gamma_{R_{lim}})$$

$$PFX(K38, K4) = -2 (\gamma_R - \gamma_{R_{lim}}) \partial \gamma_{R_{lim}}/\partial M \, \partial M/\partial V_R$$

Auxiliary Printout Variables

The auxiliary printout variables, AK( ), are functions that are desired for printout in addition to the state and control variables. The argument identifies the variable as given in figure 13. State variables and printout variables in metric units are also defined as AK's.

AK(1) $= T = (\eta\, \text{Tref}\, \text{XKT} - \text{PAe}/144) \cos \delta_c$

AK(2) $= L = C_L Q S_{ref}$

AK(3) $= D = C_D Q S_{ref}$

AK(4) $= Q = 0.5 \rho V_R^2$

AK(5) $= M = V_R / V_A$

AK(6) $= \text{GCR} = \cos^{-1}\left[\sin \beta_o \sin \beta + \cos \beta_o \cos \beta \cos (\lambda - \lambda_o)\right]$ 60 DEG

AK(7) $= \text{WDOT} = T_{ref}\, \text{XKM}/I_{S_{ref}} + \text{WDOT}_{inert}$

AK(8) $= V_I = \left[(R\omega \cos \beta + V_R \cos \gamma_R \cos \psi_R)^2 + \sin^2 \gamma_R + (V_R \cos \gamma_R \cos \psi_R)^2 \right]^{1/2}$

AK(9) $= \gamma_I = \sin^{-1}\left[V_R \sin \gamma_R / V_I\right]$ DEG

AK(10) $= \psi_I = \sin^{-1}\left[V_R \cos \gamma_R \sin \psi_R / (V_I \sin \gamma_I)\right]$ DEG

AK(11) $= \alpha = \theta - \gamma_R$

AK(12) $= \text{TEMP} = \text{TEM}\,(1 + 0.2M^2)$

AK(13) $= V_A$

AK(14) $= P$

AK(15) $= \rho$

AK(16) $= \text{TEM}$

} From ATMOS Routine

AK(17) $= W^M = 4.448221615\ \text{XK1}$

AK(18) $= H^M = 0.3048\ \text{X}\ (\text{K2})$

AK(19) $= \text{RPA} = \left[(L + T \sin \alpha)^2 + (D - T \cos \alpha)^2\right]^{1/2} G_o / W$

AK(20) $= N = (L \cos \alpha + D \sin \alpha)/W$

AK(21) $= Q\alpha = Q\alpha$

AK(22) $= G = \mu / R^2$

AK(23) $= R = H + R_o$

AK(24) $= \Delta P = \Delta PJ\ (P_o P)^{1/2}\ K_R\ (M^2 - 1)^{1/8} / H^{3/4}$

AK(25) $= C_L$ From tables 33 and 34

AK(26) $= C_D$ From tables 23, 33, and 34

AK(27) $= L/D = L/D$

AK(28) $= SFC = 3600/I_{S_{ref}}$

AK(29) $= V_R^M = 0.3048$ X (K4)

AK(30) $= T^M = 4.448221615$ AK(1)

AK(31) $= L^M = 4.448221615$ AK(2)

AK(32) $= D^M = 4.448221615$ AK(3)

AK(33) $= Q^M = 47.8802589$ AK(4)

AK(34) $= WDOT^M = 0.45387006$ AK(7)

AK(35) $= V_I^M = 0.3048$ AK(8)

AK(36) $= Q\alpha^M = 0.8356681645$ AK(21)

AK(37) $= \Delta P^M = 47.8802589$ AK(24)

AK(38) $= SFC^M = 367.322574$ AK(28)

AK(39)
AK(40) } Not defined; available for growth

VAR Array

Many variables, functions, and terms are placed in a COMMON array called VAR( ) because of convenience or frequent usage. This array allows for the addition of variables to the program without changing the COMMON block.

VAR(1) $= T_{ref}$

VAR(2) $= A_e$

VAR(3) $= I_{S_{ref}}$

} From Tables 31 and 32

VAR(4) = $\partial V_a / \partial H$

VAR(5) = $\partial P / \partial H$

VAR(6) = $\partial \rho / \partial H$

VAR(7) = $\partial TEM / \partial H$

} From numerical differentiation of ATMOS Routine

VAR(8) = Initial trajectory time (TSTART)

VAR(9) = ΔPJ From Table 30

VAR(10)

VAR(11)

VAR(12)

} Not defined; available for growth

VAR(13) = $C_{D_f} = (H - H_{ref})\ \partial C_{D_f} / \partial H$ Where H_{ref} and $\partial C_{D_f} / \partial H$ are from Table 23

VAR(14) = $\partial C_D / \partial H$

VAR(15) = $\partial C_D / \partial V_R$

VAR(16) = $\partial C_D / \partial \alpha$

VAR(17) = $\partial C_D / \partial \Lambda$

VAR(18) = $\partial C_L / \partial H$

VAR(19) = $\partial C_L / \partial V_R$

VAR(20) = $\partial C_L / \partial \alpha$

VAR(21) = $\partial C_L / \partial \Lambda$

VAR(22) = $(\cos \beta \cos \gamma_R + \sin \beta \sin \gamma_R \sin \psi_R)$

VAR(23) = $(\cos \beta \sin \gamma_R - \sin \beta \cos \gamma_R \sin \psi_R)$

VAR(24) = $Vcs_R = \left[(R_o \omega \cos \beta - Vcs_I \cos \psi_I)^2 + (Vcs_I \sin \psi_I)^2 \right]^{1/2}$

VAR(25) Not defined; available for growth

VAR(26) = W_f Last-stage final weight

VAR(27) = W_I Last-stage initial weight

VAR(28) = $\Delta \gamma_{tilt}$

VAR(29) $= \gamma_o$

VAR(30) = MLSP (Maximum last-stage propellant, input)

VAR(31) $= \partial T_{ref} / \partial H$

VAR(32) $= \partial T_{ref} / \partial V_R$

VAR(33) $= \partial T_{ref} / \partial \alpha$

VAR(34) $= \partial T_{ref} / \partial \eta$ (= 0 for current thrust options)

VAR(35) $= \partial WDOT / \partial H$

VAR(36) $= \partial WDOT / \partial V_R$

VAR(37) $= \partial WDOT / \partial \alpha$

VAR(38) $= \partial WDOT / \partial \eta$

VAR(39) $= \beta_G = \beta + (x \sin \psi_R + y \cos \psi_R) / R_o$

VAR(40) $= \lambda_G = \lambda + (x \cos \psi_R - y \sin \psi_R)/(R_o \cos \beta)$

VAR(41) $=$ PAPT $= \partial \alpha / \partial \theta = 1$

VAR(42) $=$ PAPG $= \partial \alpha / \partial \gamma_R = -1$

VAR(43) $=$ PAPF $= \partial \alpha / \partial \phi = 0$

Note: VAR(44) through VAR(50) not defined at present

CONST Array

A number of constants or functions of input constants are in the COMMON array identified as CONST( ). Again the array allows adding constants to COMMON without modifying the common block.

CONST(1) $= \sin \beta_0$

CONST(2) $= \cos \beta_0$

CONST(3) $= \cos \delta_{c1}$

CONST(4) $= \cos \delta_{c2}$

CONST(5) $= \cos \delta_{c3}$

CONST(6) $= \cos \delta_{c4}$

CONST(7) $= \cos \delta_{c5}$

CONST(8) $= \cos \delta_{c6}$

CONST(9) $= \cos \delta_{c7}$

CONST(10) $= \cos \delta_{c8}$

CONST(11) $= \cos \delta_{c9}$

CONST(12) $= \cos \delta_{c10}$

CONST(13) $= \cos \delta_{c11}$

CONST(14) $= \cos \delta_{c12}$

CONST(15) $= \cos \delta_{c13}$

CONST(16) $= \cos \delta_{c14}$

CONST(17) $= 6076.103$

CONST(18) $= K_R = 1.9$

CONST(19) $= P_o =$ (Sea level pressure)

CONST(20) $= R_D =$ (Desired final radius)

CONST(21) $= V_{csI} = \left[\mu / R_D \right]^{1/2}$

CONST(22) $= V_{Ao} =$ (Sea level speed of sound)

CONST(23) $= L =$ (Boom PF reference length)

Note: CONST(24) through CONST(30) not defined at present

Miscellaneous COMMON

Words in COMMON or in calling sequences that are not defined by an array are listed below. These are primarily defined for convenience for the programmer.

PWPH $= \partial \dot{W} / \partial H =$ VAR(35)

PWPV $= \partial \dot{W} / \partial V_R =$ VAR(36)

PWPA = $\partial \dot{W}/\partial \alpha$ = VAR(37)

PWPT = $\partial \dot{W}/\partial \eta$ = VAR(38)

PTPH = $\partial T/\partial H = (\eta \cdot XKT \cdot \partial T_{ref}/\partial H - A_e \partial P/\partial H/144) \cos \delta_c$

PTPV = $\partial T/\partial V_R = \eta \cdot XKT \cdot \partial T_{ref}/\partial V_R \cdot \cos \delta_c$

PTPA = $\partial T/\partial \alpha = \eta \cdot XKT \cdot \partial T_{ref}/\partial \alpha \cdot \cos \delta_c$

PTPT = $\partial T/\partial \eta = XKT \cdot T_{ref} \cdot \cos \delta_c$

PQSPH = $\partial Q/\partial H \cdot S_{ref} = PQPH \cdot S_{ref}$

PQSPV = $\partial Q/\partial V_R \cdot S_{ref} = PQPV \cdot S_{ref}$

PLPH = $\partial L/\partial H = \partial C_L/\partial H \cdot QS + C_L \cdot PQSPH$

PLPV = $\partial L/\partial V_R = \partial C_L/\partial V_R \cdot QS + C_L \cdot PQSPV$

PLPA = $\partial L/\partial \alpha = \partial C_L/\partial \alpha \cdot QS$

PLPS = $\partial L/\partial \Lambda = \partial C_L/\partial \Lambda \cdot QS$

PDPH = $\partial D/\partial H = \partial C_D/\partial H \cdot QS + C_D \cdot PQSPH$

PDPV = $\partial D/\partial V_R = \partial C_D/\partial V_R \cdot QS + C_D \cdot PQSPH$

PDPA = $\partial D/\partial \alpha = \partial C_D/\partial \alpha \cdot QS$

PDPS = $\partial D/\partial \Lambda = \partial C_D/\partial \Lambda \cdot QS$

PMPH = $(\partial M/\partial H)_{V_R} = -V_R/V_A^2$

PMPV = $(\partial M/\partial V_R)_H = 1/V_A$

PQPH = $(\partial Q/\partial H)_{V_R} = 0.5 V_R^2 \cdot \partial \rho/\partial H$

PQPV = $(\partial Q/\partial V)_H = \rho V_R$

RO = R_o = Earth Radius = 20,902,992 ft

DEG = Degrees/Radian

RAD = 1/DEG

CI1 = cos θ

SI1 = sin θ

CI2 = cos φ

CK6 = $\cos \psi_R$

SI2 = $\sin \varphi$

CK3 = $\cos \gamma_R$

SK3 = $\sin \gamma_R$

CK5 = $\cos \beta$

SK5 = $\sin \beta$

SK6 = $\sin \psi_R$

CAK11 = $\cos \alpha$

SAK11 = $\sin \alpha$

XK1 = W

UI1 = θ

UI2 = φ

UI3 = η

UI4 = Λ

AK11 = α

GO = G_o = Mass-To-Weight Conversion, 32.174

GR = μ = Gravitional Constant, 14.081718×10^{15}

OMEGA = ω= Earth Rotation Rate, $0.72921152 \times 10^{-4}$

PGPH = $\partial G/\partial H = -2\,G/R$

Internal Program Indicators

The following indicators (L array) are used for controlling internal operations in the program.

L(1) Number of integration points stored during integration of last valid trajectory (Set in subroutine LAMBDA). Used in the recovery of the control variable history.

L(2) Subroutine CØNTR lookup memory index during forward integration and subroutine DVAL2 lookup memory index during backward integration.

L(3) Maximum travel indicator

= 0 No constraints traveled more than maximum allowable.

= 1 One or more of the constraints traveled more than maximum allowable.

L(4) Minimum DUSQ indicator

= 0 DUSQ > minimum

= 1 DUSQ = minimum

L(5) Subroutine DVAL2 lookup control

= 0 lookup if T(time) not equal to previous time.

= 1 lookup

L(6) Current iteration phase

= -1 valid step, nominal only (set when NARBY = 0)

= 0 trial

= 1 valid step

= 2 reverse integration

= 10 end of run

L(7) Not used

L(8) Forward trajectory last integration step indicator

= 0 Intermediate integration step

= 1 Last integration step

L(9) Plot indicator

= 0 Don't plot current iteration

= 1 Plot current iteration

L(10) Gamma tilt variable initial condition indicator

= 0 Gamma tilt not selected as variable initial condition

= 1 Gamma tilt selected as variable initial condition

L(11) Current number of records of partial derivatives remaining on KPAR (partial) unit during reverse integration.

L(12) Half-step indicator

= 0 Not a half step

= 1 Half step - do not print

L(13) Iteration phase indicator

= 0 Next forward trajectory will be a valid step

= 1 Next forward trajectory will be a trial

Secondary use

Valid step rejected indicator

= 0 Valid step not rejected

= 1 Valid step rejected

L(14) Numerical partial check indicator

(Incremented by 1 every stored integration step and reset to 0 whenever numerical partials are computed)

= NC(12) Compute numerical partials

≠ NC(12) Don't compute numerical partials

APPENDIX C

PROGRAM CONTROL LOGIC

The following tests made after different phases of an iteration are in no way connected with the steepest-ascent logic. They are present only to ensure the convergence to a solution in a reasonable length of time. All of the tests are strictly empirical and all have at some time or another played an important part in the convergence of particular cases. Experience is the only answer that can be given to the question as to the need of some of the tests. The control logic as presented in its current form is by no means able to solve all problems with which it is confronted. The control logic has, however, successfully caused the convergence of a great number of cases ranging from high-speed reentry vehicles to supersonic transports, thus showing its flexibility and capability.

Accept or reject valid step tests. — Majority vote test: The constraints outside their respective tolerance bands on the previous valid step are examined with respect to their travel on this valid step. The constraints with adverse travel are assigned a value of -1; the constraints with favorable travel are assigned a value of +1. The algebraic sum of these assigned values is then taken. If the value of DUSQ is large enough to allow optimization, the performance function is treated in a similar way and added to the sum. The direction of change in performance is not meaningful when the program is not optimizing and therefore is not included in the majority vote. The change in performance is just the result of the constraint motion in this case. If the majority vote (algebraic sum) is negative, the valid step just calculated will be rejected and another valid step will later be attempted. If the majority vote is either zero or positive, the valid step will normally be accepted and another iteration will be run if desired. It should be noted that the majority vote is neither computed nor tested after the first valid step (nominal or first trajectory of a restart) of a run.

Excessive adverse performance test: An allowable adverse change in performance is computed as the larger of ten percent of the value of performance on the previous valid step, and five percent of the greatest value of performance attained on any previous trial or valid step (including the current valid step). If the change in performance is in the wrong direction and it is greater than the allowable adverse change, a step-size coefficient based on adverse performance is computed and the valid step is rejected. The nominal or first trajectory of a restart cannot be rejected because of an adverse performance change.

It should be noted that, in the above two tests, the valid step will not be rejected if the decision to attempt the valid step was made because too many trials were made or DUSQ is at its minimum value. A valid step is run after a specified maximum number of trials or when DUSQ is set to the minimum allowable value to obtain a new set of influence coefficients, and is run under the assumption that many of the constraints and possibly performance will travel in an unfavorable direction.

Abort test after a valid step. — Exit on too many valid step rejections: Under certain conditions the program control logic cannot make a successful valid step and will continue to reject consecutive valid steps. If the number of trials taken before the attempted valid step plus the number of rejected valid steps ever exceeds 10, the case being run is aborted with an appropriate message.

Abort tests following reverse integration. — Matrix inversion: If the inverse of $I_{\psi\psi}$ matrix cannot be accomplished because of overflow during matrix inversion or the matrix is singular, an appropriate message is printed and the run is aborted.

Check for optimum: If all of the constraints are within their respective input tolerance bands, the change in performance is less than or equal to 10^{-6} times the value of performance on the previous valid step, and the denominator of the first Lagrange multiplier is less than or equal to one ten thousandth of $I_{\varphi\varphi}$, the case is terminated and a message stating that the last valid step is an optimum is printed.

Detect convergence failure: If all of the constraint changes are less than their respective input tolerances and the change in performance is less than or equal to 10^{-6} times the value of performance on the previous valid step, the case is terminated and a message stating convergence failure is printed. This indicates that the constraints are not within the prescribed band and that improvement in performance and constraints cannot be made satisfactorily.

Gradient of φ is too negative test: If the denominator of the first Lagrange multiplier plus ten percent of $I_{\varphi\varphi}$ is less than zero, the case is terminated and a message stating that the gradient of φ is too negative to continue is printed.

Tests following a trial causing another trial to be made. — Majority vote test: This test is similar to the majority vote test previously mentioned, except in two areas. The constraints being examined are the ones outside of their respective temporary tolerance bands on the current valid step and if all of the constraints being counted improved, an indicator is set. If the majority vote is negative, another trial will almost always be made (only other test examined after a negative majority vote is the bounce test, which is described later). A zero or positive majority vote will cause the step-size coefficients to be calculated.

Excessive adverse performance test: This test is identical to the adverse performance test previously described. If there is excessive adverse performance on the current trial, another trial is almost always run. (Only other test examined after excessive adverse performance is detected is the bounce test.)

Tests after a trial that will cause a valid step to be run. — End point search: If the step-size coefficient equals two and the control logic is in an end point search mode (i.e., $dU^2 < d\bar{\beta}' I_{\psi\psi}^{-1} d\bar{\beta}$), the current trial will be accepted and a valid step will be run. An appropriate message will also be printed.

Penalty function test: The penalty functions are examined (if any). If one or more have been violated after having no value for at least one iteration, a valid step is run — provided that the step-size coefficient is greater than or equal to one. An appropriate message is printed.

Performance test (no constraints): If at least one trial has already been run and the case has no constraints, the performance change is examined. If the performance change is favorable and the step-size coefficient is equal to one half, the current trial is accepted and a valid step is run. An appropriate message is printed.

All constraints favorable test: If all of the constraints counted in the majority vote test improved, the step-size coefficient was not reduced due to a constraint change that was larger than its maximum permissible travel, and the step size (DUSQ) is reduced over the previous step size on the last valid step, then the current trial is accepted and a valid step is run. An appropriate message is printed.

Bounce test: If the trend after a set of trials following a reverse integration is either a monotonically increasing or decreasing step size, and on the current trial, the trend is reversed, the last consistent trial is accepted and a valid step is run. An appropriate message is printed.

Too many trials: A valid step will be run if the following conditions are met: six or more trials have been run since the last accepted valid step, the changes in constraints being asked for are not too great ($d\bar{\psi} < d\bar{\psi}_{max}$), and the control logic is not in an end point search ($dU^2 > d\bar{\beta}' I_{\psi\psi}^{-1} d\bar{\beta}$). An appropriate message is printed.

Too many trials: This test has no other conditions on it except that if eight trials have been run, the current trial is accepted and a valid step is run.

The above two tests (too many trials) are not satisfied under normal running conditions. The program normally runs one or two trials before making a valid step.

Factors affecting the selection of step size coefficient. — The step-size coefficient calculation will greatly affect the decision of the control logic in deciding to run another trial or valid step. This calculation is the dominant factor in deciding between a trial and a valid step.

A step-size coefficient based on linearity is computed for each constraint and performance. The maximum step-size coefficients are then calculated for each constraint using its respective maximum permissible constraint change in the parabolic approximation. The largest step-size coefficient based on constraint and performance linearities is then selected. The control logic is then said to be controlling with the function corresponding to the largest step-size coefficient. The largest

step-size coefficient is compared with the maximum step-size coefficients of the constraints outside their respective temporary tolerance bands. If the smallest value of the maximum step-size coefficients being examined is less than the largest step-size coefficient, then it is chosen as the new step-size coefficient. If at this point an indicator shows that the step-size coefficient calculation has not been repeated (explained later) and the step-size coefficient is less than 0.5, the control logic will return from this calculation with the step-size coefficient set equal to 0.5. This will cause the running of another trial. If the above conditions are not met, tests are made to determine if the step-size calculation should be repeated. Conditions under which the calculations can be repeated are: the logic is controlling with a constraint that is within its temporary tolerance band, and the maximum step-size coefficient corresponding to this constraint is less than the current step-size coefficient.

The maximum permissible travel of the controlling constraint is checked to ensure that it is not greater than its error. If the travel is less than the error, the logic returns from the calculation without recalculating the step-size coefficient. If the travel, however, is greater than the error, the step-size calculation is repeated. Repeating the calculation means that the next largest step-size coefficient is chosen (based on linearities) and the logic is then said to be controlling with this function (another constraint or performance). The logic then follows the same path as described above.

It is possible to repeat the calculation many times before choosing a final step-size coefficient.

The final step-size coefficient is then compared with the limits 0.5 and 2. If the step-size coefficient lies outside the limits, it is set to the closest limit. A final step-size coefficient of 0.5 or 2 will probably cause the running of another trial, whereas a step-size coefficient within the limits will cause a valid step to be attempted.

Minimum allowable value of DUSQ. — The value of DUSQ required for $d\overline{\psi}$ is calculated after a valid step as $d\overline{\psi}' \times I_{\psi\psi}^{-1} \times d\overline{\psi}$. 0.25 percent of this value is the minimum allowable value of DUSQ. After a trial, reverse integration, or rejected valid step, the value of DUSQ is compared with the minimum allowable. If the current value of DUSQ is less than the minimum allowable, DUSQ is reset to the minimum allowable, recalculations are made with the new value of DUSQ, and a valid step is run. The valid step under this mode of operation cannot be rejected. It should be noted that no trials are run while the program is in the minimum DUSQ mode.

A need for a minimum allowable DUSQ resulted from a number of difficult cases, which reduced the value of DUSQ to such a point that any changes in constraint or performance values were due strictly to computer round-off error and noise. Control logic predictions under these conditions were not reliable and therefore caused an even further reduction in the value of DUSQ. It is obvious

that once this mode of operation is entered it is extremely difficult for the program to break out of it. The minimum DUSQ concept successfully bypasses this problem, yet causes another. It is possible for the program to cause the divergence of a case when it is in the minimum DUSQ mode because the control logic is being forced to use a value of DUSQ greater than it would otherwise choose. Experience has shown that difficult cases will run in the minimum DUSQ mode for a number of iterations and break out of it and run in the normal program mode. Examining these cases in detail has shown that a very slight amount of divergence has occurred during the iterations run in the minimum DUSQ mode. This occurred at a point when the program found it extremely difficult to shape the flight path properly. Once this was overcome the program stepped out of this mode and successfully optimized.

Minimum DUSQ for optimization. — The value of the minimum DUSQ for optimization is computed as (2.1-FXTRA1)*T1 where FXTRA1 is the nondimensional amount of the constraint error to be eliminated in this iteration (increased by five percent every iteration) and T1 is the amount of DUSQ required to meet the current $\Delta\beta$.

After a trial, reverse integration, or rejected valid step, the current value of DUSQ is compared with the minimum DUSQ for optimization. If the current value is less than the minimum value, the value of DUSQ is reset to this minimum value and recalculations are made with the new value of DUSQ. The control logic continues with no other change. At the start of a case the minimum DUSQ required for optimization is more than twice as much as that needed to meet the amount of constraint error being eliminated. This essentially forces the program to optimize from the beginning of a run. Without this logic, too many cases used the entire DUSQ to meet the constraints and thus never optimized. Once the constraints were met, the program would then turn to the problem of optimization, which resulted in the convergence to a local optimum. The current logic will cause the program to optimize first and then meet the constraints.

Scaling of $d\bar{\beta}$. — If the denominator of the first Lagrange multiplier minus $10^{-8} \times I_{\varphi\varphi}$ is less than or equal to zero (1st test) and the numerator of the first Lagrange multiplier is less than zero (2nd test), DBETA ($d\bar{\beta}$), T1, DELX, XLAMDX, and DPSIP are scaled by the factor $(DUSQ/T1)^{1/2}$. If the first test above is satisfied, the program is said to be in the end point search mode. If the second test above is not satisfied, scaling does not occur and DUSQ is set equal to T1. If the first test above is not satisfied and the numerator of the first Lagrange multiplier is negative, scaling does occur and the program is said to be asking for the constraints too quickly. The program will also operate in the end point search mode when the user selects the boundary value mode.

Nondimensional ψ change. — The nondimensional amount of constraint error to be eliminated for the new iteration is computed after every valid step. The amount of constraint error asked for is normally increased by five percent each iteration. If, however, the program is not in its optimization mode and a

constraint is inside its tolerance band, its respective value of nondimensional ψ change is not increased every iteration and is set to 10 percent. This causes the program to pay more attention to constraints outside their respective tolerance bands when in an end point search mode, minimum DUSQ mode, etc. The nondimensional ψ change for a constraint is reset once it is outside its tolerance band. It should be noted that, only after a minimum of 20 iterations, will a full 100 percent of the constraint error be asked for.

Temporary tolerance bands. — Initial values of the temporary bands are equal to 10 times the final tolerance bands (input by the user). The constraints are examined after a valid step to check if they are inside their respective temporary tolerance bands. If all of the constraints are inside their respective temporary tolerance bands, all of the tolerances are reduced to one half their value. The only exception to this is that a temporary tolerance band cannot be less than its final input tolerance band.

Nondimensional permissible constraint change. — Two nondimensional permissible constraint changes are computed after a valid step for each constraint. One is for motion toward the desired constraint value and one is away from the desired constraint value. The values are a function of the nondimensional constraint error asked for (CPSI). As successive iterations are run, the nondimensional permissible constraint changes are decreased. The two formulas are:

$$\text{PSIFWD (forward motion)} = 5.0 - 3.0 \times \text{CPSI}$$

$$\text{PSIBWD (backward motion)} = 1.05 - \text{CPSI}$$

Tolerance band indicators. — The number of consecutive iterations that a constraint has been outside its respective tolerance band is accumulated and stored in the computer word INDSIC. This value is reset to zero whenever a constraint is inside its tolerance band.