NASA CR-132689

MCR-75-143 – Vol-1
NAS1-13611

Volume I

Final
Report

April 1975

PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES (POST).

Volume 1: Formulation Manual

(NASA-CR-132689) PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES (POST). VOLUME 1: FORMULATION MANUAL Final Report (Martin Marietta Corp.) 148 p HC $5.75 CSCL 22A

N75-32146

Unclas
G3/13 40814

Prepared by

G. L. Brauer, D. E. Cornick, A. R. Habeger, F. M. Petersen, and R. Stevenson

Approved

DE Cornick

D. E. Cornick
Program Manager

MARTIN MARIETTA CORPORATION
P.O. Box 179
Denver, Colorado 80201

NASA CR-132689

MCR-75-143
NAS1-13611

Volu ac I

Final Report

April 1975

PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES (POST)

Volume I - Formulation Manual

Prepared by

G. L. Brauer, D. E. Cornick, A. R. Habeger, F. M. Petersen, and R. Stevenson

Approved

[illegible]

D. E. Corn'ck
Program Manager

MARTIN MARIETTA CORPORATION
P.O. Box 179
Denver, Colorado 80201

FOREWORD

This final report describing the formulation of the Program to Optimize Simulated Trajectories (POST) is provided in accordance with Part IV of NASA Contract NAS1-13611. The report is presented in three volumes as follows:

Volume I - Program to Optimize Simulated Trajectories - Formulation Manual;

Volum II - Program to Optimize Simulated Trajectories - Utilization Manual;

Volume III - Program to Optimize Simulated Trajectories - Programmer's Manual.

This work was conducted under the direction of Joseph Rehder of the Space Systems Division, National Aeronautics and Space Administration, Langley Research Center.

CONTENTS

FINAL REPORT

PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES (POST)

VOLUME I - FORMULATION MANUAL

By G. L. Brauer, D. E. Cornick, A. R. Habeger,
F. M. Petersen, and R. Stevenson
Martin Marietta Corporation

SUMMARY

This report documents the equations and the numerical techniques used in the Program to Optimize Simulated Trajectories (POST).

POST, a generalized point mass, discrete parameter targeting and optimization program, provides the capability to target and optimize point mass trajectories for a powered or unpowered vehicle operating near a rotating oblate planet. POST has been used successfully to solve a wide variety of atmospheric flight mechanics and orbital transfer problems. The generality of the program is evidenced by its N-phase simulation capability, which features generalized planet and vehicle models. This flexible simulation capability is augmented by an efficient discrete parameter optimization capability that includes equality and inequality constraints.

POST was originally written in FORTRAN IV for the CDC 6000 series computers. However, it is also operational on the IBM 370 and the UNIVAC 1108 computers.

Other volumes in the final report are:

Volume II - Utilization Manual - Documents information pertinent to users of the program. It describes the input required and output available for each of the trajectory and targeting/optimization options.

Volume III - Programers Manual - Documents the program structure and logic, subroutine descriptions, and other pertinent programing information.

I. INTRODUCTION

POST is a general purpose FORTRAN program for simulating and optimizing point mass trajectories of aerospace type vehicles. The program can be used to solve a wide variety of performance and mission analysis problems for atmospheric and orbital vehicles. For example, typical applications of POST are outlined in Table I-1.

Table I-1. - Typical Applications of POST

Type of Mission	Type of Vehicle	Optimization Variables	Typical Constraints		CPU Time Required to Solve Problems, min
			Equality	Inequality	
Ascent to Near-Earth Orbit	Titan IIIC & D&E, Space Shuttle, Single Stage to Orbit (VTO and HTO)	Payload, Weight at Burnout Fuel, Burntime, Ideal Velocity, Initial Weight	Radius Flight Path Angle Velocity	Dynamic Pressure Accelerations	2 → 20
Ascent to Synchronous Equatorial Orbit	Titan IIIC, Shuttle/Tug	Payload	Apogee Perigee Inclination	Dynamic Pressure Angle of Attack Pitch Rates	3 → 50
Ascent Abort	Space Shuttle	Abort Interval	Landing Site Longitude and Latitude	Acceleration Dynamic Pressure	2 → 5
ICBM Ballistic Missile	Titan II, Minuteman I & II, Safeguard	Payload Misc Distance	Latitude Longitude Crossrange Downrange	Flight Path Angle at Entry Acceleration during Entry	2 → 20
Reentry	Space Shuttle, X-24C, Single Stage to Orbit	Heat Rate Total Heat Crossrange	Latitude Longitude Crossrange Downrange	Heat Rate Acceleration	3 → 15
Orbital Maneuvers	Transtage, Space Tug, IUS, Solar Electrical Propulsion	Payload Fuel	Radius Velocity Flight Path Angle Argument of Perigee Period Apogee, Perigee	Attitude Angles Perigee Altitude	0.5 → 10
Aircraft Performance	X-24B and C, Space Shuttle Subscale, Subsonic Jet Cruise, Hypersonic Bombers and Interceptors	Mach Number Cruise Time Payload	Downrange Crossrange Dynamic Pressure Velocity and Mach Altitude	Dynamic Pressure Dynamic Pressure at Max Altitude	0.1 → 5

One of the key features of POST is an easy to use NAMELIST-type input procedure. This feature significantly reduces input deck set-up time (and costs) for studies that require the normal large amount of input data. In addition, the general applicability of POST is further enhanced by a general-purpose discrete parameter targeting and optimization capability. This capability can be used to solve a broad spectrum of problems related to the performance characteristics of aerospace vehicles.

The basic simulation flexibility is achieved by decomposing the trajectory into a logical sequence of simulation segments.

These trajectory segments, referred to as phases, enable the trajectory analyst to model both the physical and the nonphysical aspects of the simulation accurately and efficiently. By segmenting the mission into phases, each phase can be modeled and simulated in a manner most appropriate to that particular flight regime. For example, the planet model, the vehicle model, and the simulation options can be changed in any phase to be compatible with the level of detail required in that phase.

Every computational routine in the program can be categorized according to five basic functional elements. These elements are: the planet model, the vehicle model, the trajectory simulation model, the auxiliary calculations module, and the targeting and optimization module. The planet model is composed of an oblate spheroid model, a gravitational model, an atmosphere model, and a winds model. These models define the environment in which the vehicle operates. The vehicle model comprises mass properties, propulsion, aerodynamics and aeroheating and a navigation and guidance model. These models define the basic vehicle simulation characteristics. The trajectory simulation models are the event-sequencing module that controls the program cycling, table interpolation routines, and several standard numerical integration techniques. These models are used in numerically solving the translational and rotational equations of motion. The auxiliary calculations module provides for a wide variety of output calculations. For example, conic parameters, range calculations, and tracking data are among the many output variables computed. The targeting and optimization module provides a general discrete parameter iteration capability. The user can select the optimization variable, the dependent variables, and the independent variables from a list of more than 400 program variables. An accelerated projected gradient algorithm is used as the basic optimization technique. This algorithm is a combination of Rosen's projection method for nonlinear programming and Davidon's variable metric method for unconstrainted optimization. In the targeting mode, the minimum norm algorithm is used to satisfy the trajectory constraints. The cost and constraint gradients required by these algorithms are computed as first differences calculated from perturbed trajectories. To reduce the costs of calculating numerical sensitivities, only that portion of the trajectory influenced by any particular independent variable is reintegrated on the perturbed runs. This feature saves a significant amount of computer time when targeting and optimization is performed.

POST is operational on several computer systems as described in the tabulation.

Location	Computer	Operating System
Martin Marietta Corporation Denver, Colorado	CDC 6400, 6500	SCOPE 3.4.1
Martin Marietta Corporation Michoud, Louisiana	UNIVAC 1110	EXEC 8
Langley Research Center Hampton, Virginia	CDC 6600	SCOPE 3.2
Johnson Spacecraft Center Houston, Texas	UNIVAC 1108	EXEC 8
Goddard Spaceflight Center Greenbelt, Maryland	IBM 370-192	OS
Marshall Spaceflight Center Huntsville, Alabama	UNIVAC 1108	EXEC 2

Basic program macrologic is outlined in figure I-1, which illustrates the linkage between the simulation and the iteration modules.

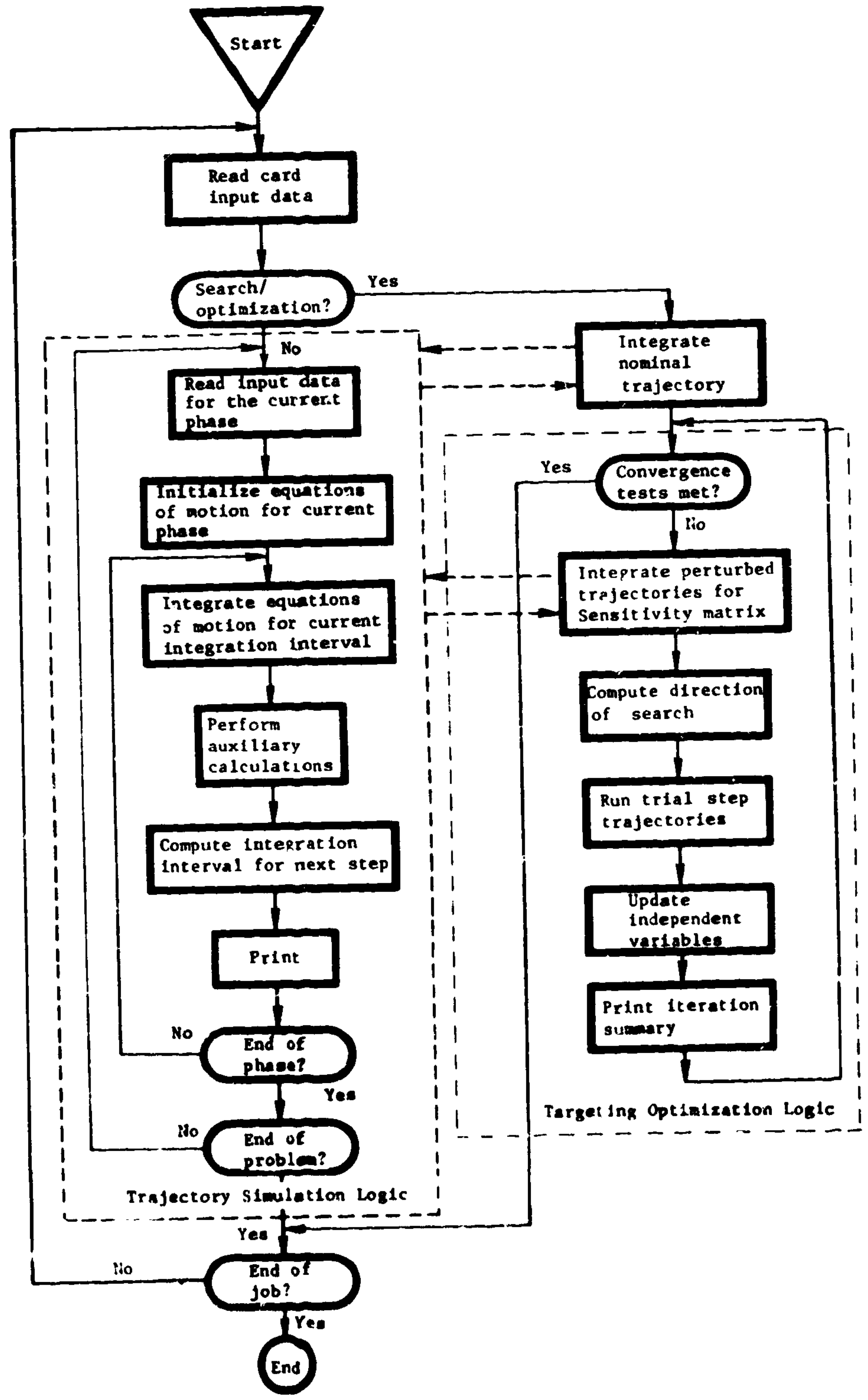

Start
Read card input data
Search/ optimization?
Yes
No
Read input data for the current phase
Initialize equations of motion for current phase
Integrate equations of motion for current integration interval
Perform auxiliary calculations
Compute integration interval for next step
Print
No
End of phase?
Yes
No
End of problem?
Trajectory Simulation Logic
Integrate nominal trajectory
Yes
Convergence tests met?
No
Integrate perturbed trajectories for Sensitivity matrix
Compute direction of search
Run trial step trajectories
Update independent variables
Print iteration summary
Targeting Optimization Logic
Yes
No
End of job?
Yes
End

Figure I-1.- Program Macrologic

II. LIST OF SYMBOLS AND ABBREVIATIONS

Math symbol	Internal Fortran symbol	Definition
a	SEMJAX	semimajor axis, m (ft)
$\underline{A}, A$	---	component of radius vector perpendicular to Sun vector, m (ft)
$\underline{A}_{AB} = \left(A_{AXB}, A_{AYB}, A_{AZB}\right)$	---	aerodynamic acceleration in the body frame, mps^2 (fps^2)
$[AB]$	AB(I)	matrix transformation from the A-frame to the B-frame
A_c	---	centrifugal acceleration, mps^2 (fps^2)
A_E	AR	nozzle exit area of each rocket engine, m^2 (ft^2)
A_H	AHORIZ	horizontal acceleration, mps^2 (fps^2)
$\underline{A}_{It}$	AXIT, AYIT, AZIT	accleration of target vehicle in the (ECI) frame, mps^2 (fps^2)
A_i, a_i	---	constants
A_M, A_{MP}, A_{MY}	AMXB, AMYB, AMZB	total aerodynamic moment about the roll, pitch, yaw axes, N-m (ft-lb)
$\left[A_n\right]$	A(I)	Davidon deflection matrix component
A_S	ASM	total sensed acceleration, mps^2 (fps^2)
$\underline{A}_{SB} = \left(A_{SXB}, A_{SYB}, A_{SZB}\right)$	AXB, AYB, AZB	total sensed acceleration in the body frame, mps^2 (fps^2)
$\underline{A}_{SI} = \left(A_{SXI}, A_{SYI}, A_{SZI}\right)$	ASXI, ASYI, ASZI	total sensed acceleration in the inertial frame, mps^2 (fps^2)
$\underline{A}_{TB} = \left(A_{TXB}, A_{TYB}, A_{TZB}\right)$	---	thrust acceleration in the body frame, mps^2 (fps^2)

Math symbol	Internal Fortran symbol	Definition
A_V	AVERT	vertical acceleration, mps^2 (fps^2)
A_{ZL}	AZL	azimuth of the z_L axis, rad (deg)
A_{ZI}, A_{ZR}, A_{ZA}	AZVELI, AZVELR, AZVELA	azimuth of the inertial, relative, and atmospheric relative velocity vectors, rad (deg)
A_{ZW}	AZWT	wind azimuth, rad (deg)
A_{ZT}	TKAZMI	azimuth of the slant range vector to the tracking station, rad (deg)
B_i	---	boundary for ith constraint
B_n	B(I)	Davidon deflection matrix
B(R)	---	boundary of region R
$B(\underline{\hat{u}})$	---	local boundary hyper surface
C_A, C_Y, C_N	CA, CY, CN	axial, side force, and normal aerodynamic force coefficients
C_{A_O}, C_{Y_O}, C_{N_O}	---	component of C_A, C_Y, C_N that is not multiplied by a mnemonic multiplier
C_D, C_L	CD, CL	drag and lift coefficients
C_{D_O}, C_{L_O}	---	drag and lift coefficient components that are not multiplied by a mnemonic variable

Math symbol	Internal Fortran symbol	Definition
C_M, C_n	CM, CW	pitch and yaw moment coefficients
C_S	CS	speed of sound, mps (fps)
$\underline{C}(\underline{u})$	E(I)	constraint functions
D	DRAG	aerodynamic drag, N (lb)
E	ECCAN	eccentric anomaly
[E]	---	Euler parameter matrix
$\underline{e}$	ECCEN	eccentricity
$\underline{e} = (e_0, e_1, e_2, e_3)$	EØ(I)	Euler parameters
$\underline{e}$	E(I)	active constraint error vector
$\underline{\tilde{e}}$	WE(I)	weighted error vector
F	ØPTVAR	optimization function
$\underline{f}$	---	nonlinear vector-valued function
$\underline{F}_{AB} = (F_{AXB}, F_{AYB}, F_{AZB})$	FAXB, FAYB, FAZB	aerodynamic forces in the body frame, N (lb)
$\underline{F}_{TB} = (F_{TXB}, F_{TYB}, F_{TZB})$	FTXB, FTYB, FTZB	thrust forces in the body frame, N (lb)
[GA]	GA(I)	matrix transformation from the G-frame to the A-frame
GHA, GHAS	GHA, GHAS	Greenwich hour angle and Greenwich hour angle of Sun, rad (deg)
$\underline{G}_I = (G_{XI}, G_{YI}, G_{ZI})$	GXI, GYI, GZI	total gravitational acceleration in the ECI-frame, mps^2 (fps^2)

Math symbol	Internal Fortran symbol	Definition
$\underline{g}_n$	DG(I)	difference in the gradient vector $\underline{\nabla F}$ between the current and previous iteration
H	---	gravitational constant
h	ALTITØ	oblate altitude, m (ft)
$\underline{h} = \left(h_{XI}, h_{YI}, h_{ZI}\right)$	ANGMØM	angular momentum, mps^2 (fps^2)
h_a, h_p	ALTA, ALTP	altitude of apogee and perigee, km (n mi)
H_B	HB	base altitude used in atmospheric calculations, m (ft)
h_c	P2	constraint function
H_g	HT	geopotential altitude, m (ft)
H_{R_i}	---	heating ratios
h_T	TRKHTI	altitude of tracker, m (ft)
h_0	P1NET	estimated net cost function
i	INC	relative-frame orbital inclination, rad (deg)
[IB]	IB(I)	matrix transformation from the ECI-frame to the body frame
[IG]	IG(I)	matrix transformation from the ECI-frame to the geographic frame

Math symbol	Internal Fortran symbol	Definition
[IL]	IL(I)	matrix transformation from the ECI-frame to the launch frame
[IP]	IP(I)	matrix transformation from the ECI-frame to the planet frame
I_{sp}	ISPV	rocket specific impulse, s
[J]	ACØB(J)	constraint Jacobian matrix
J_2, J_3, J_4	J2, J3, J4	gravitational constants
$\underline{k} = \left(k_1, k_2, k_3, k_4\right)$	---	Runge-Kutta constants
K_i	---	constants
L	LIFT	aerodynamic lift, N (lb)
[LB]	LB(I)	matrix transformation from the launch frame to the body frame
ℓ	LREF	aerodynamic reference length, m (ft)
M	MACH	Mach number
M	MEAN	mean anomaly, rad (deg)
$\underline{M}$	---	pitch and yaw moment equations
m	MASS	vehicle mass, kg (slug)
M_f	---	mnemonic table multiplier for table f
n_a	NAC	number of active constraints

Math symbol	Internal Fortran symbol	Definition
n_c	NDEPV	number of constraints
$[P]$, $[\tilde{P}]$	PRØJ(I)	projection operators used in the projected gradient method
$p(h)$	PRES	atmospheric pressure, N/m^2 (psf)
P_1	P1	weighted optimization variable
P_2	P2	weighted constraint error function
Q	TLHEAT	total heat, J/m^2 (Btu/ft^2)
q	DYNP	dynamic pressure, N/m^2 (lb/ft^2)
$\dot{Q}_{lam}$, $\dot{Q}_{turb}$	HEATRT, HTURB	laminar and turbulent heat rate, $W/m^2/s$ ($Btu/ft^2/s$)
$Q(\hat{\underline{u}})$, $\tilde{Q}(\hat{\underline{u}})$	---	linear manifold and its orthogonal complement
R_A	RTASC	right ascension of outgoing asymptote, rad (deg)
r_a	APØRAD	apogee radius, m (ft)
$[RB]$	RB(I)	matrix transformation from the body reference frame to the body frame
R_D	DPRNG1	dot-product range, km (n mi)
R_E, R_P	RE, RP	equatorial and polar radius, m (ft)
R_N	RN	nose radius, m (ft)

Math symbol	Internal Fortran symbol	Definition
R_{NU}	REYNØ	Reynolds number
$\underline{r}_I = (x_I, y_I, z_I)$	XI, YI, ZI	inertial radius vector from center of planet to the vehicle, m (ft)
r_I	GCRAD	geocentric radius, m (ft)
$\underline{r}_{It}$	XIT, YIT, ZIT	radius vector to target vehicle, m (ft)
r_p	PGERAD	perigee radius, m (ft)
R_s	RS	radius to oblate surface, m (ft)
$\underline{r}_{SR}$	---	slant range vector, m (ft)
$\underline{r}_{SRG}$	---	slant range vector in geographic frame, m (ft)
$\underline{r}_{TR}$	---	radius vector to tracking station, m (ft)
$\underline{r}_{VE}$	XVE, YVE, ZVE	Radius vector of vehicle in vernal equinox system, m (ft)
$\underline{s}$	S(I)	direction of search
$\underline{s}^c$	---	direction of search to satisfy the constraints
S	SHADF	shadow function, m (ft)
S_{loss_i}	SLØSIJ	space losses for tracking stations, dB
S_{ref}	SREF	aerodynamic reference area, m^2 (ft^2)
$\underline{s}^o$	---	direction of search for optimization
T	ATEM	atmospheric temperature, °K (°F)
t	TIME	time, s
T_j/δ	---	jet engine thrust, N (lb)

Math symbol	Internal Fortran symbol	Definition
T_{M1P}	TTMYB	total thrust moment for nontrimming engines in the pitch plane, body axis system, N-m (ft-lb)
T_{M2P}	TTMZB	total thrust moment for trimming engines in the pitch plane, body axis system, N-m (ft-lb)
T_{M1Y}	TMYB	total thrust moment in the yaw plane for nontrimming engines, body axis system, N-m (ft-lb)
T_{M2Y}	TMZB	total thrust moment in the yaw plane for the trimming engines, body axis system, N-m (ft-lb)
$T^n(y)$	---	denotes nth order table interpolation on the variable y
T_R	THRUST	total rocket thrust for all engines, N (lb)
T_{R_i}	---	total resultant rocket thrust for engine i, N (lb)
T_{vac}	TVAC	vacuum thrust for rocket engines, N (lb)
T_{SP}	TIMSP	time since perigee passage, s
T_{TP}	TIMTP	time to next perigee passage, s
U	---	gravitational potential function
$\underline{u}$	U(I)	independent variable

Math symbol	Internal Fortran symbol	Definition
$\underline{V}_{AB} = (u_B, v_B, w_B)$	UB, VB, WB	components of the atmospheric relative velocity vector expressed in the body frame, mps (fps)
$\underline{u}_{RI}$	---	unit vector along the radius vector
$\underline{u}_s$	XSUE, YSUE, ZSUE	unit vector in Sun direction in the vernal equinox system
$\underline{u}_{sI}$	XSI, YSI, ZSI	unit vector in Sun direction in the ECI system
$\underline{u}_{VI}$	---	unit vector along the velocity vector
$\Delta\underline{u}$	DU(I)	change in the independent variables
V_a	APVEL	inertial velocity at apogee, mps (fps)
$\underline{V}_{AG}$	UA, VA, WA	atmospheric relative velocity in the G-frame, mps (fps)
$\underline{V}_{AI}$	VAXI, VAYI, VAZI	atmospheric relative velocity vector in the inertial frame, mps (fps)
$\underline{V}_I = (V_{XI}, V_{YI}, V_{ZI})$	VXI, VYI, VZI	inertial velocity vector and its magnitude, mps (fps)
V_I	VELI	magnitude of $\underline{V}_I$, mps (fps)
$\underline{V}_{IG}$	U, V, W	inertial velocity in the G-frame, mps (fps)
$\underline{V}_{It}$	VXIT, VYIT, VZIT	velocity of target vehicle in ECI system, mps (fps)
$\underline{V}_R$	VELR	relative velocity, mps (fps)
$\underline{V}_{RG}$	UR, VR, WR	relative velocity in the G-frame, mps (fps)
$\underline{V}_{RI} = (V_{RXI}, V_{RYI}, V_{RZI})$	VRXI, VRYI, VRZI	relative velocity vector in the inertial frame, mps (fps)

Math symbol	Internal Fortran symbol	Definition
$\underline{V}_{VE}$	VXVE, VYVE, VZVE	velocity of vehicle in vernal equinox system, mps (fps)
$\underline{V}_{WI} = (V_{WXI}, V_{WYI}, V_{WZI})$	VWXI, VWYI, VWZI	wind velocity vector in the inertial frame, mps (fps)
V_W	VW	wind velocity, mps (fps)
$\underline{V}_{WG}$	UW, VW, WW	wind velocity vector in the G-frame, mps (fps)
V_p	PGVEL	perigee velocity, mps (fps)
V_∞	HYPVEL	outgoing asymptote velocity, mps (fps)
$\dot{W}$	WDØT	total time rate of change of vehicle weight, N/s (lb/s)
W_C	WEICØN	total weight of propellant consumed, N (lb)
W_G	WEIGHT	gross vehicle weight, N (lb)
W_{jett}	WJETTM	jettison weight, N (lb)
W_{PC}	---	weight of propellant consumed per phase, N (lb)
W_{P_i}	---	initial propellant weight, N (lb)
$\dot{W}_{P_i}^{max}$	---	maximum flowrate for the i th engine, N/s (lb/s)
W_{PR}	WPRØP	weight of propellant remaining, N (lb)
W_{stg}	WGTSG	vehicle stage weight, N (lb)
$[W_u]$, $[W_f]$, $[W_e]$	WU, WØPT, WE	weighting matrices for $\underline{u}$, $\underline{f}$, and $\underline{e}$

Math symbol	Internal Fortran symbol	Definition
x_B, y_B, z_B	---	coordinate axes of the body frame
x_{BR}, y_{BR}, z_{BR}	---	coordinate axes of the body reference frame
x_{cg}, y_{cg}, z_{cg}	XCG, YCG, ZCG	coordinates of the center of gravity in the body reference system, m (ft)
x_G, y_G, z_G	---	components of a vector in the geographic frame, m (ft)
x_I, y_I, z_I	XI, YI, ZI	components of the radius vector in the inertial frame, m (ft)
x_i	---	general state variable
x_L, y_L, z_L	---	coordinate axes of the launch frame
$\underline{x}_n$	GINTJ	state vector at the nth event
x_R, y_R, z_R	---	components of the radius vector in the planet frame, m (ft)
x_{ref}, y_{ref}, z_{ref}	XREF, YREF, ZREF	coordinates of the aerodynamic reference point in the body reference system, m (ft)
$\underline{y}$	DGENV	general dependent variable
α, β, σ	ALPHA, BETA, BNKANG	aerodynamic angle of attack, sideslip, and bank, rad (deg)
α_T	ALPTØT	total angle of attack, rad (deg)

Math symbol	Internal Fortran symbol	Definition
$\gamma_I, \gamma_R, \gamma_A$	GAMMAI, GAMMAR, GAMMAA	inertial, relative, and atmospheric relative flight path angles, rad (deg)
γ_j	GAMMA(I)	step-size parameter on the j th trial step
ΔE	---	increment in eccentric anomaly, rad (deg)
Δh	---	increment in altitude, m (ft)
Δt	DT	increment in time or integration step size, s
ΔV	DV	increment in velocity, mps (fps)
ΔV^*	VIDEAL	ideal velocity, mps (fps)
ΔV_A	DLR	atmospheric velocity loss, mps (fps)
ΔV_c	DVCIR	velocity required to circularize an orbit, mps (fps)
ΔV_E	DVEXS	excess velocity, mps (fps)
ΔV_G	GLR	gravity loss, mps (fps)
ΔV_M	DVMAR	velocity margin, mps (fps)
ΔV_P	ATLR	atmospheric pressure loss, mps (fps)
ΔV_{TV}	TVLR	thrust vector velocity loss, mps (fps)
δ_A	RTASC	right ascension, rad (deg)
$\delta_{cone}, \delta_{clock}$	SCONE, SCLOCK	cone and clock angles of Sun vector in body system, rad (deg)

Math symbol	Internal Fortran symbol	Definition
η	ETA	engine throttling parameter
θ	LØNG	planet relative longitude, rad (deg)
θ^*	---	longitude reference, rad (deg)
θ_I	LØNGI	inertial longitude, rad (deg)
θ_L, ϕ_L, A_{ZL}	LØNL, LATL, AZL	longitude, latitude, and azimuth of L-frame, rad (deg)
θ_{max}	TRUNMX	maximum true anomaly for hyperbolic orbit, rad (deg)
θ_{T_i}	TRKLNI	longitude of tracker i, rad (deg)
λ	AZREF	azimuth reference, rad (deg)
λ	STPMAX	maximum admissible step size for the iteration algorithm
μ	MU	gravitational constant, m^3/s^2 (ft^3/s^2)
ν	---	index

Math symbol	Internal Fortran symbol	Definition
ρ	ARGV	argument of vehicle (i.e., angular location of vehicle, measured from ascending node in orbital plane), rad (deg)
$\rho(h)$	---	atmospheric density, kg/m^3 (slug/ft^3)
T	---	trajectory propagation
ϕ_c	GCLAT	geocentric latitude. rad (deg)
ϕ_g	GDLAT	geodetic latitude, rad (deg)
ϕ_I, ψ_I, θ_I	RØLI, YAWI, PITI	inertial roll, yaw, and pitch measured as positive rotations from the L-frame, rad (deg)
ψ_R, θ_R, ϕ_R	YAWR, PITR, RØLR	relative yaw, pitch, and roll, measured in a positive sense from the geographic frame, rad (deg)
Ω	LAN	longitude of ascending node, rad (deg)
Ω_p	ØMEGA	angular rotation rate of planet about the polar axis, rad/s (deg/s)
Ω_S	RASGM	right ascention of Greenwich meridian, rad (deg)
ω	---	argument of perigee, rad (deg)
$\underline{\omega} = \left(\omega_x, \omega_y, \omega_z\right)$	RØLBD, PITBD, YAWBD	inertial angular velocity components about the body axis, rad/s (deg/s)
$\underline{\dot{\omega}}$	RØLBDD, PITBDD, YAWBDD	inertial angular acceleration components about the body axis, rad/s^2 (deg/s^2)

Math symbol	Internal Fortran symbol	Definition
$(\ )_A$		refers to atmosphere relative variables
$(\ )_{cg}$		refers to center of gravity
$(\ )_I$		refers to inertial variables
$(\ )_n$		refers to nth event
$(\ )_p$		refers to thrust application
$(\ )_R$		refers to Earth-relative variables
$(\ )_{Ref}$		refers to aerodynamic reference point
$(\ )_{SL}$		refers to sea-level conditions
$(\ )_{vac}$		refers to vacuum conditions
$(\ )_W$		refers to wind relative variables
$(\ )^*$		refers to state from which downrange and crossrange are referenced; refers to optimal conditions
$\underline{(\ )}$		denotes vector quantity
$(\ )'$		denotes transpose of a vector
$\dot{(\ )}$		denotes total derivative with respect to time

III. COORDINATE SYSTEMS

POST uses numerous coordinate systems to provide the necessary reference systems for calculating required and optional data. These coordinate systems and the key transformations are described below.

Coordinate System Definitions

Earth-centered inertial (ECI) axes $\left(x_I, y_I, z_I\right)$.- This system is an Earth-centered Cartesian system with z_I coincident with the North Pole, x_I coincident with the Greenwich Meridian at time zero and in the equatorial plane, and y_I completing a right-hand system. The translational equations of motion are solved in this system (fig. III-1).

Earth-centered rotating (ECR) axes $\left(x_R, y_R, z_R\right)$.- This system is similar to the ECI system except that it rotates with the Earth so that x_R is always coincident with the Greenwich Meridian (fig. III-1).

Earth position coordinates $\left(\phi_g, \theta, h\right)$. These are the familiar latitude, longitude, and altitude designators. Latitude is positive in the Northern Hemisphere. Longitude is measured positive East of Greenwich. Altitude is measured positive above the surface of the planet (fig. III-1).

Geographic (G) axes $\left(x_G, y_G, z_G\right)$. - This system is located at the surface of the planet at the vehicle's current geocentric latitude and longitude. The x_G axis is in the local horizontal plane and points North, the y_G axis is in the local horizontal plane and points East, and z_G completes a right-hand system. This system is used to calculate parameters associated with azimuth and elevation angles (fig. III-2).

Inertial launch (L) axes $\left(x_L, y_L, z_L\right)$. - This is an inertial Cartesian system that is used as an inertial reference system from which the inertial attitude angles of the vehicle are measured. This coordinate system is automatically located at the

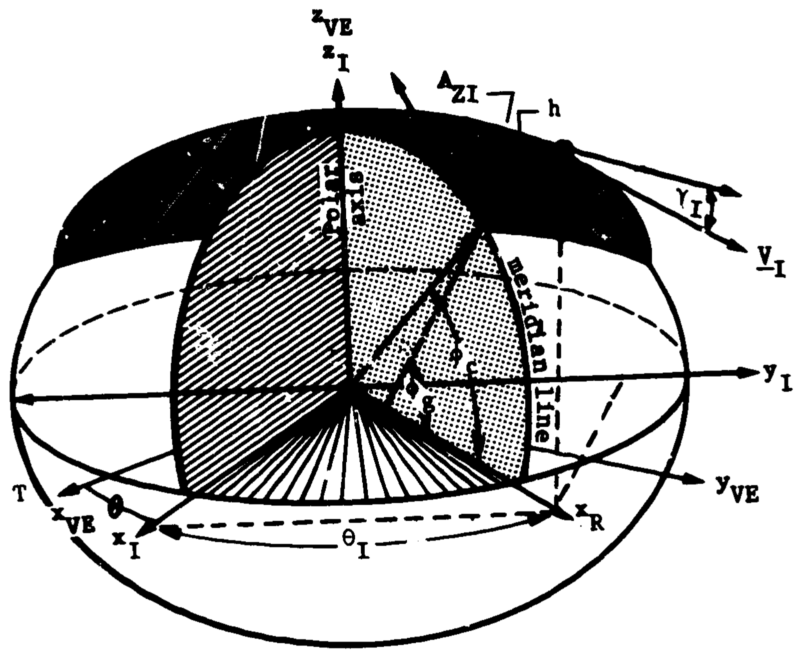


Figure III-1.- Coordinate Systems

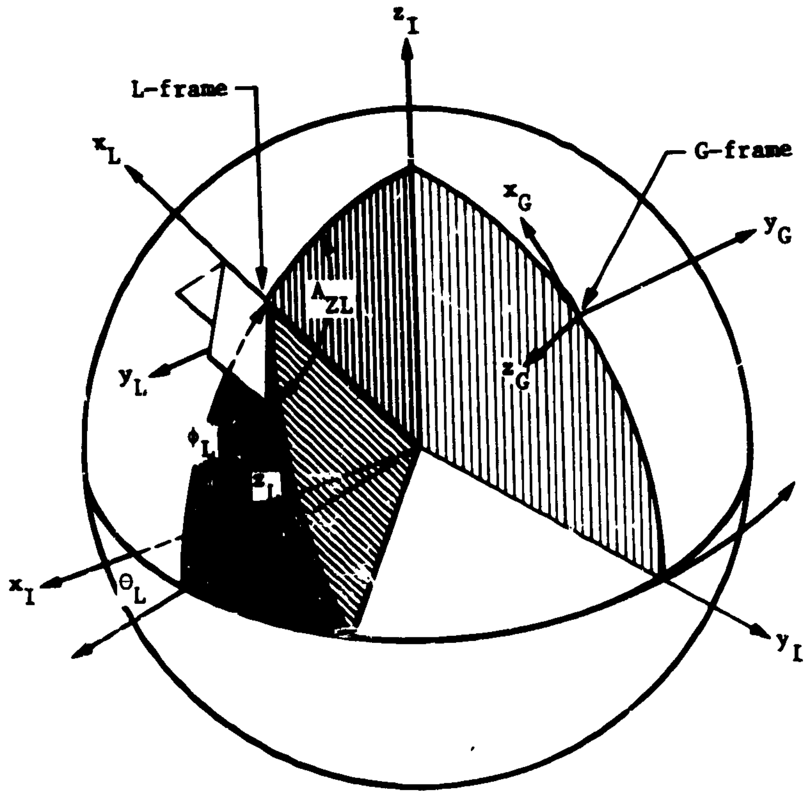


Figure III-2.- Launch Frame

geodetic latitude and inertial longitude of the vehicle at the beginning of the simulation unless overridden by user input of LATL and LØNL. The azimuth, A_{ZL}, is zero unless overidden by user input. The orientation of this system is such that x_L is along the positive radius vector if ϕ_L is input as the geocentric latitude, or along the local vertical if ϕ_L is not input or is input as the geodetic latitude. z_L is in the local horizontal plane and is directed along the azimuth specified by A_{ZL}, and y_L completes a right-hand system. This system is intended for use in simulating ascent problems for launch vehicles that use either inertial platform or strapdown-type angular commands. The inertial angles, $(\phi_I, \psi_I, \theta_I)$ are always measured with respect to this system and are automatically computed regardless of the steering option (IGUID) being used (fig. III-2).

Body (B) axes (x_B, y_B, z_B).- The body axes form a right-hand Cartesian system aligned with the axes of the vehicle and centered at the vehicle's center of gravity. The x_B axis is directed forward along the longitudinal axis of the vehicle, y_B points right (out the right wing), and z_B points downward, completing a right-hand system. All aerodynamic and thrust forces are calculated in the body system. These forces are then transformed to the inertial (I) system where they are combined with the gravitational forces (fig. III-3)

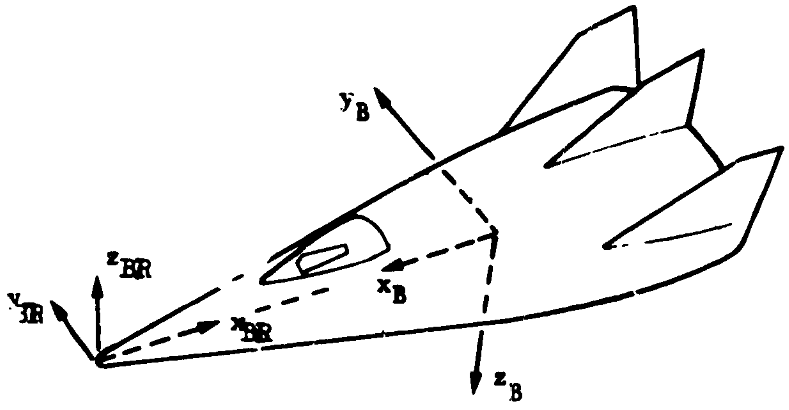


Figure III-3.- Body Frame.

Body reference (BR) axes $\left(x_{BR}, y_{BR}, z_{BR}\right)$.- The body reference system is a right-hand Cartesian system aligned with the body axes as follows. The x_{BR} axis is directed along the negative x_B axis, the y_{BR} axis is directed along the positive y_B axis, and the z_{BR} is directed along the negative z_B axis. This system is used to locate the vehicle's center of gravity, aerodynamic reference point, and engine gimbal locations for the static trim operation (fig. III-3).

Orbital elements $\left(h_a, h_p, i, \omega, \theta, \Omega\right)$.- This is a nonrectangular coordinate system used in describing orbital motion. The orbital elements are apogee altitude, perigee altitude, inclination, longitude of the ascending node, true anomaly, and argument of perigee. The apogee and perigee altitudes replace the standard orbital elements of semimajor axis and eccentricity (fig. III-4).

Vernal Equinox (VE) Axes $\left(x_{VE}, y_{VE}, z_{VE}\right)$.- This is the 1950 mean equator and equinox Earth centered inertial system. The x_{VE} axis is in the equatorial plane and is directed forward of the vernal equinox of 1950, the z_{VE} axis is directed along the north pole, and y_{VE} completes the right hand system (fig. III-1).

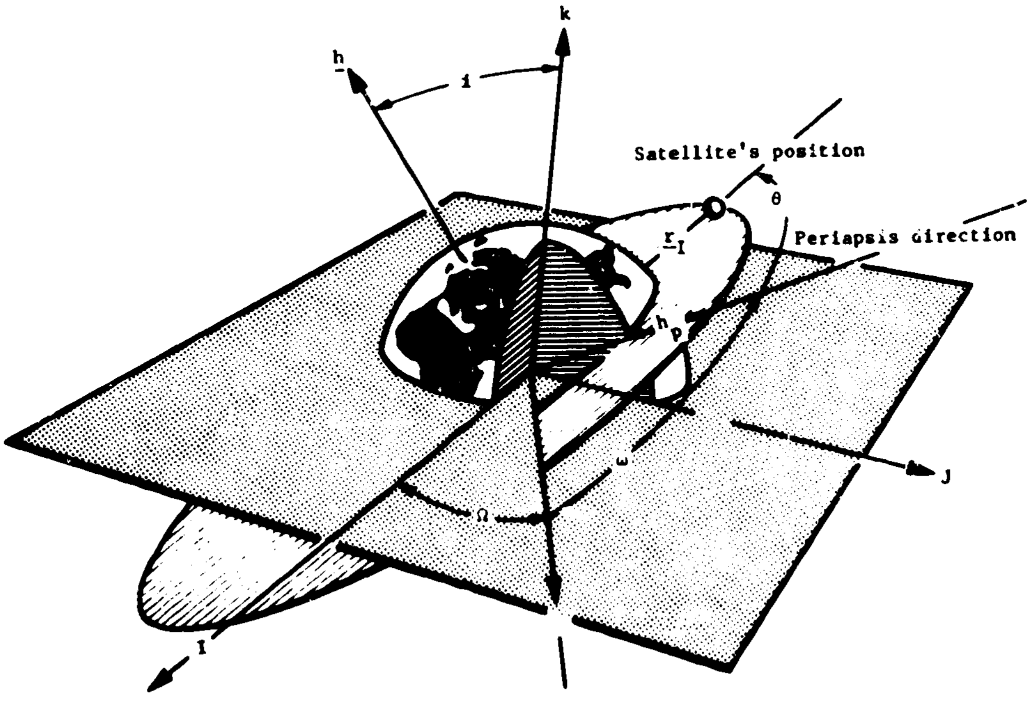


Figure III-4.- Orbital Parameters

Attitude Angles

The program contains the following standard attitude refe ence systems:

1) Inertial Euler angles;

2) Relative Euler angles;

3) Aerodynamic angles;

4) Inertial aerodynamic angles;

These variables are defined and illustrated below:

1) Inertial Euler angles (fig. III-5):

ϕ_I - Inertial roll angle. The roll angle with respect to the L-frame (first rotation),

ψ_I - Inertial yaw angle. The yaw angle with respect to the L-frame (second rotation),

θ_I - Inertial pitch angle. The pitch angle with respect to the L-frame (third rotation);

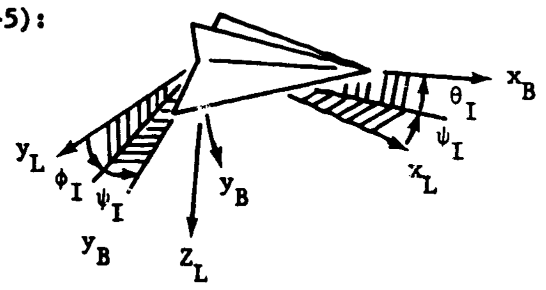


Figure III-5.- Inertial Euler Angles

2) Relative Euler angles (fig. III-6):

ψ_R - Relative yaw angle. This is the azimuth angle of the x_B axis measured clockwise from the reference direction (first rotation),

θ_R - Relative pitch angle. This is the elevation angle of the x_B axis above the local horizontal plane (second rotation),

ϕ_R - Relative roll angle. This is the roll angle about the x_B axis (third rotation).

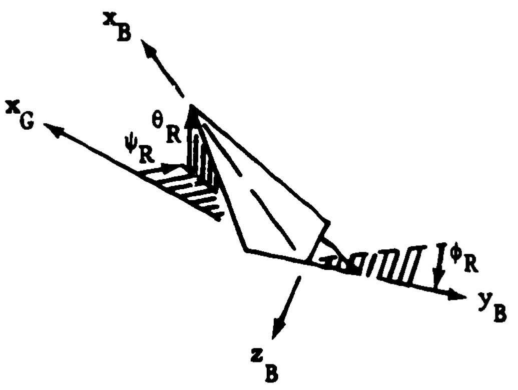


Figure III-6.- Relative Euler Angles

3) Aerodynamic angles (fig. III-7):

σ - Bank angle. Positive σ is a positive rotation about the atmosphere relative velocity vector (first rotation),

β - Sideslip. Positive β is a nose-left (negative) rotation when flying the vehicle upright (second rotation),

α - Angle of attack. Positive α is a nose-up (positive) rotation when flying the vehicle upright (third rotation);

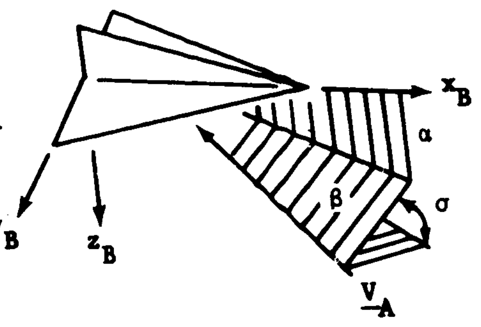


Figure III-7.- Aerodynamic Angles

4) Inertial aerodynamic angles (fig. III-8):

σ_I - Bank angle. Positive σ_I is a positive rotation about the atmosphere inertial velocity vector (first rotation),

β_I - Sideslip. Positive β_I is a nose-left (negative) rotation when flying the vehicle upright (second rotation),

α_I - Angle of attack. Positive α_I is a nose-up (positive) rotation when flying the vehicle upright (third rotation);

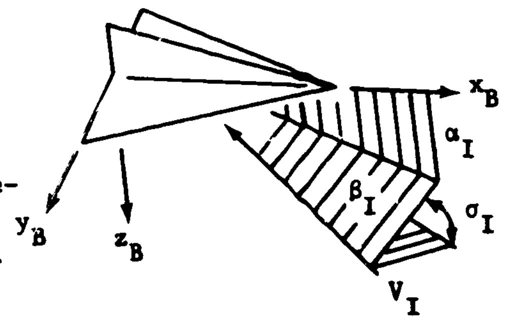


Figure III-8.- Inertial Aerodynamic Angles

Transformations

Numerous matrix transformations are required to transform data between the coordinate systems described in the previous section. The most important of these transformations is the [IB] matrix. The inverse (transpose) of this matrix is used to transform accelerations in the body frame to the planet-centered inertial frame. The remaining transformations are generally used to either compute [IB] or to transform auxiliary data into some convenient output coordinate system.

The [IB] matrix is functionally dependent on the attitude of the vehicle. This dependence is described by equations related to the attitude steering option selected by the user. The following matrix equations, which depend on this steering option, are used to compute the [IB] matrix.

$$\left.\begin{aligned} [IB] &= [LB][IL] \text{ (body rates or inertial Euler angles)} \\ [IB] &= [GB][IG] \text{ (relative Euler angles)} \\ [IB] &= [AB][GA][IG] \text{ (aerodynamic angles)} \end{aligned}\right\} \quad \text{(III-1)}$$

The basic relationships between the coordinate systems defined by these equations are illustrated in figure III-9. The inverse transformation can generally be computed by merely transposing the matrix elements because of the orthonormality of these matrices.

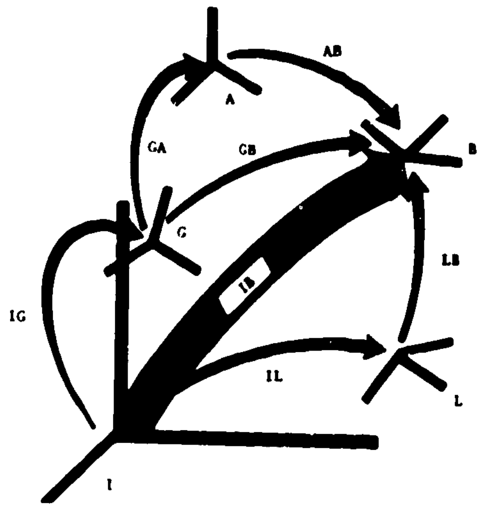


Figure III-9.- Matrix Transformations

A summary of these matrices is given below. The symbols s and c denote sin and cos, respectively.

[IL], inertial to launch.- The [IL] matrix depends on ϕ_L, θ_L, and A_{ZL}, and is given by

$$[IL] = \begin{bmatrix} c\phi_L c\theta_L & c\phi_L s\theta_L & s\phi_L \\ s\phi_L c\theta_L sA_{ZL} - cA_{ZL} s\theta_L & cA_{ZL} c\theta_L + sA_{ZL} s\phi_L s\theta_L & -sA_{ZL} c\phi_L \\ -sA_{ZL} s\theta_L - cA_{ZL} s\phi_L c\theta_L & sA_{ZL} c\theta_L - cA_{ZL} s\phi_L s\theta_L & cA_{ZL} c\phi_L \end{bmatrix} \quad \text{(III-2)}$$

[LB], launch to body.- The [LB] matrix is computed indirectly from the body rates by integrating the quaternion equations, or directly from inertial Euler angles. When the body rate option is used, the quaternion rate equation

$$\begin{bmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -e_1 & e_2 & e_3 \\ e_0 & e_2 & -e_3 \\ e_0 & -e_1 & e_3 \\ e_0 & e_1 & -e_2 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad \text{(III-3)}$$

is integrated to compute the [LB] matrix, which is then given by

$$[LB] = \begin{bmatrix} e_0^2 + e_1^2 - e_2^2 - e_3^2 & 2(e_1e_2 + e_0e_3) & 2(e_1e_3 - e_0e_2) \\ 2(e_1e_2 - e_0e_3) & e_0^2 - e_1^2 + e_2^2 - e_3^2 & 2(e_0e_1 + e_2e_3) \\ 2(e_1e_3 + e_0e_2) & 2(e_2e_3 - e_0e_1) & e_0^2 - e_1^2 - e_2^2 + e_3^2 \end{bmatrix} \quad \text{(III-4)}$$

When the inertial Euler angle option is used, [LB] is computed directly as

$$[LB] = \begin{bmatrix} c\psi_I c\theta_I & c\phi_I s\psi_I c\theta_I + s\phi_I s\theta_I & s\phi_I s\psi_I c\theta_I - c\phi_I s\theta_I \\ -s\psi_I & c\phi_I c\psi_I & s\phi_I c\psi_I \\ c\psi_I s\theta_I & c\phi_I s\psi_I s\theta_I - s\phi_I c\theta_I & s\phi_I s\psi_I s\theta_I + c\phi_I c\theta_I \end{bmatrix} \qquad \text{(III-5)}$$

<u>[IG], inertial to geographic</u>.- The [IG] matrix depends on the geocentric latitude and the inertial longitude, and is giver by

$$[IG] = \begin{bmatrix} -s\phi_c c\theta_I & -s\phi_c s\theta_I & c\phi_c \\ -s\theta_I & c\theta_I & 0 \\ -c\phi_c c_I\theta & -c\phi_c s\theta_I & -s\phi_c \end{bmatrix}. \qquad \text{(III-6)}$$

<u>[GB], geographic to body</u>.- The [GB] matrix depends on the relative Euler angles, and is given by

$$[GB] = \begin{bmatrix} c\theta_R c\psi_R & c\theta_R s\psi_R & -s\theta_R \\ s\phi_R s\theta_R c\psi_R - c\phi_R s\psi_R & s\phi_R s\theta_R s\psi_R + c\phi_R c\psi_R & s\phi_R c\theta_R \\ c\phi_R s\theta_R c\psi_R + s\phi_R s\psi_R & c\phi_R s\theta_R s\psi_R - s\phi_R c\psi_R & c\phi_R c\theta_R \end{bmatrix} \qquad \text{(III-7)}$$

<u>[GA], geographic to atmospheric relative velocity system (ARVS)</u>.- The [GA] matrix depends on the atmospheric relative flight azimuth and flightpath angles, and is given by

$$[GA] = \begin{bmatrix} c\gamma_A c\lambda_A & c\gamma_A s\lambda_A & -s\gamma_A \\ -s\lambda_A & c\lambda_A & 0 \\ s\gamma_A c\lambda_A & s\gamma_A s\lambda_A & c\gamma_A \end{bmatrix}. \qquad \text{(III-8)}$$

[AB], ARVS to body.- The [AB] matrix depends on the aerodynamic angles, and is given by

$$[AB] = \begin{bmatrix} c\alpha c\beta & -c\alpha s\beta c\sigma + s\alpha s\sigma & -c\alpha s\beta s\sigma - s\alpha c\sigma \\ s\beta & c\beta c\sigma & c\beta s\sigma \\ s\alpha c\beta & -s\alpha s\beta c\sigma - c\alpha s\sigma & -s\alpha s\beta s\sigma + c\alpha c\sigma \end{bmatrix} \qquad \text{(III-9)}$$

Other transformations, which are not related to the calculation of the [IB] matrix, are presented below.

[IP], inertial to planet relative.- The [IP] matrix transforms between the Earth-centered inertial frame and the Earth-centered rotating frame. This matrix depends on the rotation rate of the planet and the total elapsed time of flight, and is given by

$$[IP] = \begin{bmatrix} c\Omega_P t & s\Omega_P t & 0 \\ -s\Omega_P t & c\Omega_P t & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad \text{(III-10)}$$

[RB], body reference to body.- The [RB] matrix transforms data in the body reference system to the body frame. This matrix has a constant value and is given by

$$[RB] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \qquad \text{(III-11)}$$

[IV], inertial to vernal equinox.- The [IV] matrix transforms between the ECI frame and the vernal equinox frame, and is given by

$$[IV] = \begin{bmatrix} c\ \theta & s\ \Theta & 0 \\ -s\ \Theta & c\ \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \text{(III-12)}$$

where $\Theta = \text{GHA} + \dot{\Omega}\left(t - t_{\text{reference}}\right)$.

IV. PLANET MODEL

The planet model is composed of three types of data and equations. These are: (1) oblate planet geometry and constants, (2) an atmosphere model that computes atmospheric pressure, density, temperature, and speed of sound, and (3) a gravitational model that computes the gravitational accelerations. The user selects the appropriate models and inputs the corresponding data. The input data and the equations used in these models are described below.

Oblate Spheroid

The 1960 Fisher Earth model is preloaded into the program. This model is defined by the equatorial radius R_E, the polar radius R_P, the rotation rate Ω_P, the gravitational constant μ, and the second, third, and fourth gravitational harmonics, J_2, J_3, and J_4, respectively. The stored values for these constants are:

$$R_E = 2.0925741 \times 10^7 \text{ ft},$$
$$R_P = 2.0855590 \times 10^7 \text{ ft},$$
$$\Omega_P = 7.29211 \times 10^{-5} \text{ rad/s},$$
$$\mu = 1.4076539 \times 10^{16} \text{ ft}^3/\text{s}^2,$$
$$J_2 = 1.0823 \times 10^{-3},$$
$$J_3 = 0,$$
$$J_4 = 0.$$

The constants J_3 and J_4 are preloaded as zero, but can be initialized by input. For example, if the Smithsonian Earth model is desired, then these constants would be input as

$$J_2 = 1.082639 \times 10^{-3},$$
$$J_3 = -2.565 \times 10^{-6},$$
$$J_4 = -1.608 \times 10^{-6},$$
$$\mu = 1.407645794 \times 10^{16} \text{ ft}^3/\text{s}^2,$$

$$\Omega_P = 7.29211515 \times 10^{-5} \text{ rad/s},$$

$$R_E = 2.092566273 \times 10^{7} \text{ ft},$$

$$R_P = 2.08550242 \times 10^{7} \text{ ft}.$$

The geometry of this spheroid is illustrated in figure IV-1. The pertinent equations related to this model are

$$\left.\begin{aligned} \phi_c &= \sin^{-1}\left(z_I/r_I\right) \\ \phi_g &= \tan^{-1}\left(k \tan \phi_c\right), \quad k = \left(R_E/R_P\right)^2 \\ R_s &= R_E\left(1 + (k - 1)\sin^2 \phi_c\right)^{-\frac{1}{2}} \\ h &= r_I - R_s, \end{aligned}\right\} \qquad \text{(IV-1)}$$

where ϕ_c is the geocentric latitude, ϕ_g is the geodetic latitude, θ_I is the inertial longitude, θ is the relative longitude with respect to the planet, r_I is the distance from the center of the planet to the vehicle, R_s is the distance from the center of the planet to the planet surface, and h is the distance from the planet surface to the vehicle.

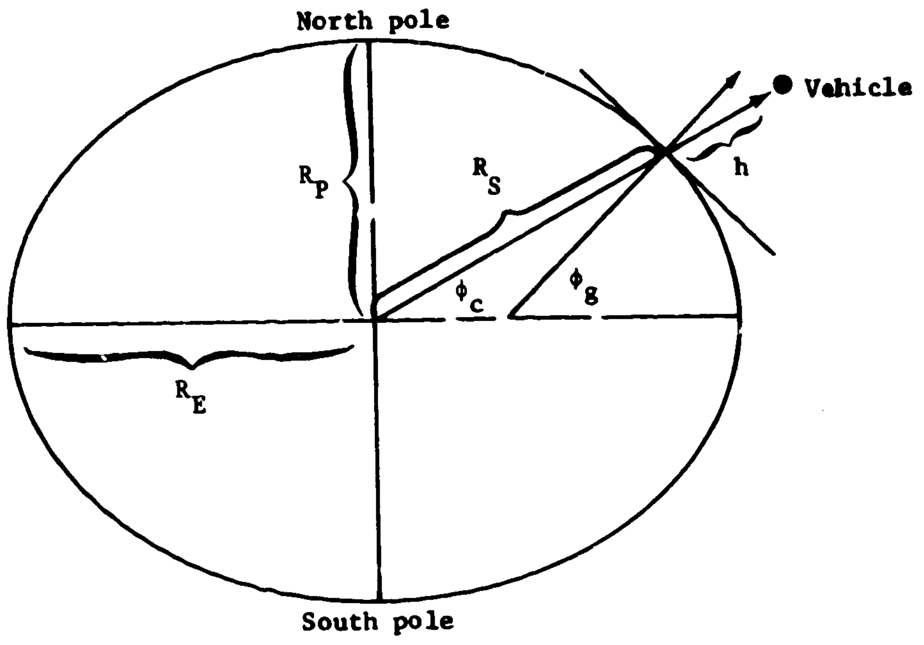


Figure IV-1.- Oblate Planet

Gravitational Model

The gravitational model includes optionally second, third, and fourth harmonic terms. The potential function for this model is

$$U = -\mu\left[\frac{1}{r} - \frac{J_2}{2} R_E^2\left(\frac{3z^2}{r^5} - \frac{1}{r^3}\right) - \frac{J_3}{2} R_E^3\left(5\frac{z^3}{r^7} - \frac{3z}{r^5}\right) - \frac{J_4}{8} R_E^4\left(35\frac{z^4}{r^9} - 30\frac{z^2}{r^7} + \frac{3}{r^5}\right)\right]. \tag{IV-2}$$

The gravitational accelerations calculated from this potential function are:

$$\left.\begin{aligned}
G_{XI} &= -\frac{\partial U}{\partial x_I} \\
&= -\mu \frac{x}{r^3} P(z, r) \\
G_{YI} &= -\frac{\partial U}{\partial y_I} \\
&= -\mu \frac{y}{r^3} P(z, r) \\
G_{ZI} &= -\frac{\partial U}{\partial z_I} \\
&= -\frac{\mu}{r^3}\left[\left(1 + JR^2 (3 - 5Z^2)\right) z + H \frac{R^3}{r}\left(6z^2 - 7z^2 Z^2 - \frac{3}{5} r^2\right)\right. \\
&\qquad \left. + DR^4 \left(\frac{15}{7} - 10Z^2 + 9Z^4\right) z\right],
\end{aligned}\right\} \quad \text{(IV-3)}$$

where $x = x_I$, $y = y_I$, $z = z_I$, $r = r_I$, and

$$\left.\begin{aligned}
R &= R_E/r_I \\
Z &= z_I/r_I \\
J &= \frac{3}{2} J_2 \\
H &= \frac{5}{2} J_3 \\
D &= -\frac{35}{8} J_4 \\
P(z, r) &= \left[1 + JR^2 (1 - 5Z^2) + H \frac{R^3}{r} (3 - 7 Z^2) z\right. \\
&\qquad \left. + DR^4 \left(9Z^4 - 6Z^2 + \frac{3}{7}\right)\right].
\end{aligned}\right\} \quad \text{(IV-4)}$$

Atmosphere Models

POST has the optional capability of three atmospheric models--the general table lookup, the 1962 U.S. standard atmosphere, and the 1963 Patrick AFB atmosphere using polynomials. The general table lookup model gives the user the flexibility of inputing his own atmospheric model if none of the preloaded models is adequate. This is particularly useful in performing trajectory analysis for planets other than Earth. The parameters required to define the atmospheric effects are the atmospheric pressure p, atmospheric density ρ, speed of sound C_s, and atmospheric temperature T. These parameters are functions of the oblate altitude h.

Table lookup atmosphere model.- The table lookup atmosphere model can be defined entirely by using tables that show pressure, temperature, speed of sound, and density as functions of altitude. The speed of sound and density tables can be omitted if desired; in this case, the speed of sound and density are computed as

$$C_s = \sqrt{K_1 T}$$
$$\rho = K_2 \frac{p}{T}, \qquad \text{(IV-5)}$$

where

$$K_1 = \frac{\gamma R^*}{M_0}$$

$$K_2 = \frac{M_0}{R^*}$$

γ = ratio of specific heats

M_0 = molecular weight

R^* = universal gas constant.

1962 U.S. standard atmosphere model.- The 1962 U.S. standard atmosphere model is given as a function of geopotential altitude (H_g), which is computed as

$$H_g = \frac{R_A h}{R_A + h}, \qquad \text{(IV-6)}$$

where

$$R_A = \text{average Earth radius} = \frac{1}{2}\left(R_E + R_P\right)$$

$$h = \text{oblate altitude.}$$

The molecular scale temperature, T_M, is defined by a series of linear segments $\left(L_M\right)$ as a function of geopotential altitude $\left(H_g\right)$.

The corner points connecting the straight-line segments are referred to as base altitudes $\left(H_B\right)$, base temperatures $\left(T_{M_B}\right)$, etc. From a table of base altitudes, base temperatures, and dT_M/dH $\left(L_M\right)$ (the slope within the linear segments), the temperature at any desired altitude can be calculated from the following equation:

$$T_M = T_{M_B} + L_{M_B}\left(H_g - H_B\right). \qquad \text{(IV-7)}$$

Values of P_B, T_{M_B}, and L_{M_B} versus H_B are presented in table IV-1.

The atmospheric pressure is determined as follows:

$$\left.\begin{aligned} P &= P_B\left[\frac{T_{M_B}}{T_M}\right]\exp\left[\left(\frac{g_0\,M_0}{R^*}\right)\Big/L_{M_B}\right] \text{ for segments with } L_{M_B} \neq 0, \text{ and} \\ P &= P_B \exp\left[-\frac{g_0\,M_0}{R^*}\,\frac{\left(H - H_B\right)}{T_{M_B}}\right] \text{ for segments with } L_{M_B} = 0, \end{aligned}\right\} \text{(IV-8)}$$

where P_B is the base pressure corresponding to the given base altitude H_g. These base pressures can be calculated once the sea-level pressure, P_0, and the temperature profile have been specified.

Having calculated the temperature and pressure, the density, ρ, speed of sound, C_s, and atmospheric viscosity, μ_A, are determined as follows:

$$\left.\begin{aligned} \rho &= \left(\frac{M_0}{R^*}\right)\frac{P}{T_M} \\ C_s &= \left(\frac{\gamma\ R^*}{M_0}\right)^{1/2} T_M^{1/2} \\ \mu_A &= \frac{\beta\ T_M^{3/2}}{T_M + S}, \end{aligned}\right\} \quad \text{(IV-9)}$$

where g_0 is the acceleration of gravity at sea level, M_0 is the molecular weight of air at sea level, R^* is the gas constant, γ is the ratio of specific heats, and β and S are Sutherland's constants.

$$\left.\begin{aligned} M_0 &= 28.9644 \\ R^* &= 8.31432 \cdot 10^3 \frac{J}{(°K)\ (kg\text{-}mol)} \\ \gamma &= 1.40 \\ \beta &= 1.458 \times 10^{-6} \frac{kg}{sec\ m\ (°K)^{1/2}} \\ S &= 110.4°K = 198.72°R \\ g_0 &= 9.80665\ m/sec^2 = 32.174\ ft/sec^2. \end{aligned}\right\} \quad \text{(IV-10)}$$

In the 1962 U.S. standard atmosphere, the molecular weight varies with altitude above approximately 90 km; in POST the molecular weight is assumed constant, resulting in a slight discrepancy above 90 km. In the 1962 U.S. standard atmosphere, geometric altitude is transformed to geopotential altitude, which is used throughout. Thus, above 90 km, a constant slope of molecular scale temperature versus geopotential altitude is used instead of the constant slope of temperature versus geometric altitude.

Table IV-1. - 1962 U. S. Standard Atmosphere Profile

H_B, ft	P_B, psf	T_{M_B}, °R	L_{M_B}, °R/ft
0.0	0.21162166 + 4	518.67	-0.35661600 - 2
36 089.239	0.47268050 + 3	389.97	0.0
65 616.797	0.11434543 + 3	389.97	0.54863995 - 3
104 986.87	0.18128948 + 2	411.57	0.15361920 - 2
154 199.48	0.23163263 + 1	487.17	0.0
170 603.68	0.12322603 + 1	487.17	-0.10972801 - 2
200 131.23	0.38032532 + 0	454.77	-0.21945600 - 2
259 186.35	0.21673064 - 1	325.17	0.0
291 151.57	0.34333824 - 2	325.17	0.16953850 - 2
323 002.74	0.62814785 - 3	379.17	0.28345707 - 2
354 753.59	0.15361733 - 3	469.17	0.56867005 - 2
396 406.39	0.52676024 - 4	649.17	0.11443751 - 1
480 781.04	0.10566108 - 4	1 729.17	0.86358208 - 2
512 046.16	0.77263469 - 5	1 999.17	0.57749093 - 2
543 215.48	0.58405376 - 5	2 179.17	0.40610461 - 2
605 268.45	0.35246030 - 5	2 431.17	0.29274135 - 2
728 243.91	0.14559124 - 5	2 791.17	0.23812804 - 2
939 894.74	0.39418091 - 6	3 295.17	0.20152600 - 2
1 234 645.7	0.84380249 - 7	3 889.17	0.16354849 - 2
1 520 799.4	0.22945543 - 7	4 357.17	0.11010085 - 2
1 798 726.4	0.72259271 - 8	4 663.17	0.73319725 - 3
2 068 776.5	0.24958752 - 8	4 861.17	0.0

1963 Patrick AFB atmosphere using polynomials.- In this model, pressure and temperature are calculated as functions of geometric altitude (h). These parameters are calculated in metric units and converted to English units if required.

Pressure:

1) Altitude region = 0 to 28 000 meters:

$$P = P_1 \exp (A + A_1 h + A_2 h^2 + A_3 h^3 + A_4 h^4 + A_5 h^5)$$

where $P_1 = 10.0$ Newtons/cm^2;

2) Altitude region = 28 000 to 83 004 meters:

$$P = g_0 \times 10^{-4} \exp (A + A_1 h + A_2 h^2 + A_3 h^3 + A_4 h^4 + A_5 h^5);$$

3) Altitude region = 83 004 to 90 000 meters:

$$P = P_B \exp \left(\frac{-1.373301523 \times 10^{12} \quad h - h_B}{T_B \ (6344860 + h) \ (6344860 + h_B)} \right);$$

4) Altitude region = 90 000 to 700 000 meters:

$$L_n (P) = L_n (P_B) + \left(\frac{1.373301523 \times 10^{12}}{L_m \ (6344860 + h) \ (6344860 + h_B)} \right)$$

$$L_n \left(\frac{T_{M_B}}{T_{M_B} + L_m \left(h - h_B\right)} \right). \qquad \text{(IV-11)}$$

Temperature:

1) Altitude region = 0 to 10 832.1 meters:

$$T = T^* = A + A_1 h + A_2 h^2 + A_3 h^3 + A_4 h^4 + A_5 h^5;$$

2) Altitude region = 10 832.1 to 83 004 meters:*

$$T = A + A_1 h + A_2 h^2 + A_3 h^3 + A_4 h^4 + A_5 h^5;$$

3) Altitude region = 83 004 to 90 000 meters:

$$T = T_B + L_k \left(h - h_B\right).$$

However, in this region $L_k = 0$, and thus

$$T = T_B = 180.65°K;$$

4) Altitude region = 90 000 to 700 000 meters:

$$T = T_M = T_{M_B} + L_m \left(h - h_B\right). \qquad \text{(IV-12)}$$

Density:

1) Altitude region = 0 to 28 000 meters:

$$\rho = \rho_1 \exp \left(A + A_1 h + A_2 h^2 + A_3 h^3 + A_4 h^4 + A_5 h^5\right);$$

2) Altitude region = 28 000 to 700 000 meters:

$$\rho = (34.83676) \frac{P}{T}. \qquad \text{(IV-13)}$$

---

*Virtual temperature is the same as kinetic temperature above the 10 832.1-meter altitude.

Table IV-2. - Derived Coefficients for the 1963 Patrick AFB Atmosphere Model

Parameter	Geometric altitude, h, m	Derived coefficient		
		A_0	A_1	A_2
Pressure	0 to 10 832.1	1.6871582×10^{-2}	$-1.1425176 \times 10^{-4}$	$-1.3612327 \times 10^{-9}$
Temperature		299.37265	$-7.7176284 \times 10^{-3}$	9.4867202×10^{-7}
Density		1.3302117×10^{-2}	$-8.8502064 \times 10^{-5}$	$-4.2143056 \times 10^{-9}$
Pressure	10 832.1 to 17 853.3	$-7.9910777 \times 10^{-2}$	$-8.1046438 \times 10^{-5}$	$-5.5522383 \times 10^{-9}$
Temperature		268.92151	4.3075352×10^{-3}	$-8.9159672 \times 10^{-7}$
Density		0.12667122	$-1.3373147 \times 10^{-4}$	2.0667371×10^{-9}
Pressure	17 853.3 to 28 000	0.98414277	$-2.6976917 \times 10^{-4}$	8.5227541×10^{-9}
Temperature		370.64557	$-3.2858965 \times 10^{-2}$	2.0645636×10^{-6}
Density		0.92751266	$-1.4349679 \times 10^{-4}$	$-2.8271736 \times 10^{-9}$
Pressure	28 000 to 49 000	11.4118495	$-4.11497477 \times 10^{-4}$	$1.33664855 \times 10^{-8}$
Temperature		20.44798	$2.07698384 \times 10^{-2}$	$-8.63038789 \times 10^{-7}$
Pressure	49 000 to 83 004	9.99324461	$-2.58298177 \times 10^{-4}$	$3.76139346 \times 10^{-9}$
Temperature		-498.865953	$3.92137281 \times 10^{-2}$	$-4.95180601 \times 10^{-7}$

Table IV-2. - Derived Coefficients for the 1963 Patrick AFB Atmosphere Model

Parameter	Geometric altitude, h, m	Derived coefficient		
		A_3	A_4	A_5
Pressure	0 to 10 832.1	$7.3624145 \times 10^{-14}$	$-1.0800315 \times 10^{-17}$	$3.3046432 \times 10^{-22}$
Temperature		$-1.7136592 \times 10^{-10}$	$1.1074297 \times 10^{-14}$	$-2.3294094 \times 10^{-19}$
Density		$5.9517557 \times 10^{-13}$	$-3.9744789 \times 10^{-17}$	$7.8771273 \times 10^{-22}$
Pressure	10 832.1 to 17 853.3	$3.1116969 \times 10^{-13}$	$-1.6687827 \times 10^{-17}$	$3.8319351 \times 10^{-22}$
Temperature		$-2.8929791 \times 10^{-11}$	$5.0724856 \times 10^{-15}$	$-1.1490372 \times 10^{-19}$
Density		$2.3396109 \times 10^{-13}$	$-3.2562503 \times 10^{-17}$	$7.9035209 \times 10^{-22}$
Pressure	17 853.3 to 28 000	$-3.9620263 \times 10^{-13}$	$1.0146471 \times 10^{-17}$	$-1.0264318 \times 10^{-22}$
Temperature		$-4.3283944 \times 10^{-11}$	$-5.7507242 \times 10^{-17}$	$8.2924583 \times 10^{-21}$
Density		$4.7480092 \times 10^{-14}$	$1.8863246 \times 10^{-18}$	$-4.2702411 \times 10^{-23}$
Pressure	28 000 to 49 000	$-3.59518975 \times 10^{-13}$	$5.10097254 \times 10^{-18}$	$-2.89055894 \times 10^{-23}$
Temperature		$1.66392417 \times 10^{-11}$	$-9.30076185 \times 10^{-17}$	$-4.09005108 \times 10^{-22}$
Pressure	49 000 to 83 004	$-4.20887236 \times 10^{-14}$	$1.60182148 \times 10^{-19}$	$-1.92508927 \times 10^{-25}$
Temperature		$-3.26219854 \times 10^{-12}$	$9.66650364 \times 10^{-17}$	$-4.78844279 \times 10^{-22}$

Table IV-3. - 1963 Patrick AFB Molecular Temperature Profile and Gradient Profile

h_B, km*	T_{M_B}, °K	L_m, °K/km
90	180.65	
		3.0
100	210.65	
		5.0
110	260.65	
		10.0
120	360.65	
		20.0
150	960.65	
		15.0
160	1 110.65	
		10.0
170	1 210.65	
		7.0
190	1 350.65	
		5.0
230	1 550.65	
		4.0
300	1 830.65	
		3.3
400	2 160.65	
		2.6
500	2 420.65	
		1.7
600	2 590.65	
		1.1
700	2 700.65	

*Altitude range: 90 000 to 700 000 meters.

Pressure and density ratios:

Altitude region = 0 to 700 000 meters:

$$\left.\begin{aligned} \rho_R &= \frac{\rho}{\rho_o} \\ P_R &= \frac{P}{P_o}. \end{aligned}\right\} \qquad \text{(IV-14)}$$

Velocity of sound:

$$V_S = (20.046707)\ (T)^{\frac{1}{2}}. \qquad \text{(IV-15)}$$

The atmosphere model-derived coefficients are presented in table IV-2. The molecular temperature gradient is documented in table IV-3 for geometric altitudes from 90 to 700 km.

Winds

The atmospheric wind velocity components are input in tables using either meteorological or vector notation. If these tables, which are normally functions of oblate altitude, are not input, then the atmosphere is assumed to rotate uniformly with the planet.

The wind velocity components can be input directly in the geographic frame by defining u_W, v_W, and w_W, or by defining the wind speed $\left(V_W\right)$, the vertical component $\left(W_W\right)$, the wind azimuth $\left(A_{ZW}\right)$, and the wind azimuth bias $\left(A_{ZWB}\right)$. The resulting wind velocity components in the G-frame are:

$$\underline{V}_{WG} = \begin{bmatrix} V_W\ (h)\ \cos\left(A_{ZW}\ (h) + A_{ZWB}\right) \\ V_W\ (h)\ \sin\left(A_{ZW}\ (h) + A_{ZWB}\right) \\ w_W\ (h) \end{bmatrix} \qquad \text{(IV-16)}$$

It is clear from the above equation that in order to input vector wind data A_{ZWB} must be input as zero, whereas for meteorologic data the preloaded value of 180° should be used.

The wind velocity in the ECI frame is then given by

$$\underline{V}_{WI} = [IG]^{-1} \underline{V}_{WG} \qquad (IV\text{-}17)$$

Thus, the atmospheric relative velocity vector in the ECI frame is

$$\underline{V}_{AI} = \underline{V}_{I} - \underline{\Omega}_{p} \times \underline{r}_{I} - \underline{V}_{WI} \qquad (IV\text{-}18)$$

and its magnitude is given by

$$V_{A} = \sqrt{\underline{V}_{AI} \cdot \underline{V}_{AI}} \qquad (IV\text{-}19)$$

V. VEHICLE MODEL

The various physical properties of the vehicle are modeled by the user when he selects the pertinent options from the set of vehicle simulation modules. The equations used in these modules are presented below.

Mass Properties Model

The gross weight of the vehicle at the beginning of each phase is given by

$$W_G = W_{stg} + W_{pld}, \tag{V-1}$$

where W_{stg} is gross weight without payload and W_{pld} is the payload weight. For phases other than the first, the gross weight can optionally be computed as

$$W_G^+ = W_G^- - W_{jett} - W_{PR}, \tag{V-2}$$

where W_G^+ is the gross weight on the positive side of the current event, W_G^- is the gross weight on the negative side of the current event, W_{jett} is the jettison weight, and W_{PR} is the weight of propellant remaining. These options are obtained automatically, based on user input.

The propellant remaining is given by

$$W_{PR} = W_{P_i} - W_{PC}, \tag{V-3}$$

where W_{P_i} is the initial weight of propellant and W_{PC} is the amount of propellant consumed. This latter term is given by

$$W_{PC} = \int \dot{W}\, dt + W_{C_0} \tag{V-4}$$

where $\dot{W}$ is the total rate of change of the vehicle's weight.

At the beginning of each phase, the constant W_{C_O} can be either input or carried across the event as the total amount of weight consumed in the previous phase.

The amount of propellant jettisoned can be computed as:

1) The amount of propellant remaining at the beginning of the current phase,

2) The amount of propellant remaining at some prescribed prior event.

The constant jettison weight is either computed from an input constant value or determined from an input mass-fraction table. When a mass-fraction table is used the jettison weight is given by

$$W_{jett} = W_{P_i} \left[\frac{1}{\lambda} - 1\right], \qquad \text{(V-5)}$$

where λ is the mass fraction computed from the table.

Propulsion Calculations

POST can simulate both rocket and jet engines. The program can simulate up to 15 engines in either mode.

<u>Rocket engines</u>.- There are two input options for engine data in the rocket mode. In the first option, tables for vacuum thrust and maximum weight flowrate are input for each engine. In the second option, tables for vacuum thrust are input, along with the vacuum specific impulse for each engine. The vacuum specific impulse is then used to calculate the mass flowrate.

The rocket thrust per engine is given by

$$T_{R_i} = \eta\, T_{vac_i} - A_{E_i}\, p(h), \qquad \text{(V-6)}$$

where η is the throttle setting, T_{vac_i} is the vacuum thrust of the ith engine, A_E is the nozzle exit area, and p(h) is the atmospheric pressure. Summing over all engines yields the total rocket thrust

$$T_R = \sum_{i=1}^{N_{eng}} T_{R_i}, \tag{V-7}$$

where N_{eng} is the number of thrusting engines, and $N_{eng} \leq 15$.

The weight flowrate in the rocket mode is given by

$$W = \begin{cases} -\eta \sum_{i=1}^{N_{eng}} \left(\dot{W}_p^{max}{}_i\right) \\ -\eta \sum_{i=1}^{N_{eng}} \left(T_{vac}/I_{sp_{vac}}\right)_i \end{cases} \tag{V-3}$$

Jet engines.- In the jet engine mode the net jet thrust per engine is given by

$$\left(\frac{T_J}{\delta}\right)_i = f(M, \eta), \tag{V-9}$$

where

$$\delta = p(h)/p_{SL} \tag{V-10}$$

and $\frac{T_J}{\delta}$ (M, a monovariant table. The total jet thrust is then given by

$$T_J = \eta \sum_{i=1}^{N_{eng}} p(h) / (P_{SL}) \; (T_J/\delta_i). \qquad \text{(V-11)}$$

The weight flowrate in the jet engine mode is

$$\dot{W} = - \sum_{i=1}^{N_{eng}} \sqrt{\frac{T(h)}{T_{SL}}} \left(\frac{p(h)}{P_{SL}}\right) \left(\frac{SFC}{\sqrt{\theta}}\right)_i \left(\frac{T_J}{\delta}\right)_i \delta . \qquad \text{(V-12)}$$

The thrust vector components for both rocket and jet engines are determined from the thrust magnitude T_{R_i} or T_{J_i} and the thrust incidence angles i_{p_i} and i_{y_i}. The thrust accelerations in the body axes are then given by

$$\underline{A}_{TB} = \sum_{i=1}^{N_{eng}} \eta_i \frac{T_i}{m} \begin{bmatrix} \cos i_{y_i} \cos i_{p_i} \\ \sin i_{y_i} \\ \cos i_{y_i} \sin i_{p_i} \end{bmatrix} = \frac{\underline{T}_R}{m} \qquad \text{(V-13)}$$

In the above equation T_i denotes the thrust magnitude for the i^{th} engine (either rocket or jet) and m denotes the instantaneous mass of the vehicle. The engine gimbal angles are determined from the static trim equations in the moment balance option or by input if the moment balance option is not used. The engine gimbal angles are illustrated in figure V-1.

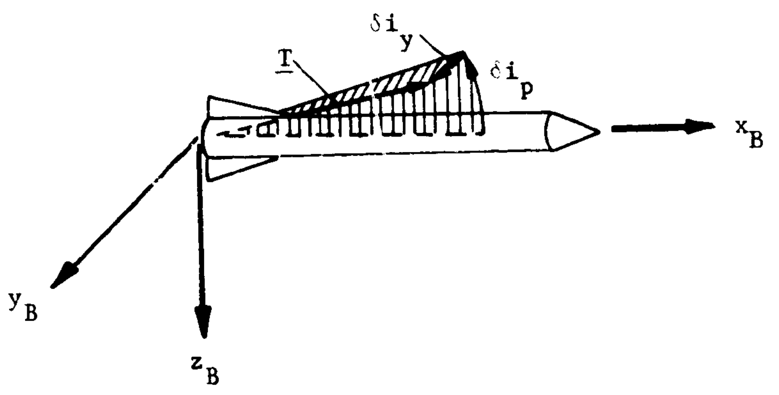


Figure V-1.- Engine Gimbal Angles

Note that thrust misalignments can be simulated by inputting the engine gimbal angles and using the standard three-degree-of-freedom option.

Aerodynamic Calculations

The aerodynamic force coefficients can be expressed in terms of the lift, drag, and side-force coefficients C_L, C_D, and C_Y (fig. V-2), where C_L and C_D are directed normal to, and along the velocity projection in the x_B-z_B plane. Note that C_Y produces a side-force, $F_{A_{YB}}$, acting in the direction of y_B.

Lift and drag force coefficients are transformed to axial and normal force coefficients as follows:

$$\begin{bmatrix} C_A \\ C_N \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} C_D \\ C_L \end{bmatrix} \qquad \text{(V-14)}$$

where α is the angle-of-attack.

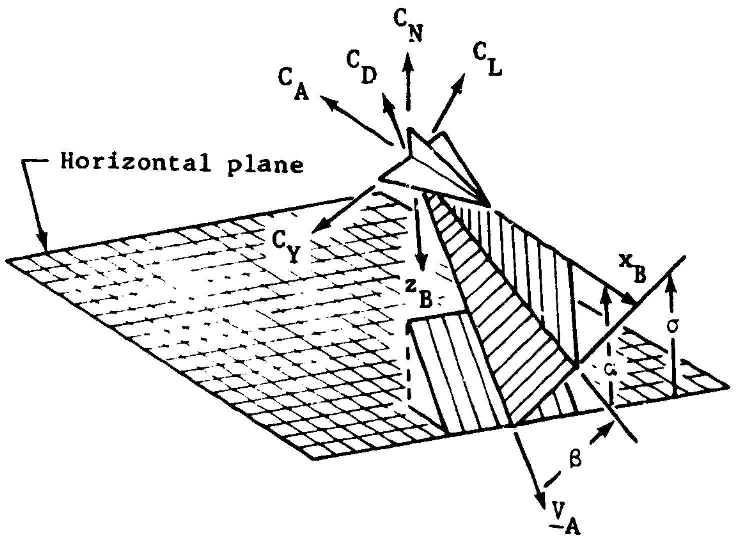


Figure V-2.- Aerodynamic Angles

The aerodynamic coefficients can also be expressed in terms of the axial force, normal force, and side force, C_A, C_N, and C_Y, respectively. Here C_A and C_N produce forces that act in the $-x_B$ and $-z_B$ directions, and C_Y produces a force acting along y_B.

Each aerodynamic coefficient is computed by interpolating the values in the table. In general, eight tables are allocated to each coefficient. These tables can be monovariant, bivariant, or trivariant, and seven tables per coefficient can have arbitrary hollerith mnemonic multipliers. This generality enables all standard forms of aerodynamic data to be directly input into the program.

The aerodynamic force coefficients are given by:

$$\left.\begin{aligned}
C_D &= C_{D_O} + C_D M_{C_D} + C_{D_{\delta p}} M_{C_{D_{\delta p}}} + C_{D_{\delta y}} M_{C_{D_{\delta y}}} \\
C_L &= C_{L_O} + C_{L_O} M_{C_{L_O}} + C_{L_{\delta p}} M_{C_{L_{\delta p}}} \\
C_A &= C_{A_O} + C_A M_{C_A} + C_{A_{\delta p}} M_{C_{A_{\delta p}}} + C_{A_{\delta y}} M_{C_{A_{\delta y}}} \\
C_N &= C_{N_O} + C_N M_{C_N} + C_{N_{\delta p}} M_{C_{N_{\delta p}}} \\
C_Y &= C_{Y_O} + C_Y M_{C_Y} + C_{Y_{\delta y}} M_{C_{Y_{\delta y}}} \quad ;
\end{aligned}\right\} \qquad \text{(V-15)}$$

the aerodynamic moment coefficients are given by:

$$\left.\begin{aligned}
C_M &= C_{M_O} + C_M M_{C_M} + C_{M_{\delta p}} M_{C_{M_{\delta p}}} \\
C_n &= C_{n_O} + C_n M_{C_M} + C_{n_{\delta y}} M_{C_{n_{\delta y}}} \quad .
\end{aligned}\right\} \qquad \text{(V-16)}$$

In the above expressions, C_{D_O}, C_D, $C_{D_{\delta p}}$, $C_{D_{\delta y}}$, etc, denote the tables, and $M_{C_{D_O}}$, M_{C_D}, $M_{C_{D_{\delta p}}}$, $M_{C_{D_{\delta y}}}$, etc, denote the mnemonic table multipliers. Typical table arguments and multipliers would be α, β, M, R_N, δp, and δy.

The Mach number and dynamic pressure are given by:

$$\left.\begin{aligned}
M &= \frac{V_A}{C_S} \\
q &= \frac{1}{2}\,\rho\, V_A^2,
\end{aligned}\right\} \qquad \text{(V-17)}$$

where ρ is the atmospheric density, V_A is the velocity of the vehicle with respect to the atmosphere, and C_S is the speed of sound. These atmospheric parameters are determined from the atmospheric models as a function of the altitude h above the oblate spheroid; i.e.,

$$\left.\begin{aligned} \rho &= \rho(h) \\ C_S &= C_S(h) \\ p &= p(h) \\ T &= T(h). \end{aligned}\right\} \qquad \text{(V-18)}$$

The angle of attack in pitch (α) and the angle of sideslip (β) required to determine the aerodynamic coefficients are calculated as follows:

$$\left.\begin{aligned} \alpha &= \tan^{-1}\left[\frac{\sin\alpha}{\cos\alpha}\right] \\ \beta &= \tan^{-1}\left[\frac{\sin\beta}{\cos\beta}\right], \\ \sin\alpha &= \frac{w_B}{\sqrt{u_B^2 + w_B^2}} \\ \cos\alpha &= \frac{u_B}{\sqrt{u_B^2 + w_B^2}} \\ \sin\beta &= \frac{v_B}{V_A} \\ \cos\beta &= \frac{\sqrt{u_B^2 + w_B^2}}{V_A} \end{aligned}\right\} \qquad \text{(V-19)}$$

The total angle of attack is

$$\alpha_T = \cos\left(V_{AB_x}/V_A\right). \qquad \text{(V-20)}$$

The aerodynamic forces in the body frame are

$$\underline{F}_{AB} = q\ S\begin{bmatrix} -C_A \\ C_Y \\ -C_N \end{bmatrix}, \qquad \text{(V-21)}$$

where q is the dynamic pressure and S is the reference area.

The resulting accelerations in the body system are then obtained from

$$\underline{A}_{AB} = \frac{1}{m} \underline{F}_{AB} \; . \qquad \text{(V-22)}$$

Aeroheating Calculations

POST provides for a wide variety of aeroheating calculations. Some of these options are specific in nature and apply only to particular vehicles, whereas others are quite general. The general heat rate option is based on trivariant table interpolation and provides complete flexibility with regards to vehicle shape and heat-transfer methodology. The various heat rate equations are described below.

Heat rate equations.-

1) Chapman's equations. In this calculation the heat rate is given by

$$\dot{Q} = \frac{17\ 600}{\sqrt{R_N}} \left(\frac{\rho}{\rho_{SL}}\right)^{\frac{1}{2}} \left(\frac{V_R}{V_C}\right)^{3.15} \tag{V-23}$$

where R_N is the nose radius, ρ is the atmospheric density, and V_C is the reference circular orbital velocity.

2) General table lookup. This heat rate is given by

$$\dot{Q} = Q_t\ (x_1,\ x_2,\ x_3), \tag{V-24}$$

where x_1, x_2, and x_3 can be any internally computed variables. For example, the values that would normally be selected are $x_1 = \alpha$, $x_2 = h$, and $x_3 = V_R$.

3) Modified Chapman's equation. Here the heat rate is given by

$$\dot{Q} = Q_t\ (x_1,\ x_2,\ x_3)\ Q_c, \tag{V-25}$$

where Q_t is an arbitary table and $\dot{Q}_c$ is the standard Chapman's equation.

4) Turbulent-flow heat rate. The turbulent-flow heat rate is given by

$$\dot{Q} = Q_t\ (x_1,\ x_2,\ x_3) \left[1500 \left(\frac{\rho}{\rho_{SL}}\right)^{0.8} \left(\frac{V_A}{10^4}\right)^{3.18}\right]. \tag{V-26}$$

5) Maximum centerline heating. The equations for this method are given below in sequence.

a) Altitude-velocity correction:

$$\left.\begin{aligned} \Delta h = 10^5 \Big[& 1.06112 - 6.16586\ V_A/10^4 \\ & + 51.12090 \left(V_A/10^4\right)^4 - 20.66258\left(V_A/10^4\right)^5\Big] \\ & + 22.52598 \left(V_A/10^4\right)^2 - 48.28080\left(V_A/10^4\right)^3 \\ h_{ref} = h + \Delta h. & \end{aligned}\right\} \tag{V-27}$$

b) Maximum centerline heat rate at reference conditions:

$$\left.\begin{array}{l}
\text{-- if } h_{ref} \geq 103\,600 \text{ m:} \\
\dot{q}_{ref} = 10^2\left[277.93332 + 134.55760\, h_{ref}/10^5 - 807.75941\left(h_{ref}/10^5\right)^2 \right. \\
\quad + 2.90536\left(h_{ref}/10^5\right)^3 + 722.36896\left(h_{ref}/10^5\right)^4 - 311.40176 \\
\quad \left. \left(h_{ref}/10^5\right)^5\right]; \\
\text{-- if } h_{ref} \leq 103\,600 \text{ m;} \\
\dot{q}_{ref} = 10^4\left[7115.39692 - 34\,881.13588\, h_{ref}/10^5 + 69\,844.23141 \right. \\
\quad \left(h_{ref}/10^5\right)^2 \\
\quad - 71\,534.98453\left(h_{ref}/10^5\right)^3 + 37\,506.13054\left(h_{ref}/10^5\right)^4 - \\
\quad \left. - 8048.55112\left(h_{ref}/10^5\right)^5\right].
\end{array}\right\} \quad \text{(V-28)}$$

c) Angle of attack correction:

$$\dot{q}_{max,\alpha}/\dot{q}_{max,\alpha=50^\circ} = [\ell n\ (x)]^2,$$

where

$$x = 10^2\ [0.01136 + 0.01343\ \alpha/10^2 + 1.42672\ (\alpha/10^2)^4 - 0.75623 \quad \text{(V-29)}$$
$$(\alpha/10^2)^5] + 0.30535\ (\alpha/10^2)^2 - 1.06269\ (\alpha/10^2)^3.$$

d) Maximum centerline heat rate:

$$\dot{q}_{max} = \left(\dot{q}_{max,\alpha}\right)\Big/\left(\dot{q}_{max,\alpha=50^\circ}\right)\ \left(\dot{q}_{ref}\right). \quad \text{(V-30)}$$

In addition to the heat rate calculations, the program also provides the capability to calculate other aeroheating indicators that can be used for trajectory shaping purposes.

Aerodynamic heating indicators.- The heating rate for zero total angle of attack α_T is

$$\dot{Q} = q\ V_A. \quad \text{(V-31)}$$

The aerodynamic heating indicator for zero total angle of attack is

$$Q = \int_0^t \dot{Q}\, dt. \tag{V-32}$$

The heating indicator for non-zero angles of attack is given by

$$Q' = \int_0^t f\,(\alpha', M)\, \dot{Q}\, dt, \tag{V-33}$$

where

$$f\,(\alpha', M) = \left(1 + \frac{7}{5} M^2 \sin^2 \alpha'\right)^{5/7} K,$$

$$K = \left\{1 + \frac{5}{M^2}\left[1 - \left(1 + \frac{7}{5} M^2 \sin^2 \alpha'\right)^{2/7}\right]\right\}^{1/2}$$

and

$$\left.\begin{array}{l} \left.\begin{array}{ll} \alpha' & = \alpha \\ Q_T' & = Q' \end{array}\right\} \text{for } \alpha < 0° \\ \left.\begin{array}{ll} \alpha' & = \alpha \\ Q_B' & = Q' \end{array}\right\} \text{for } \alpha > 0° \\ \left.\begin{array}{ll} \alpha' & = \beta \\ Q_L' & = Q' \end{array}\right\} \text{for } \beta < 0° \\ \left.\begin{array}{ll} \alpha' & = \beta \\ Q_R' & = Q' \end{array}\right\} \text{for } \beta > 0°. \end{array}\right\} \tag{V-34}$$

The heating indicator for laminar flow is calculated as

$$Q_{lam} = \int_0^t 17\,600\, K_{\alpha_T} \left(\frac{\rho}{\rho_0}\right)^{\frac{1}{2}} \left(\frac{V_A}{26\,000}\right)^{3.15} dt, \tag{V-35}$$

where

$$K_{\alpha_T} = f\,(\alpha_T). \tag{V-36}$$

The heating indicator for turbulent flow is calculated as

$$Q_{turb} = \int 1500\, K_{\alpha_T} \left(\frac{\rho}{\rho_0}\right)^{0.8} \left(\frac{V_A}{10\,000}\right)^{3.18} dt \tag{V-37}$$

Ten-Panel Vehicle Heating Model.- Special aeroheating calculations are available for a ten-panel vehicle model. The heating ratios are referenced to the heat rate calculation. The total heat for each panel is given by

$$Q_i = H_{R_i}\, Q, \tag{V-38}$$

where Q is the total heat and H_{R_i} is the heat ratio for panel i. The weight for each panel is the product of the weight per unit area and the area of the panel. The total weight is the sum of the individual weights for each panel:

$$W_P = \sum_{i=1}^{10} W_{uA_i}\, A_i \tag{V-39}$$

where W_{uA_i} is the weight per unit area and A_i is the area of the ith panel.

Steering Model

The steering options control the attitude of the vehicle during the trajectory simulation. The general types of steering options available are:

1) Body rates:

2) Aerodynamic angles;

3) Inertial Euler angles;

4) Relative Euler angles.

The body rates are generally used to simulate strapdown-type systems and are computed from user-specified rate polynominals. The aerodynamic angles are generally used for reentry problems, and the inertial and relative Euler angles are usually used to simulate vehicles that employ inertial or local horizontal reference systems. All of these angles can be computed from: (1) polynominals; (2) tables; (3) piecewise linear functions; or (4) closed-loop linear feedback systems.

The functional relationship used to compute the steering commands suggest two natural steering classifications:

1) Rate steering;

2) Angular steering.

These classifications provide an efficient outline for presenting the equations used to compute the steering commands.

<u>Rate steering</u>.- Rate steering uses the body rates, in conjunction with the quaternion equations, to determine the attitude of the vehicle. When using this option the user must specify:

1) The initial attitude of the vehicle;

2) The polynominal option used to compute the body rates.

The initial attitude is used internally to initialize the quaternion rate equation

$$\underline{\dot{e}} = \frac{1}{2} [E] \, \underline{\omega} \qquad \text{(V-40)}$$

where

$$\underline{e} = (e_0, e_1, e_2, e_3),$$

$$\underline{\omega} = (\omega_x, \omega_y, \omega_z),$$

$$[E] = \begin{bmatrix} -e_1 & e_2 & e_3 \\ e_0 & e_2 & -e_3 \\ e_0 & -e_1 & e_3 \\ e_0 & e_1 & -e_2 \end{bmatrix} \qquad \text{(V-41)}$$

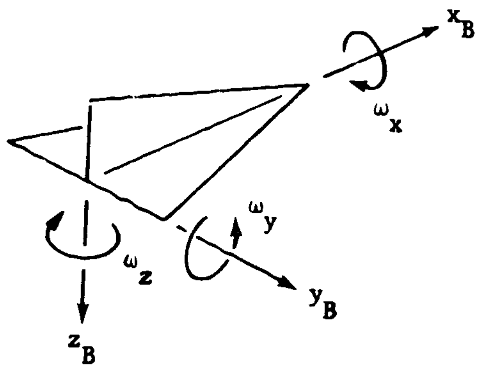


Figure V-3.- Body Rates

There are two options available for initializing the quaternion elements: (1) inertial Euler angles, and (2) aerodynamic angles. When inertial Euler angles are input the initial quaternion vector is given by

$$\underline{e}_0 = \underline{e}\left(\phi_I\right) * \underline{e}\left(\psi_I\right) * \underline{e}\left(\theta_I\right) \qquad \text{(V-42)}$$

where the asterisk denotes quaternion multiplication and where

$$\underline{e}\left(\phi_I\right) = \cos\left(0.5\ \phi_I\right) + \sin\left(0.5\ \phi_I\right)\ i$$

$$\underline{e}\left(\psi_I\right) = \cos\left(0.5\ \psi_I\right) + \sin\left(0.5\ \psi_I\right)\ k \qquad \text{(V-43)}$$

$$\underline{e}\left(\theta_I\right) = \cos\left(0.5\ \theta_I\right) + \sin\left(0.5\ \theta_I\right)\ j.$$

When aerodynamic angles are input, then the initial quaternion vector is given by

$$\underline{e}_0 = \underline{e}\left(A_{ZL}\right) * \underline{e}(90) * \underline{e}\left(\phi_L\right) * \underline{e}\left(-\theta_L\right) * \underline{e}\left(\theta_I\right) * \underline{e}\left(-\phi_c\right) * \underline{e}(-90) *$$

$$\underline{e}\left(\lambda_A\right) * \underline{e}\left(\gamma_A\right) * \underline{e}(\phi) * \underline{e}(-\beta) * \underline{e}(\alpha), \qquad \text{(V-44)}$$

where

$$\left.\begin{aligned}
\underline{e}\left(A_{ZL}\right) &= \cos\left(0.5\ A_{ZL}\right) - \sin\left(0.5\ A_{ZL}\right)\ k \\
\underline{e}(90) &= \cos(45) + \sin(45)\ j \\
\underline{e}\left(\phi_L\right) &= \cos\left(0.5\ \phi_L\right) + \sin\left(0.5\ \phi_L\right)\ j \\
\underline{e}\left(-\theta_L\right) &= \cos\left(0.5\ \theta_L\right) - \sin\left(0.5\ \theta_L\right)\ k \\
\underline{e}\left(\theta_I\right) &= \cos\left(0.5\ \theta_I\right) + \sin\left(0.5\ \theta_I\right)\ k \\
\underline{e}\left(\phi_c\right) &= \cos\left(0.5\ \phi_c\right) - \sin\left(0.5\ \phi_c\right)\ j \\
\underline{e}(\sigma) &= \cos\left(0.5\sigma\right) + (\sin 0.5\sigma)\ i \\
\underline{e}(-\beta) &= \cos\left(0.5\beta\right) - (\sin 0.5\beta)\ k \\
\underline{e}(\alpha) &= \cos\left(0.5\alpha\right) + (\sin 0.5\alpha)\ j.
\end{aligned}\right\} \qquad \text{(V-45)}$$

The user must also select the option for computing the body rates. These options are combinations of the basic rate polynominals shown at the top of next page:

$$\left.\begin{aligned}
\omega_x &= \sum_{i=0}^{2} a_i x^i \\
\omega_y &= \sum_{i=0}^{2} b_i y^i \\
\omega_z &= \sum_{k=0}^{2} c_i z^i \\
\dot{\alpha} &= \sum_{i=0}^{2} a_i x^i \\
\dot{\beta} &= \sum_{i=0}^{2} b_i y^i \\
\dot{\sigma} &= \sum_{i=0}^{2} c_i z^i ,
\end{aligned}\right\} \qquad \text{(V-46)}$$

where a_i, b_i, and c_i are the polynomial coefficients, and x, y, and z are the polynomial arguments.

The available combinations of these basic rate polynomials are:

1) Input the coefficients and the arguments of ω_x, ω_y, ω_z;

2) Input the coefficients and the arguments of α, β, σ; and calculate ω_x, ω_y, and ω_z via

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} a_1 + d_1\sigma + (\sin\alpha)\,\dot{\beta} \\ a_2 + d_2\sigma + \dot{\alpha} \\ a_3 + d_3\sigma - (\cos\alpha)\,\dot{\beta} \end{bmatrix}; \tag{V-47}$$

3) Input the coefficients of the $\dot{\sigma}$, ω_y, and ω_z polynomials and calculate ω_x via

$$\omega_x = a_1 + d_1\dot{\sigma} + \tan\alpha\left(a_3 + d_3\dot{\sigma} - \omega_z\right); \tag{V-48}$$

4) Input the coefficients of the ω_x, $\dot{\alpha}$, and ω_z polynomials and calculate ω_y via

$$\omega_y = a_2 + \dot{\alpha} + d_2\left[\frac{\omega_z - a_1 + \left(\omega_z - a_3\right)\tan\alpha}{d_1 + d_3\tan\alpha}\right]; \tag{V-49}$$

5) Input the coefficients of the ω_x, ω_y, and $\dot{\beta}$ polynomials and calculate ω_z via

$$\omega_z = a_3 - (\cos\alpha)\,\dot{\beta} + \frac{d_3}{d_1}\left[\omega_x - a_1 - (\sin\alpha)\,\dot{\beta}\right]; \tag{V-50}$$

6) Input the coefficients of the $\dot{\sigma}$, $\dot{\alpha}$, and ω_z polynomials and calculate ω_x and ω_y via

$$\begin{aligned} \omega_x &= a_1 + d_1\dot{\sigma} + \tan\alpha\left(a_3 + d_3\dot{\sigma} - \omega_z\right) \\ \omega_z &= a_2 + d_2\dot{\sigma} + \dot{\alpha}\ ; \end{aligned} \tag{V-51}$$

7) Input the coefficients of the ω_x, α, and $\dot{\beta}$ polynomials and calculate ω_y and ω_z via

$$\left.\begin{aligned} \omega_y &= a_2 + \dot{\alpha} + \frac{d_2}{d_1}\left(\omega_x - a_1 - (\sin \alpha)\ \beta\right) \\ \omega_z &= a_3 - \cos \alpha\ \dot{\beta} + \frac{d_3}{d_1}\left(\omega_x - a_1\ (\sin \alpha)\ \dot{\beta}\right)\ ; \end{aligned}\right\} \qquad \text{(V-52)}$$

8) Input the coefficients of the $\dot{\sigma}$, ω_y, and $\dot{\beta}$ polynomials and calculate ω_x and ω_z via

$$\left.\begin{aligned} \omega_x &= a_1 + d_1\dot{\sigma} + (\sin \alpha)\ \dot{\beta} \\ \omega_x &= a_3 + d_3\dot{\sigma} - (\cos \alpha)\ \dot{\beta}; \end{aligned}\right\} \qquad \text{(V-53)}$$

9) Input the coefficients of the $\dot{\psi}_R$, θ_R, ϕ_R polynomials and and calculate ω_x, ω_y, and ω_z via

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \underline{a} + \begin{bmatrix} \dot{\phi}_R - \sin \theta_R\ \dot{\psi}_R \\ \cos \phi_R\ \dot{\theta}_R + \sin \phi_R \cos \theta_R\ \dot{\psi}_R \\ \cos \phi_R \cos \theta_R\ \dot{\psi}_R - \sin \phi_R\ \dot{\theta}_R \end{bmatrix} \qquad \text{(V-54)}$$

where

$$\underline{a} = [GB] \begin{bmatrix} \frac{v}{r} \\ \frac{-u}{r} \\ \frac{-v}{r} \tan \phi_c \end{bmatrix} \qquad \text{(V-55)}$$

The variables used in these equations are:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = [IG]\,\underline{V}_I, \tag{V-56}$$

$$\begin{bmatrix} u_A \\ v_A \\ w_A \end{bmatrix} = [IG]\,\underline{V}_{AI}, \tag{V-57}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = [GB] \begin{bmatrix} \frac{v}{r_I} - \dot{\gamma}_A \sin\lambda_A \\ \frac{-u}{r_I} + \dot{\gamma}_A \cos\lambda_A \\ \frac{-v \tan\phi_c}{r_I} + \dot{\lambda}_A \end{bmatrix} \tag{V-58}$$

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = [GB] \begin{bmatrix} \cos\lambda_A \cos\gamma_A \\ \sin\lambda_A \cos\gamma_A \\ -\sin\gamma_A \end{bmatrix} \tag{V-59}$$

where

$$\dot{\lambda}_A = \frac{u_A \dot{v}_A - v_A \dot{u}_A}{u_A^2 + v_A^2} \tag{V-60}$$

$$\dot{\gamma}_A = \frac{-V_A \dot{w}_A + w_A \dot{V}_A}{V_A \sqrt{V_A^2 - w_A^2}}$$

and

$$\left.\begin{aligned}
\dot{u}_A &= \dot{u} + \frac{du_w}{dh}\dot{h} \\
\dot{v}_A &= \dot{v} - \Omega_p\left(\dot{r}\cos\phi_c + r\,\dot{\phi}_c\sin\phi_c\right) + \frac{dv_w}{dh}\dot{h} \\
\dot{w}_A &= w + \frac{dw_w}{dh}\dot{h} \\
\dot{V}_A &= \frac{1}{V_A}\left[u_A\dot{u}_A + v_A\dot{v}_A + w_A\dot{w}_A\right] \\
\dot{h} &= -w - \frac{dR_s}{d\phi_c}\dot{\phi}_c \\
\frac{dR_s}{d\phi_c} &= \frac{-R_s^3}{R_E^2}\left[\left(\frac{R_E}{R_p}\right)^2 - 1\right]\sin\phi_c\cos\phi_c
\end{aligned}\right\} \qquad \text{(V-61)}$$

Angular steering.- Angular steering uses four different functional relationships to determine the attitude of the vehicle. These functional steering equations are:

1) Cubic polynomials;

2) Tables;

3) Piecewise linear equations;

4) Closed-loop linear feedback systems.

Each of these steering techniques is available for all of the steering angles; i.e., aerodynamic angles, inertial Euler angles, and relative Euler angles. There is also a separate channel steering capability. This option enables the user to specify different functional relationships in each triplet of steering angles. This means, for example, that the angle of attack could be computed from a polynomial, the bank angle from a table, and the sideslip angle by yet another method. However, this option does not allow the user to mix the steering triplets (i.e., mix the aerodynamic angles with the Euler angles).

In the following discussion, it is convenient to let θ denote an arbitrary steering angle. The steering equations described in terms of this variable then apply to all of the steering angles.

Polynomial steering: Under this option the steering angles are computed from a cubic polynomial

$$\theta(y) = \sum_{i=0}^{3} c_i y^i, \tag{V-62}$$

where the user selects the coefficient c_i and the independent variable y. The highest-order coefficient that is input determines the degree of the polynomial. The argument can be selected as any internally computed variable; e.g., time, velocity, altitude, etc. The constant term of the polynomial, c_0, can be either input at the beginning of each phase or carried across as the value of the angle at the end of the previous phase. The polynomial coefficients are generally used as the independent variables for targeting/optimization.

Table steering: Under this option the steering angles are computed via table interpolation, which is denoted by

$$\theta(\underline{y}) = \theta_m \, T^n[\underline{f}(\underline{y})]. \tag{V-63}$$

The user initially inputs the table multiplier θ_m, the order or interpolation n, and the table data $(\underline{y}, \underline{f}(\underline{y}))$. The table multiplier or the dependent table values can be used as independent variables for targeting or optimization. The order of interpolation can either be linear or cubic. The tables can be monovariant, bivariant, or trivariant functions of the table arguments.

Piecewise linear steering: Under this option the steering angles are computed from a general piecewise linear function of the form

$$\theta(y) = c_1 + \left[\frac{c_2 - c_1}{y_2 - y_1}\right]\left(y - y_1\right), \tag{V-64}$$

where c_1 is equal to θ at the beginning of the current phase, c_2 is the desired value of θ at a designated later event, y is equal to the value of the designated event criterion at the beginning of the current phase, y_2 is the desired value of the designated event criterion at the designated event, and y is the current value of the designated event criterion.

This option is simular to the polynomial option except that the values of θ are specified directly rather than as θ_0, $\dot{\theta}$, $\ddot{\theta}$ and $\dddot{\theta}$. Clearly, θ is linear in time if $y = t$; otherwise θ is only linear in y. When the desired values of the steering angles are used as independent variables, the problem of cascaded steering effects is avoided and the targeting/optimization algorithm generally converges faster. This option also automatically computes the steering angle rates required to change the attitude to the desired value at the designated event, which reduces the problems related to guessing accurate initial pitch rates.

Linear feedback steering: Under this option the steering angles are computed from the linear error-error rate feedback control law

$$\theta = c_1 + K_D\left(F_a - F_d\right) + K_R\left(F_a' - F_d'\right), \tag{V-65}$$

where c_1 is a nominal steering angle profile, K_D is the displacement error gain, K_R is the rate gain, $F_a - F_d$ is the error in the steering function F, and $F_a' - D_d'$ is the derivative of the steering function error.

This option is, of course, the classic path control law, and enables the user to steer to a wide variecy of trajectory profiles, such as velocity vs altitude profile, acceleration vs altitude profile, etc. This option is particularly useful for reentry trajectory shaping.

Generalized Acceleration Steering. - Under this option the steering angles are computed by solving a set of user specified equations. The dependent and independent variables in these equations must be selected from the dictionary of variables already computed in POST. The only restriction is that these equations must be explicityly a function of some derivative computed in the inner loop of the program. As a consequence, this option cannot be used to solve equations that are functions of integrals of the equations of motion. For example, this option cannot be used to maintain constant altitude by zeroing $\dot{h}$. This is because the time derivative of altitude is computed from velocity, and velocity is computed from the integral of acceleration. The linear feedback model should be used to solve problems involving integrals of the equations of motion.

In more precise terms, the steering variables are determined from the iterative solution of the problem:

> For each instant of time, determine the values of the steering variables, $\underline{\theta}$, that satisfy the steering equations,

$$\underline{e}(\underline{\theta}) = \underline{y}(\underline{\theta}) - \underline{y}_d = \underline{0}, \qquad \text{(V-66)}$$

where $\underline{y}$ is a n-component vector of dependent variables, $\underline{y}_d$ is the desired value of these variables and $\underline{e}$ the error in dependent variables.

Typical applications of this option are given as

1) Control normal acceleration to 1g and axial acceleration to 3g by calculating the angle of attack and throttle setting that solves the equations

$$\begin{aligned} A_{xB}(\alpha, \eta) - 96.6 &= 0 \\ A_{zB}(\alpha, \eta) - 32.2 &= 0 \end{aligned} \qquad \text{(V-67)}$$

2) Obtain level unaccelerated flight by solving the equations

$$\begin{aligned} \dot{V}_A(\alpha, \eta) &= 0 \\ \dot{\gamma}_A(\alpha, \eta) &= 0 \;. \end{aligned} \qquad \text{(V-68)}$$

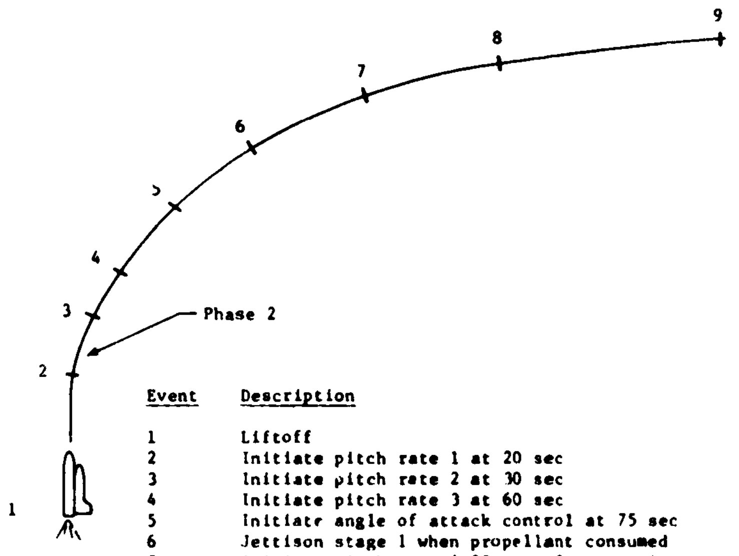
Level unaccelerated flight is implicitly acieved in example 2 because $\dot{V}_A = 0$ implies constant velocity, and $\dot{\gamma}_A = 0$ implies constant altitude (that is, if $\gamma_A = 0$ when this option is initiated).

VI. TRAJECTORY SIMULATION

The following sections present the equations used in the trajectory simulation subroutines. These equations summarize the principal computations performed by the program, and motivate many of the program input procedures.

Events/Phases

Simulation data are input according to phase, where the phases are defined by a user-specified sequence of events. The simulation equations are then solved sequentially by phase. Therefore, the user is required to input a sequence of trajectory segments that define the problem being simulated from beginning to end. These trajectory segments, or phases, are defined by two events--a beginning event and an ending event. An event is an interruption of the trajectory simulation that occurs when a user-specified variable reaches a user-specified value. An event must be created whenever the user wishes to change any input data for the problem or to cause any change in the method of simulating the problem. For example, the sequence of events for a typical ascent problem could result in a simulation setup similar to that shown in figure VI-1.



Event	Description
1	Liftoff
2	Initiate pitch rate 1 at 20 sec
3	Initiate pitch rate 2 at 30 sec
4	Initiate pitch rate 3 at 60 sec
5	Initiate angle of attack control at 75 sec
6	Jettison stage 1 when propellant consumed
7	Initiate pitch rate 4 20 sec after staging
8	Initiate yaw rate 1 100 sec after event 7
9	Orbit injection at inertial velocity of 25 568.0 fps

Figure VI-1.- Event Sequence Setup

The event numbers for a given problem must be specified as real numbers by the user in monotonic increasing order. These event numbers are then used by the program to determine the order in which the events are to occur. The program requires that each problem have a minimum of two events--an initial event and a final event. Since a phase is initiated by the corresponding event, the event criterion for a given event specifies the conditions at the beginning of the corresponding phase. A problem is terminated by specifying the last event that is to occur. The problem can also be terminated in a psuedo-abort mode by specifying the maximum trajectory time, maximum altitude, or minimum altitude.

Although event numbers must be monotonic increasing, they need not be consecutive. This allows the user to easily add or delete events from an input deck.

Four types of events have been defined to provide flexibility in setting up a given problem:

1) Primary events - These describe the main sequential events of the trajectory being simulated. These events must occur, and must occur in ascending order according to the event number. Most problems will usually be simulated by a series of primary events;

2) Secondary events - These are events that may or may not occur during the specified trajectory segment. Secondary events must occur in ascending order during the interval bounded by the primary events. The occurrence of a primary event will nullify the secondary events associated with the previous primary event if they have not already occurred;

3) Roving primary events - These events can occur any time after the occurrence of all primary events with smaller event numbers. They can be used to interrupt the trajectory on the specified criterion regardless of the state of the trajectory or vehicle.

4) Repeating roving events - These events are the same as primary roving events except the interrupt values are input differently. There are two options for criteria value specifications. Option 1: Input the initial value, the increment, and the number of times the event is to be repeated. Option 2: Input an array of event criteria values.

The program monitors as many as ten events at a time, depending on the types of events to determine which event is to occur next. This gives the user a powerful tool for simulating complex problems.

Multiple events are monitored in the following sequence:

1) The next primary event is monitored;

2) As many as nine primary roving events are then monitored, provided there are no secondary events. A roving primary event is added to the list of those being monitored as soon as the primary event immediately preceding that roving event has occurred;

3) Next, as many as nine secondary events are monitored, provided there are no primary roving events. (Note that caution must be exercised when using secondary events because of their nature. Since as many as nine secondary events are monitored at a time, any one of those nine will occur as soon as its criterion has been met. Because they are secondary events, the event that occurs will cancel all secondary events with smaller event numbers.);

4) Finally, a total of nine primary roving and secondary events are monitored.

Since the program can only monitor nine events (in addition to the next primary event), the sum of the primary roving events and the secondary events must be less than or equal to nine or a fatal error will result.

The time-to-go model (TGØM) determines when the events occur during the trajectory simulation. Basically, TGØM checks the values of the criteria being monitored at each integration step. If none of the criteria values has bracketed the desired cutoff value, then another integration step is taken. If a criterion variable is bracketed with the input step size, then TGØM computes a new stepsize equal to the predicted time-to-go.

The predicted time-to-go for each event is computed from the equation

$$\Delta t^* = - y(t) \Delta t / (y(t + \Delta t) - y(t)) \qquad \text{(VI-1)}$$

where $y(t)$ is the difference between the actual and the desired value of the event criterion. If more than one event is bracketed, then the minimum predicted time-to-go is used as the integration stepsize. This process is repeated until the criterion value is within the specified tolerance of the desired value. If the desired condition cannot be achieved in 20 iterations, an error message is printed and the program stops. Generally this situation is caused by an input error. The fundamental features of the time-to-go logic are shown in figure VI-2.

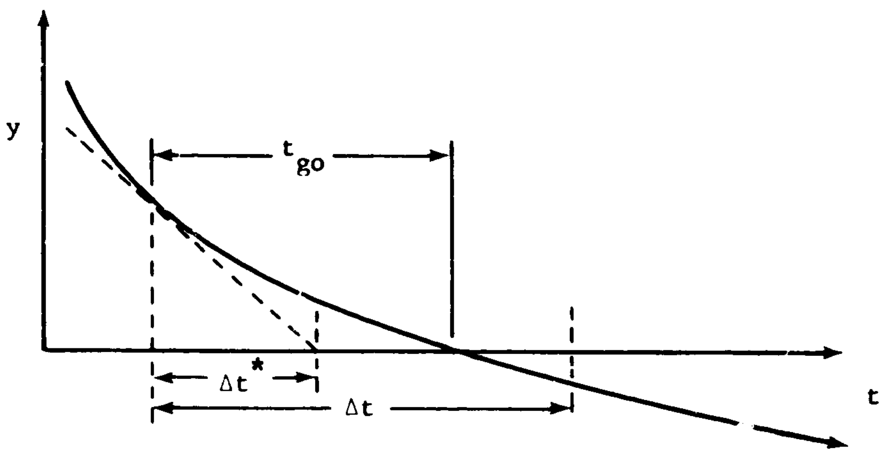


Figure VI-2.- Illustration of Time-to-Go Logic

Translational Equations

The translational equations of motion are solved in the planet-centered inertial coordinate system. These equations are

$$\dot{\underline{r}}_I = \underline{V}_I \tag{VI-2}$$

$$\dot{\underline{V}}_I = [IB]^{-1} \left[\underline{A}_{TB} + \underline{A}_{AB}\right] + \underline{G}_I , \tag{VI-3}$$

where $\underline{A}_{TB}$ is the thrust acceleration in the body frame, $\underline{A}_{AB}$ is the aerodynamic acceleration in the body frame, and $\underline{G}_I$ is the gravitational acceleration in the ECI frame.

Initialization.- There are five options for initializing the velocity vector and two options for initializing the position vector. These options are described below.

Inertial position components (x_I, y_I, z_I).- The inertial position components can be input directly since no transformation is required.

Earth-relative position $\left(\theta_I \text{ or } \theta, \phi_c \text{ or } \phi_g, h \text{ or } r\right)$.- In this option the equations vary and the sequence of calculation varies according to the choice of input. However, the basic equations used are:

$$\left.\begin{aligned} \theta_I &= \theta + \Omega_p\left(t - t_0\right) && \text{if } \theta \text{ is input,} \\ \phi_c &= \tan^{-1}\left(k^2 \tan \phi_g\right) && \text{if } \phi_g \text{ is input,} \\ r_I &= h + R_s\left(\phi_c\right) && \text{if } h \text{ is input,} \end{aligned}\right\} \qquad \text{(VI-4)}$$

and

$$\underline{r}_I = r_I \begin{bmatrix} \cos\phi_c \cos\theta_I \\ \cos\phi_c \sin\theta_I \\ \sin\phi_c \end{bmatrix} \qquad \text{(VI-5)}$$

Inertial velocity components (V_{XI}, V_{YI}, V_{ZI}).- These variables can be input directly.

Inertial local horizontal (V_I, γ_I, A_{ZI}).- The inertial components in the horizontal frame are first transformed to the geographic frame as

$$\underline{V}_{IG} = V_I \begin{bmatrix} \cos\gamma_I \cos A_{ZI} \\ \cos\gamma_I \sin A_{ZI} \\ -\sin\gamma_I \end{bmatrix} \qquad \text{(VI-6)}$$

and then transformed to the ECI frame by

$$\underline{V}_I = [IG]^{-1} \underline{V}_{IG}. \qquad \text{(VI-7)}$$

Earth-relative local horizontal (V_R, γ_R, A_{ZR}).- The Earth-relative velocity components are first transformed to the geographic frame as

$$\underline{V}_{RG} = V_R \begin{bmatrix} \cos \gamma_R \cos A_{ZR} \\ \cos \gamma_R \sin A_{ZR} \\ -\sin \gamma_R \end{bmatrix} \qquad \text{(VI-8)}$$

and then transformed to the ECI frame by

$$\underline{V}_I = [IG]^{-1} \underline{V}_{RG} + \underline{\Omega}_p \times \underline{r}_I \qquad \text{(VI-9)}$$

Atmospheric relative local horizontal (V_A, γ_A, A_{ZA}).- The atmospheric relative velocity components are first transformed to the geographic frame as

$$\underline{V}_{AG} = V_A \begin{bmatrix} \cos \gamma_A \cos A_{ZA} \\ \cos \gamma_A \sin A_{ZA} \\ -\sin \gamma_A \end{bmatrix} + \begin{bmatrix} V_{WXG} \\ V_{WYG} \\ V_{WZG} \end{bmatrix}, \qquad \text{(VI-10)}$$

and then transformed to the ECI frame by

$$\underline{V}_I = [IG]^{-1} \underline{V}_{AG} + \underline{\Omega}_p \times \underline{r}_I \qquad \text{(VI-11)}$$

Orbital parameters $(h_p, h_a, i, \Omega, \omega_p, \theta)$.- This option initializes both position and velocity. The equations used to transform the orbital parameter to the ECI position and velocity are:

$$\left.\begin{aligned}
r_p &= h_p + R_E \\
r_a &= h_a + R_E \\
&= \left(r_a + r_p\right)/2 \\
e &= \left(r_a - r_p\right)/\left(r_a + r_p\right) \\
\rho &= \theta + \omega \\
p &= a\ (1 - e^2) \\
H &= \mu p \\
r &= p/(1 + e \cos \theta) \\
\underline{u}_r &= \begin{bmatrix} \cos \Omega & -\sin \Omega & 0 \\ \sin \Omega & \cos \Omega & 0 \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & -\sin i \\ 1 & \sin i & \cos i \end{bmatrix} \begin{bmatrix} \cos \rho \\ \sin \rho \\ 0 \end{bmatrix}
\end{aligned}\right\} \quad \text{(VI-12)}$$

and

$$\left.\begin{aligned}
\underline{r} &= r\ \underline{u}_r \\
V &= \mu \left[\frac{2}{r} - \frac{1}{a}\right] \\
\gamma &= \sin^{-1}\ (H/rV) \\
\underline{u}_V &= \begin{bmatrix} \cos \rho & -\sin \rho & 0 \\ \sin \rho & \cos \rho & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & -\sin i \\ 0 & \sin i & \cos i \end{bmatrix} \\
&\qquad \begin{bmatrix} \cos \Omega & -\sin \Omega & 0 \\ \sin \Omega & \cos \Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \gamma \\ \sin \gamma \\ 0 \end{bmatrix}
\end{aligned}\right\} \quad \text{(VI-13)}$$

(VI-14)

<u>Instantaneous velocity additions</u>.- At the beginning of each event an instantaneous velocity can be added in the direction of the thrust vector. The magnitude of the velocity addition can be input directly, or can be calculated from the rocket equation from the amount of propellant consumed

$$\Delta V = g_0 \; I_{sp} \; \ell n \left[\frac{W_G}{W_G - W_{PC}} \right] \tag{VI-15}$$

If ΔV is input, then the amount of propellant required to achieve this change in velocity is given by

$$W_{PC} = W_G \left[1 - \exp\left(- \Delta V / g_0 \; I_{sp}\right)\right]. \tag{VI-16}$$

The inertial velocity after the impulse is then given by

$$\underline{V}_I^+ = \underline{V}_I^- + \Delta V \; [IB]^{-1} \begin{bmatrix} \cos i_y \cos i_p \\ \sin i_y \\ \cos i_y \sin i_p \end{bmatrix} \tag{VI-17}$$

This option is generally used to simulate short burns for orbital maneuvers. The direction of the impulse is controlled via the attitude of the vehicle and the engine gimbal angles.

Static Trim

The static trim option is used to calculate the engine gimbal angles or the flap deflection angles required to balance the pitch and yaw moments cuased by the thrust and aerodynamic forces.

The static moment equation,

$$\underline{M} = \underline{0}, \tag{VI-18}$$

is generally nonlinear, and thus an iterative algorithm is used to determine the required solutions. This algorithm is a successive approximation technique based on the analytical solutions that result from small-angle approximations. The computation of the pitch and yaw moments is presented below.

Moment equation in the pitch plane.- The aerodynamic moment and thrust moments for the ith engine are shown in figure 16. The locations of the center of gravity, the aerodynamic reference, and the engine gimbal points are specified in the body references system.

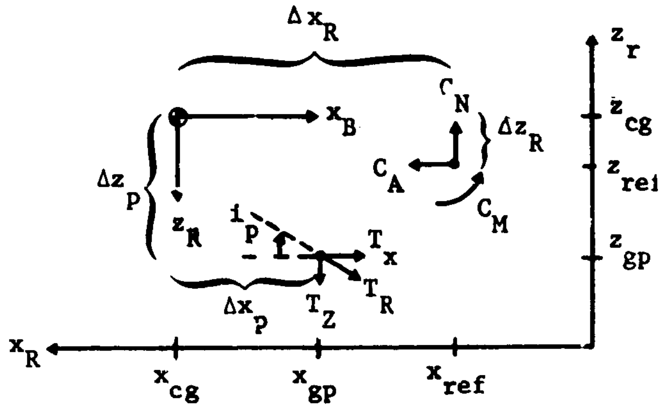


Figure VI-3. - Moments in Pitch

The total aerodynamic moment in pitch is

$$A_{MP} = qS\left(C_M \ell_{RP} + C_N \Delta x_R - C_A \Delta z_R\right). \quad \text{(VI-19)}$$

The total thrust moment in pitch for the nontrimming engines is

$$T_{M1P} = \sum_{i \notin N_T} T_{R_i} \cos i_{y_i} \left(\cos i_{p_i} \Delta z_{p_i} - \sin i_{p_i} \Delta x_{p_i}\right), \quad \text{(VI-20)}$$

and the moment for the trimming engines is

$$T_{M2P} = \cos i_p \sum_{i\epsilon N_T} T_{R_i} \cos i_{y_i} \Delta z_{p_i}$$

$$- \sin i_p \sum_{i\epsilon N_T} T_{R_i} \cos i_{y_i} \Delta x_{p_i} . \qquad \text{(VI-21)}$$

The static moment equation in the pitch plane is then given by

$$M_P = A_{MP} + T_{M1P} + T_{M2P}, \qquad \text{(VI-22)}$$

where A_{MP} depends on the flap deflection angles and T_{M2P} depends on the engine gimbal angles.

Moment equation in the yaw plane.- The aerodynamic moment and the thrust moment for the ith engine are shown in figure 17.

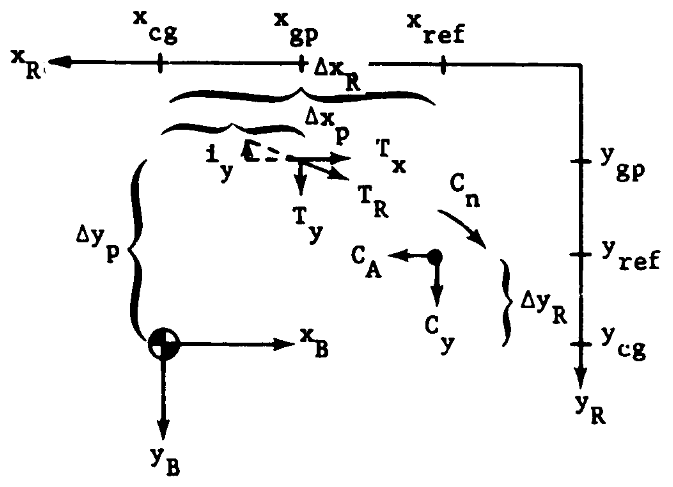


Figure VI-4. - Moments in Yaw

The total aerodynamic moment in yaw is

$$A_{MY} = qS \left(C_n \ell_{RY} + C_Y \Delta x_R - C_A \Delta y_R\right). \tag{VI-23}$$

The total thrust moment in yaw for the nontrimming engines is

$$T_{M1Y} = \sum_{i \notin N_T} T_{R_i} \left(\cos i_{p_i} \cos i_{y_i} \Delta y_{p_i} + \sin i_{y_i} \Delta x_{p_i}\right), \tag{VI-24}$$

and the moment for the trimming engines is

$$T_{M2Y} = \cos i_p \sum_{i \varepsilon N_T} T_{R_i} \cos i_{p_i} \Delta y_{p_i}$$

$$+ \sin i_y \sum_{i \varepsilon N_T} T_{R_i} \Delta x_{p_i} . \tag{VI-25}$$

The static moment equation in the yaw plane is then given by

$$M_Y = A_{MY} + T_{M1Y} + T_{M2Y}, \tag{VI-26}$$

where A_{MY} depends on the flap deflections and T_{M2Y} depends on the engine gimbal angles.

Integration Methods

The number of integrals computed during any particular phase is determined from the options requested by the user. As a minimum, the translational equations of motion are integrated to give the position and velocity of the center of mass of the vehicle. The user may also select additional variables to be integrated. The only restriction is that no more than 30 integrals can be computed per phase.

POST contains 3 general purpose integration methods and several special purpose orbital propagation methods. These methods are summarized below.

Runge-Kutta Methods.- POST contains two Runge-Kutta integration methos: (1) the standard 4th order method, and (2) a 8th order method. The calculations for these integration methods are based upon the formula

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i,$$

where

$$k_i = hf\left(x_n + c_i h,\ y_n + \sum_{i=1}^{s} a_{ij} k_j\right),\ i=1,2,\cdots,s.$$

These formula are represented by the array

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

The coefficients for the two methods are given in Tables VI-1 and VI-2, respectively

Table IV-1

Runge-Kutta 4th order (Kutta)

0	--			
1/2	1/2	--		
1/2	0	1/2	---	
1	0	0	1	--
	1/6	1/3	1/3	1/6

Runge-Kutta 8th order (Shanks (1966))

0.000	----											
0.111	0.1111	----										
0.167	0.0042	0.1250	----									
0.250	0.0069	0.0000	0.1875	----								
0.100	0.0580	0.0000	0.0780	-0.0360	----							
0.167	0.0340	0.0000	0.0000	0.0041	0.1286	----						
0.500	-0.5838	0.0000	0.0000	2.1111	3.4722	-4.5000	----					
0.666	-0.1235	0.0000	0.0000	-0.1317	0.5144	0.0000	0.4074	----				
0.333	3.6265	0.0000	0.0000	-10.6667	19.2901	26.0000	0.7469	-0.0833	----			
0.833	0.9043	0.0000	0.0000	-2.6296	4.2438	5.6667	-0.3642	0.5000	1.000	----		
0.833	0.8043	0.0000	0.0000	2.6296	-4.2438	6.1667	0.6358	0.0000	0.0000	0.1000	----	
1.000	-1.9411	0.0000	0.0000	6.9377	11.0095	-14.9268	0.0854	-0.1646	-0.4390	-0.2927	0.7317	----
	0.0488	0.0000	0.0000	0 0.0000	0.0000	0.2571	0.3238	0.0321	0.0321	0.0429	0.2143	0.0488

Krogh Variable-Step Variable-Order Integrator

The variable-step length variable-order predictor-corrector developed by F. T. Krogh of the Jet Propulsion Laboratories procedure represents the state-of-the-art in numerical solution of systems of ordinary differential equations. It includes all of the following facilities:

1) A core integrator for advancing the solution from one uniform step to the next consisting of variable order Adams predictor-corrector equations requiring the storage of a difference table for only the highest ordered derivatives;

2) A method to start integration with first-order equations and increase the order to as high a level as numerical stability permits;

3) Algorithms for changing the step size and updating accordingly the difference tables of the highest-order derivatives including appropriate smoothing to prevent numerical instability;

4) Algorithms for deciding when and by how much to change the step size based upon the accuracy requested by the user;

5) Tests for numerical stability of the predictor-corrector order and step size tentatively chosen in the context of the current differential equation set;

6) Test for accuracy requests that are so stringent that round-off error prevents their satisfaction;

7) Algorithms for the automatic selection of the core integrator to fit the characteristics of the set of differential equations at hand.

8) An interpolation algorithm for obtaining dependent variable values to the user-specified accuracy at values of the independent variable different from normal integration steps.

The Krogh integrator was developed to meet the conflicting goals of (1) reliability, (2) efficiency (in the sense of minimizing the number of derivative evaluations to obtain a given accuracy), (3) flexibility, (4) low integration overhead, and (5) small storage requirements. The goals are listed in the order in which they are emphasized in the procedures. The package has no equal in reliability. All the user need provide the integrator is his accuracy requirement and a tentative step size that is less than the integration interval. The integrator then uses all eight of its algorithms to provide a solution requiring as few derivative evaluations as possible while remaining within accuracy tolerances.

This documentation restricts itself to algorithms (1), (2), and (3). Algorithm (3) is thoroughly described in Reference 9. Algorithms (4) through (7) are not amenable to rigorous mathematical analysis, but rather have evolved from extensive numerical experimentation by Krogh and his associates at JPL. They are adequately described in the source code and references. Further, to simplify exposition, equations will be given for the single d^{th} order differential equation

$$Y^{(d)} = f\left(t, Y, Y^1, \cdots, Y^{d-1}\right) \qquad [1]$$

The following notation is used:

t = independent variable [2]
h = integration step size [3]
$y(t)$ = corrected (final estimated) value of $Y(t)$ [4]
t_n = value of t at current integration step [5]

$$t_{n+k} = t_n + kh \qquad [6]$$

$$y_n, Y_n, y_n^1, \cdots = y(t_n), Y(t_n), y^1(t_n), \cdots \qquad [7]$$

p_n = predicted (initial estimated) value of $Y(t_n)$ [3]

$$f_n = f\left(t_n, y_n, y_n^1, \cdots, y_n^{(d-1)}\right) \qquad [9]$$

$$\nabla^i f_n = \begin{cases} f_n & \text{for } i=0 \\ \nabla^{i-1} f_n - \nabla^{i-1} f_{n-1} & \text{for } i=1,2,\cdots \end{cases} \qquad [10]$$

$$\nabla_p^i f_n = \begin{cases} f\left(t_n, p_n, p_n^1, \cdots, p_n^{(d-1)}\right) & \text{for } i=0 \\ \nabla_p^{i-1} f_n - \nabla^{i-1} f_{n-1} & \text{for } i=1,2,\cdots \end{cases} \qquad [11]$$

The predictor and corrector equations of the core integrator are obtained by integrating Newton's interpolating polynomials for the most recent q and q+1 values, respectively, of the highest order derivative, f. For the predictor, f is approximated as the (q-1)th degree polynomial

$$f_{n+s}^p = \sum_{i=0}^{q-1} \binom{s+i-1}{i} \nabla^i f_n \approx f(x_n + sh). \qquad [12]$$

Successively integrating approximation [12] from s=o to s=1 yields the predictor equations.

$$p_{n+1}^{(d-j)} = \sum_{k=0}^{j-1} \frac{h^k}{k!} y_n^{(d-j+k)} + h^j \sum_{i=0}^{q-1} \gamma_{ij} \nabla^i f_n \text{ for } j=1,2,\cdots,d \qquad [13]$$

where $\sigma_{i0}(s) = \binom{s+i-1}{i}$ for $i=1,\cdots,q,$ [14]

$$\sigma_{ij}(s) = \int_0^s \sigma_{i(j-1)}(r)\, dr \qquad \text{for } i=1,\cdots,q, \; j=1,\cdots,d \qquad [15]$$

and $\gamma_{ij} = \sigma_{ij}(1)$ for $i=1,\cdots,q$, $j=1,\cdots,d$ [16]

For the corrector, f is approximated as the q^{th} degree polynomial

$$f_{n+s}^c = f_{n+s}^p + \sigma_{q0}(s)\, \nabla_p^i f_n \approx f(x_n + sh). \qquad [17]$$

Successively integrating approximation [17] from s=o to s=1 generates the corrector equations

$$y_{n+1}^{(d-j)} = p_{n+1}^{(d-j)} + h^j \gamma_{qj} \nabla_p{}^q f_{n+1} \quad \text{for } j=1, \cdots, d. \qquad [18]$$

Notice that in the equations [13] and [18] the coefficients γ_{ij} are independent of the order, q. Hence, Adam's method has the practical advantage that its order can be increased simply by adding more terms to the predictor and corrector equations without revising their coefficients. Systems of differential equations are handled by applying equations [13] and [18] to each component of the system. Different components may have different values of d and q.

The starting algorithm requires no special techniques such as Runge-Kutta equations or Taylor-series expansions. Instead, q is simply taken as 1 in the normal predictor and corrector equations. There is, nonetheless, a starting phase during which the highest-order derivative evaluation following the calculation of the corrected values of the integrals is skipped. During this starting phase, the values of q and h increase quite rapidly. When numerical stability problems appear during this initial phase, the ordinary corrector equation, [18], is replaced by

$$y_{n+1}^{(d-j)} = p_{n+1}^{(d-j)} + h^j \alpha \gamma_{ij} \nabla_p^1 f_{n+1} \quad \text{for } j=1,2,\cdots,d. \qquad [19]$$

Here α is chosen on the closed interval from ¼ to 1 to give optimal stability characteristics.

Algorithms [3] for changing the step size are only equipped to double or halve h. Hence, algorithm [4] for selecting the step size is restricted in the same way. Both procedures are, however, very efficient so that adjustment to the optimal step length occurs as quickly as reliability permits. The two algorithms are described in detail in Reference 11.

Like the predictor and corrector equations, the interpolation equations of algorithm [8] are also generated by successively integrating a qth degree interpolating polynomial for the highest order derivative. Indeed

$$f_{n+s}^{o} = \sum_{i=0}^{q} \binom{s+i-2}{i} \nabla^i f_{n+1} \approx f(x_n + sh) \qquad [20]$$

$$y_{n+s}^{(d-1)} = y_n^{(d-1)} + \sum_{\ell=0}^{q} \rho_{i1} \nabla^{\ell} f_{n+s}^{o} \approx Y^{(d-1)} (x_n + sh) \qquad [21]$$

where

$$\rho_{i1} = \int_0^1 \binom{s + i-2}{i} ds \qquad [22]$$

Evaluating the differences, $\nabla^{\ell} f_{n+s}^{o}$, using definition [20] and substituting the results into definition [21] yields

$$y_{n+s}^{(d-j)} = y_n^{(d-1)} + \sum_{i=0}^{q} \left[\sum_{\ell=0}^{i} \rho_{\ell 1} \binom{s + i-\ell-2}{i-\ell} \right] \nabla^{i} f_{n+1} \qquad [23]$$

Generalizing this procedure to derivatives of lower order produces

$$y_{n+s}^{(d-j)} = \sum_{k=0}^{j-1} \frac{s^k}{k!} y_n^{(d-j+k)} + h^j \sum_{i=0}^{q-j} \tau_i^{j} \nabla^{i} f_{n+1} \qquad [24]$$

for $j=1, \cdots, d$

where

$$\tau_i^{o} (s) = \binom{s \quad i-\ell-2}{i} \qquad [25]$$

and

$$\tau_i^{j}(s) = \sum_{\ell=0}^{q-j} \rho_{\ell 1} \tau_{(i-\ell+1)\ (j-1)}(s) \ . \qquad [26]$$

Laplace's method.- Laplace's method is an iterative technique for propagating the position and velocity (in planet-centered coordinates) of a nonthrusting vehicle in vacuum during flight over a spherical planet. The technique is based on the analytical solution of the two-body equations, and yields the inertial state at time $t + \Delta t$ as a function of the state at time t.

The equations used in Laplace's method are:

$$\left.\begin{aligned} \underline{r}_I\,(t+\Delta t) &= f\,\underline{r}_I\,(t) + g\,\underline{V}_I\,(t) \\ \underline{V}_I\,(t+\Delta t) &= \dot{f}\,\underline{r}_I\,(t) + \dot{g}\,\underline{V}_I\,(t), \end{aligned}\right\} \qquad \text{(VI-27)}$$

where the scalar coefficients f, g, $\dot{f}$, and $\dot{g}$ are given in Table VI-3.

Table VI-3 - f and g Series

Coefficient	Elliptical*	Hyperbolic*
f	$1 - 2\frac{a}{r_n}\sin^2\frac{\Delta E}{2}$	$1 + 2\frac{a}{r}\sinh^2\frac{\Delta E}{2}$
g	$\Delta t - \frac{1}{n}(\Delta E - \sin\Delta E)$	$\Delta t - \frac{1}{n}(\Delta E - \sinh\Delta E)$
$\dot{f}$	$-\frac{\sqrt{\mu a}\,\sin\Delta E}{r_n\,r_{n+1}}$	$\frac{-\sqrt{\mu a}\,\sinh\Delta E}{r_n\,r_{n+1}}$
$\dot{g}$	$1 - 2\frac{a}{r_{n+1}}\sin^2\frac{\Delta E}{2}$	$1 + 2\frac{a}{r_{n+1}}\sinh^2\frac{\Delta E}{2}$
*Note that $r_j = \lvert\underline{r}(t + j\,\Delta t)\rvert$, and $n = \frac{2\pi}{P}$.		

As indicated in the above table, the f and g coefficients are computed analytically from the change in eccentric anomaly during the time period Δt. This is in contrast to the standard Laplacian method, where these coefficients are infinite series in Δt. The change in eccentric anomaly is calculated by solving a special version of Kepler's equation via the Newton-Raphson algorithm

$$\Delta E_{n+1} = \Delta E_n - \frac{F(\Delta E)}{F^1(\Delta E)}\ , \ n = 1, 2, \ldots, 10. \qquad \text{(VI-28)}$$

This algorithm converges rapidly from the initial estimate

$$\Delta E_0 = n\ \Delta t. \tag{VI-29}$$

The form of Kepler's equation that is computationally efficient for this application is

$$F(\Delta E) = \begin{cases} \left[\dfrac{r_n}{a} - 1\right] \sin \Delta E + 2\, \dfrac{\underline{r}_n \cdot \underline{v}_n}{\sqrt{\mu}\ a} \sin^2 \dfrac{\Delta E}{2} + \Delta E - n\ \Delta t, & \text{for } e < 1 \\ \left[\dfrac{r_n}{a} - 1\right] \sinh \Delta E - 2\, \dfrac{\underline{r}_n \cdot \underline{v}_n}{\sqrt{\mu}\ a} \sinh^2 \Delta E + \Delta E - n\ \Delta t, & \text{for } e > 1. \end{cases} \tag{VI-30}$$

The derivative dF/dE used in the Newton-Raphson algorithm is given by

$$F'(\Delta E) = \begin{cases} \left[\dfrac{r_n}{a} - 1\right] \cos \Delta E + \dfrac{\underline{r}_n \cdot \underline{v}_n}{\sqrt{\mu}\ a} \sin \Delta E + 1, & \text{for } e < 1 \\ \left[\dfrac{r_n}{a} - 1\right] \cosh \Delta E - \dfrac{\underline{r}_n \cdot \underline{v}_n}{\sqrt{\mu}\ a} \sinh \Delta E - 1, & \text{for } e > 1. \end{cases} \tag{VI-31}$$

Encke's method.- The Encke method used in this program is modified from the usual Encke technique in that it rectifies the reference conic at every integration step and does not use the standard Q-series expansion in calculating the gravitational increment.

The Encke method should be used for orbital problems where small perturbing accelerations, such as the oblateness of the planet, atmospheric drag, or solar electric propulsion, must be included in the simulation. Numerical results indicate that, for problems involving small perturbations from Keplerian motion, Encke's method is approximately four times faster than Cowell methods, which integrate the total acceleration.

The Encke method determines the total motion by summing the motion due to the two-body equations and the motion due to the perturbations to the two-body equations. The position and velocity in the inertial planet-centered system at time $t + \Delta t$ is given by

$$\left.\begin{aligned} \underline{r}_I\ (t + \Delta t) &= \underline{r}_2\ (t + \Delta t) + \Delta\underline{r}\ (t + \Delta t) \\ \underline{V}_I\ (t + \Delta t) &= \underline{V}_2\ (t + \Delta t) + \Delta\underline{V}\ (t + \Delta t), \end{aligned}\right\} \quad \text{(VI-32)}$$

where $(\underline{r}_2, \underline{V}_2)$ denotes the Keplerian motion computed by Laplace's equations; that is,

$$\left.\begin{aligned} \underline{r}_2\ (t + \Delta t) &= f\ \underline{r}_2\ (t) + g\ \underline{V}_2\ (t) \\ \underline{V}_2\ (t + \Delta t) &= \dot{f}\ \underline{r}_2\ (t) + \dot{g}\ \underline{V}_2(t), \end{aligned}\right\} \quad \text{(VI-33)}$$

and $(\Delta\underline{r}, \Delta\underline{V})$ denotes the numerical solution of the differential equations

$$\left.\begin{aligned} \Delta\dot{\underline{r}} &= \Delta\underline{V} \\ \Delta\dot{\underline{V}} &= \left[[IB]^{-1}\left[\underline{A}_{TB} + \underline{A}_{AB}\right] + \underline{G}_I\right] - \underline{g}_2\ (\underline{r}_2 + \Delta\underline{r}) \\ \Delta\underline{r}(t) &= \Delta\underline{V}\ (t) = \underline{0}, \end{aligned}\right\} \quad \text{(VI-34)}$$

where $\underline{g}_2\ (\underline{r}_2 + \Delta\underline{r})$ is the two-body acceleration at $\underline{r}_2 + \Delta\underline{r}$. The Runge-Kutta or Adams-Bashforth method can then be used to solve the above equations.

Integration step models.- The integration step, Δt, is generally specified in terms of an increment in time. However, this option enables the user to specify the integration step in terms of an input increment in true anomaly. This option is useful in orbital problems where the geometry is easily expressed as a function of true anomaly.

The following equations are then used to calculate Δt as a function of $\Delta\theta$.

$$\left.\begin{aligned} \theta_2 &= \theta_1 + \Delta\theta \\ E_2 &= 2\ \tan^{-1}\left\{\left(\frac{1-e}{1+e}\right)^{1/2} \tan\frac{\theta_2}{2}\right\} \\ r_2 &= a(1 - e\cos E_2) \\ \Delta E &= E_2 - E_1 \\ \Delta t &= \sqrt{\frac{a^3}{\mu}}\ (\Delta E - \sin\Delta E) + \frac{r_2 r_1}{H}\ \sin\Delta\theta. \end{aligned}\right\} \quad \text{(VI-35)}$$

In these equations, subscripts 1 and 2 denote current and future values, respectively.

Launch Options

There are two specific options for simulating particular launch conditions: (1) hold-down for vertical takeoff, and (2) hold-down for horizontal takeoff. These specialized options are required to simulate certain physical constraints that are not modeled in the equations of motion.

Holddown for vertical takeoff.- This option is used to simulate vertical (rocket type) takeoff. When using this option, the relative position and velocity remains constant while the inertial position changes by the Earth rotation. The inertial velocity magnitude remains constant while its direction changes. This model simulates physical constraints that hold the rocket on the launch pad until the rocket is released. The equations used to calculate the accelerations that produce this motion are

$$\underline{A}_I = \underline{\Omega} \times \underline{V}_I \tag{VI-36}$$

Holddown for horizontal takeoff.- This option is used to simulate horizontal (aircraft type) takeoff. When using this option the vehicle accelerates in the local horizontal plane according to the forces described by the user input. The vertical component of acceleration is internally computed to produce the proper horizontal motion. The equations used are:

$$\underline{A}_{XG} = \left[IG\right] \underline{A}_{SI}, \tag{VI-37}$$

$$\begin{bmatrix} A_V \\ A_H \\ A_C \end{bmatrix} = \begin{bmatrix} -A_{ZG} \\ \sqrt{A_{XG}^2 + A_{YG}^2} \\ \left(V_{XG}^2 + V_{YG}^2\right) / r_I \end{bmatrix},$$

and

$$\underline{A}_I = \left[IG\right]^{-1} \underline{A}_{XG} \tag{VI-38}$$

VII. AUXILIARY CALCULATIONS

In addition to computing the basic variables, POST also computes numerous auxiliary variables that are related to: (1) conic parameters, (2) range calculations, (3) tracking data, (4) analytic impact calculations, (5) velocity losses, and (6) velocity margins. The equations used to calculate these variables are presented below.

Conic Calculations

The following Keplerian conic variables are computed.

$\mathcal{E}$	energy, $\frac{V_I^2}{2} - \frac{\mu}{r_I}$
a	semimajor axis, $-\mu/2\mathcal{E}$
h	angular momentum, $\lvert \underline{r}_I \times \underline{V}_I \rvert$
p	semilatus rectum, h^2/μ
e	eccentricity, $\sqrt{\lvert 1 - p/a \rvert}$
ΔV	velocity required to circularize orbit, $\sqrt{\Delta\underline{V} \cdot \Delta\underline{V}}$, where

$$\underline{u}_h = \underline{h}/h$$

$$\underline{u}_{RI} = \underline{r}_I/r_I$$

$$\underline{u}_v = \underline{u}_h \times \underline{u}_{rI}/\lvert \underline{u}_h \times \underline{u}_{rI} \rvert$$

$$\underline{V}_c = (\mu/r_I)^{\frac{1}{2}}\, \underline{u}_v$$

$$\Delta\underline{V} = \underline{V}_c - \underline{V}_I$$

i	inclination, $\cos^{-1}\left(h_z/h\right)$
Ω	longitude of ascending node, $\cos^{-1}\left(\hat{\underline{x}}_I \cdot \underline{u}_\Omega\right)$, where

	$\underline{u}_\Omega = \hat{\underline{z}}_I \times \underline{h} / \lvert \hat{\underline{z}}_I \times \underline{h} \rvert$
ρ	argument of vehicle, $\rho = \cos^{-1}\left(\underline{u}_r \cdot \underline{u}_\Omega\right)$
T_{SP}	time since perigee, $\frac{P}{2\pi} M$
T_{TP}	time to perigee, $P - T_{SP}$
ϕ_p	latitude of perigee, $\tan^{-1}\left(u_3/\sqrt{u_1^2 + u_2^2}\right)$, where $\underline{u} = \cos(\omega)\underline{u}_\Omega + \sin(\omega)\left(\underline{u}_h \times \underline{u}_\Omega\right)$
θ_p	longitude of perigee, $\tan^{-1}(u_2/u_1)$
h_p	altitude of perigee, $r_p - R_s(\phi_p)$
h_a	altitude of apogee, $r_a - R_s(\phi_p)$
V_p	velocity at perigee, $\sqrt{\frac{\mu}{a}\left(\frac{1+e}{1-e}\right)}$
V_a	velocity at apogee, $\sqrt{\frac{\mu}{a}\left(\frac{1-e}{1+e}\right)}$
V_∞	hyperbolic excess velocity, $\sqrt{2\mathcal{E}}$
θ_{max}	maximum true anomaly for hyperbolic orbit, $\cos^{-1}(-1/e)$
δ_{RA}	declination of outgoing asymptote, $\sin^{-1}[u_{r_\infty}(3)]$, where $\underline{u}_T = \underline{u}_h \times \underline{u}_{RI}$; $u_{r_\infty} = \cos(\theta_{max} - \theta)\,\underline{u}_{rI} + \sin(\theta_{max} - \theta)\,\underline{u}_T$
R_A	right ascension of outgoing asymptote, $\tan^{-1}\left(\frac{u_{r_\infty}(2)}{u_{r_\infty}(3)}\right)$

C-2

θ	true anomaly, $\cos^{-1}\left(\frac{1}{e}\left(\frac{p}{r} - 1\right)\right)$
E	eccentric anomaly, $2\tan^{-1}\left(\sqrt{\frac{1-e}{1+e}}\tan\frac{\theta}{2}\right)$
M	mean anomaly, $E - e\sin E$
ω	argument of perigee, $\rho - \theta$
r_p	perigee radius, $a(1 - e)$
r_a	apogee radius, $a(1 + e)$
P	period, $2\pi\sqrt{\frac{a^3}{\mu}}$

Range Calculations

The progam provides for various types of range calculations. The equations for these calculations are given below.

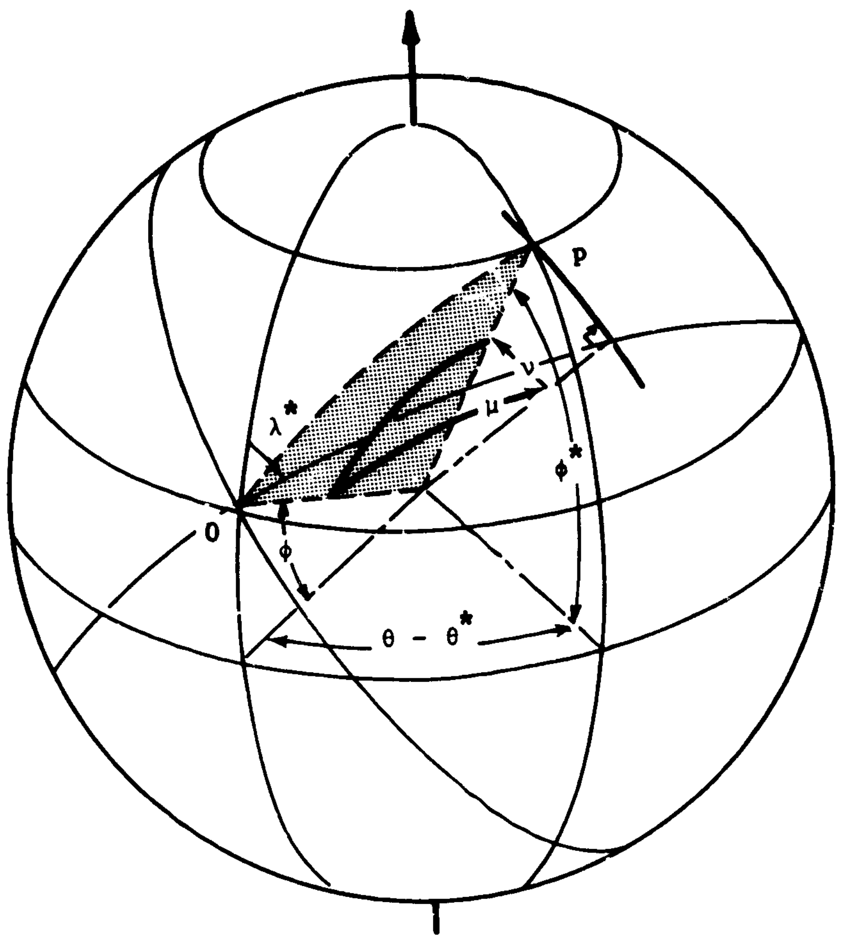
Dot product downrange.- The relative range angle, measured from the vehicle's initial position to its current position, is given by

$$\phi_R = \cos^{-1}\left(\underline{u}_{r_{so}} \cdot \underline{u}_{r_s}\right), \tag{VII-1}$$

where $\underline{u}_{r_{so}}$ is a unit vector along the initial position vector in Earth-centered rotating coordinates and $\underline{u}_{r_s}$ is a unit vector along the current position vector in Earth-centered rotating coordinates. The range over an oblate spheroid is calculated from the average radius to the surface, and is given by

$$R_D = \left[\frac{r_{so} + r_s}{2}\right]\phi_R \tag{VII-2}$$

Crossrange and downrange via orbital plane reference.- Referring to figure VII-1, identify the vehicle's position at time t^* by O, and at a later time t by P. At time t^*, the vehicle has a latitude of ϕ^*, a longitude of θ^*, and a velocity



Note: 0 - position at initial time,
P - position at subsequent time.

Figure VII-1.- Downrange and Crossrange Angles

heading of λ^*. At time t the vehicle is at latitude ϕ and longitude θ. The downrange angle (μ) and the crossrange angle (ν) shown in the illustration are measured along, and normal to, the great circle through O, and are inclined to the meridian by λ^*. From analytical geometry, ν and μ can be expressed as

$$\left.\begin{aligned}
\sin \nu &= -\sin \lambda^* \sin \phi^* \cos \phi_c \cos \theta' - \cos \lambda^* \cos \phi_c \sin \theta \\
&\qquad + \sin \lambda^* \cos \phi^* \sin \phi_c \\
\sin \mu &= \left(-\cos \lambda^* \sin \phi^* \cos \phi \cos \theta' + \sin \lambda^* \cos \phi_c \sin \theta' \right. \\
&\qquad \left. + \cos \lambda^* \cos \phi^* \sin \phi_c\right)/\cos \nu
\end{aligned}\right\} \text{(VII-3)}$$

$$\cos \mu = \left(\cos \phi^* \cos \phi \cos \theta' + \sin \phi^* \sin \phi_c\right)/\cos \nu,$$

where θ^* and λ^* can be defined in either of two ways:

1) The great circle to which ν and μ are referenced is fixed and rotating with the Earth. Then

$$\left.\begin{aligned}
\lambda^* &= \text{Earth's relative heading} = \sin^{-1} \frac{v_R}{\sqrt{u_R^2 + v_R^2}} \\
\theta' &= \theta - \theta^*;
\end{aligned}\right\} \text{(VII-4)}$$

2) The great circle to which μ and ν are referenced is inertially fixed, having the Earth rotating below it. Then

$$\begin{aligned}
\lambda^* &= \text{inertial heading} = \sin^{-1} \frac{v}{\sqrt{u^2 + v^2}} \\
\theta' &= \theta - \theta^* + \Omega_p (t - t^*).
\end{aligned} \qquad \text{(VII-5)}$$

Knowing ν and μ, and crossrange C_R and downrange D_R distances are

$$\left.\begin{aligned} C_R &= R_{ave}\,\nu \\ D_R &= R_{ave}\,\mu, \end{aligned}\right\} \quad \text{(VII-6)}$$

where R_{ave} is the average Earth radius between the initial and final points.

Auxiliary Position and Velocity Calculations

The solution from the translational equations is then used to calculate numerous output variables. The key variables directly computed from (x_I, y_I, z_I) and (V_{XI}, V_{YI}, V_{ZI}) summarized below.

$$\left.\begin{aligned}
r_I &= \text{geocentric radius} \\
&= \left(\underline{r}_I \cdot \underline{r}_I\right)^{1/2} \\
V_I &= \text{magnitude of the inertial velocity} \\
&= \left(\underline{V}_I \cdot \underline{V}_I\right)^{1/2} \\
\underline{V}_R &= \text{relative velocity} \\
&= \underline{V}_I - \underline{\Omega}_P \times \underline{r}_I \\
\underline{V}_A &= \text{atmospheric relative velocity} \\
&= \underline{V}_R + \underline{V}_W \\
V_R &= \text{magnitude of the relative velocity} \\
&= \left(\underline{V}_R \cdot \underline{V}_R\right)^{1/2} \\
V_A &= \text{magnitude of the atmospheric relative velocity} \\
&= \left(\underline{V}_A \cdot \underline{V}_A\right)^{1/2} \\
\underline{u}_{RI} &= \text{unit vector along radius vector} \\
&= \underline{r}_I / r_I \\
\underline{u}_{VI} &= \text{unit vector along inertial velocity vector} \\
&= \underline{V}_I / V_I \\
\gamma_I &= \text{inertial flight path angle} \\
&= \sin^{-1}\left[\underline{u}_{RI} \cdot \underline{u}_{VI}\right]
\end{aligned}\right\} \text{(VII-7)}$$

$$\left.\begin{aligned}
\gamma_R &= \text{relative flight path angle}\\
&= \sin^{-1}\left[\underline{u}_{RI}\cdot\underline{u}_{VR}\right]\\
\gamma_A &= \text{atmospheric relative flight path angle}\\
&= \sin^{-1}\left[\underline{u}_{RI}\cdot\underline{u}_{VA}\right]\\
\underline{V}_{IG} &= \text{inertial velocity in the G-frame}\\
&= [IG]\ \underline{V}_I\\
\underline{V}_{RG} &= \text{relative velocity in the G-frame}\\
&= [IG]\ \underline{V}_R\\
\underline{V}_{AG} &= \text{atmospheric relative velocity in the G-frame}\\
&= [IG]\ \underline{V}_A\\
A_{ZI} &= \text{inertial azimuth}\\
&= \tan^{-1}\left[V_{YG}/V_{XG}\right]\\
A_{ZR} &= \text{relative azimuth}\\
&= \tan^{-1}\left[V_{RYG}/V_{RXG}\right]\\
A_{ZA} &= \text{atmospheric relative azimuth}\\
&= \tan^{-1}\left[V_{AYG}/V_{AXG}\right]\\
\phi_c &= \text{geocentric latitude}\\
&= \sin^{-1}\left[z_I/r_I\right]\\
\theta_I &= \text{inertial longitude}\\
&= \tan^{-1}\left[y_I/x_I\right]
\end{aligned}\right\}\quad \text{(VII-7)}$$

$$\left.\begin{array}{ll}
\theta_R & = \text{relative longitude} \\
& = \theta_I - \Omega_P \left(t - t_0\right) \\
\underline{A}_{SB} & = \text{sensed acceleration in the B-frame} \\
& = \underline{A}_{TB} + \underline{A}_{AB} \\
A_S & = \text{magnitude of the sensed acceleration} \\
& = \left(\underline{A}_S \cdot \underline{A}_S\right)^{\frac{1}{2}} \\
\underline{A}_{SI} & = \text{sensed acceleration in the ECI-frame} \\
& = [IB]^{-1} \left[\underline{A}_{TB} + \underline{A}_{AB}\right]
\end{array}\right\} \quad \text{(VII-7)}$$

Auxilary Attitude Calculations

The attitude angles that are not used to generate the steering commands are computed for output in the auxiliary calculation subroutine. These equations are summarized below:

1) Aerodynamic angles:

$$\left.\begin{aligned} \alpha &= \tan^{-1}\left(V_{AZB}/V_{AXB}\right), \\ \beta &= \tan^{-1}\left(V_{AYB}\Big/\sqrt{V^2_{AXB} + V^2_{AZB}}\right), \\ \sigma &= \tan^{-1}\left(\frac{GB_{23} + \sin\beta \sin_A}{GB_{22}\cos A_{ZA} - GB_{21}\sin A_{ZA} \quad \cos\gamma_A}\right). \end{aligned}\right\} \quad \text{(VII-8)}$$

2) Inertial Euler angles:

$$\left.\begin{aligned} \phi_I &= \tan^{-1}\left(LB_{23}/LB_{22}\right), \\ \psi_I &= -\sin^{-1}\left(LB_{21}\right), \\ \theta_I &= \tan^{-1}\left(LB_{31}/LB_{11}\right); \end{aligned}\right\} \quad \text{(VII-9)}$$

3) Relative Euler angles:

$$\left.\begin{aligned} \psi_R &= \tan^{-1}\left(GB_{12}/GB_{11}\right), \\ \theta_R &= -\sin^{-1}\left(GB_{13}\right), \\ \phi_R &= \mathrm{ta} \quad \left(GB_{23}/GB_{33}\right). \end{aligned}\right\} \quad \text{(VII-10)}$$

Th~ ~elationship between the body rates and the attitude angles are:

1) Aerodynamic angles:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} a_1 + d_1\dot{\sigma} + (\sin\alpha)\,\dot{\beta} \\ a_2 + d_2\sigma + \dot{\alpha} \\ a_3 + d_3\dot{\sigma} - (\cos\alpha)\,\dot{\beta} \end{bmatrix} ; \qquad \text{(VII-11)}$$

2) Inertial Euler angles:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi}_I \cos\psi_I \cos\theta_I - \dot{\psi}_I \sin\theta_I \\ \dot{\theta}_I - \dot{\phi}_I \sin\psi_I \\ \dot{\phi}_I \cos\psi_I \sin\theta_I + \dot{\psi}_I \cos\theta_I \end{bmatrix} ; \qquad \text{(VII-12)}$$

3) Relative Euler angles:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = [GB] \begin{bmatrix} \frac{v}{r} \\ \frac{-u}{r} \\ \frac{-v}{r} \tan\phi_c \end{bmatrix} + \begin{bmatrix} \dot{\phi}_R - \dot{\psi}_R \sin\theta_R \\ \dot{\theta}_R \cos\phi_R + \dot{\psi}_R \sin\phi_R \cos\theta_R \\ \dot{\psi}_R \cos\phi_R \cos\theta_R - \dot{\theta}_R \sin\phi_R \end{bmatrix} \qquad \text{(VII-13)}$$

Tracking Data

POST computes tracking information for as many as ten tracking stations per phase. The tracking stations are located on a reference ellipsoid and are specified in terms of their latitude, longitude, and altitude above the ellipsoid. These variables are illustrated in figure VII-2.

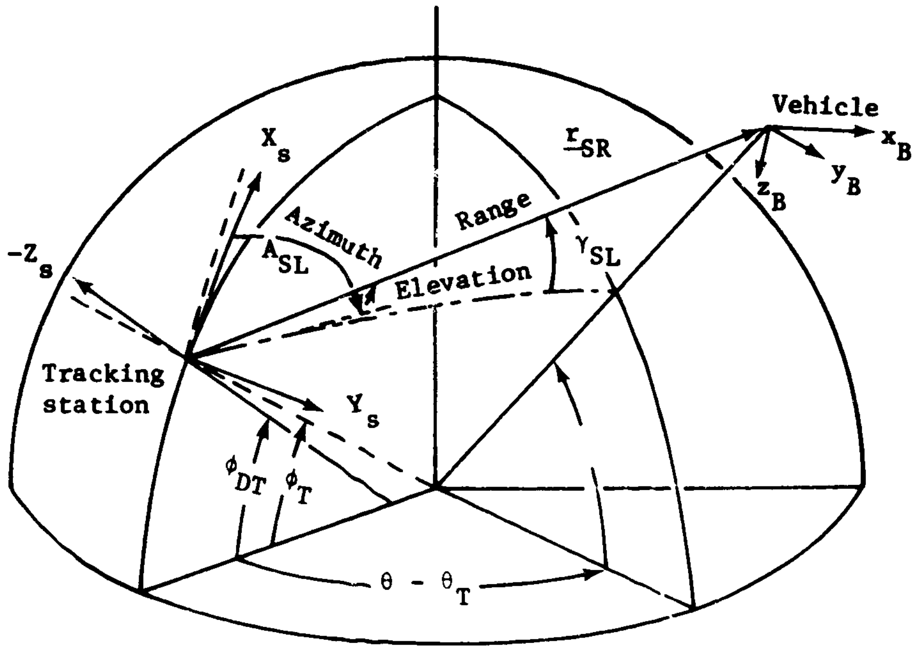


Figure VII-2.- Radar Tracking Schematic

The position components of the tracker in the Earth-relative frame are given by

$$\underline{r}_{TR} = \left(R_s + h_T\right) \begin{bmatrix} \cos \phi_T \cos \theta_T \\ \cos \phi_T \sin \theta_T \\ \sin \phi_T \end{bmatrix}, \tag{VII-14}$$

where h_T is the altitude of the tracker, ϕ_T the latitude of the tracker, and θ_T the longitude of the tracker.

The slant range vector in the ECI frame is given by

$$\underline{r}_{SR} = \underline{r}_I - [IP]^{-1}\, \underline{r}_{TR}, \tag{VII-15}$$

and the slant range is then computed as

$$r_{SR} = \sqrt{\underline{r}_{SR} \cdot \underline{r}_{SR}}. \tag{VII-16}$$

The elevation angle can then be computed as

$$\gamma_T = \sin^{-1} \left(\underline{u}_{TR} \cdot \underline{r}_{SR}/r_{SR}\right), \tag{VII-17}$$

where

$$\underline{u}_{TR} = [IP]^{-1}\, \underline{r}_{TR}/\left|[IP]^{-1}\, \underline{r}_{TR}\right|. \tag{VII-18}$$

The slant range vector, transformed to the geographic frame, is

$$\underline{r}_{SRG} = [IG]\, \underline{r}_{SR}, \tag{VII-19}$$

and thus the tracker's azimuth is given by

$$A_{ZT} = \tan^{-1} \left(y_{SRG}/x_{SRG}\right). \tag{VII-20}$$

The look angles are calculated from the slant range vector transformed to the body frame; i.e.,

$$\underline{r}_{SRB} = [IB]\, \underline{r}_{SR}. \tag{VII-21}$$

Using the components of $\underline{r}_{SRB}$, the cone angle is then given by

$$\psi_T = \cos^{-1}\left(x_{SRB}/r_{SR}\right), \qquad \text{(VII-22)}$$

and the clock angle is given by

$$\alpha_c = \tan^{-1}\left(y_{SRB}/z_{SRB}\right). \qquad \text{(VII-23)}$$

Space losses are calculated for the tracking stations as follows:

$$\left.\begin{aligned} SL_1 &= 36.56 + 20\ \mathrm{Log}_{10}\left(R_{SLM}\cdot FR_1\right)\\ SL_2 &= 36.56 + 20\ \mathrm{Log}_{10}\left(R_{SLM}\cdot FR_2\right)\\ SL_3 &= 36.56 + 20\ \mathrm{Log}_{10}\left(R_{SLM}\cdot FR_3\right), \end{aligned}\right\} \qquad \text{(VII-24)}$$

where

FR_1 = 420.0 (command frequency)

FR_2 = 2287.5 (telemetry frequency)

FR_3 = 5765.0 (tracking frequency)

R_{SLM} = slant range distance in statute miles.

Analytic Impact Calculations

The analytic impact calculations predict the geodetic latitude, longitude, and time of flight at impact for a vehicle with a given position and velocity to its intersection with the surface of the oblate planet. These calculations assume Keplerian motion and are not corrected for drag effects.

The basic problem in determining an impact point from a specified position and velocity $\left(\underline{r}_{IO}, \underline{V}_{IO}\right)$ is in calculating the impact eccentric anomaly. This angle is determined by iteratively solving the equation

$$r_I(E) = R_s\left(\phi_c\right) + h_{ip} \qquad \text{(VII-25)}$$

where h_{ip} is the desire: impact altitude above the oblate planet and the position vector is given by

$$\underline{r}_I(E) = f(E)\,\underline{r}_{I0} + g(E)\,\underline{V}_{I0}$$

$$f(E) = \left(\cos\left(E - E_0\right) - e\cos E_0\right) \Big/ \left(1 - e\cos E_0\right) \qquad \text{(VII-26)}$$

$$g(E) = \sqrt{\frac{a^3}{\mu}}\left(\sin\ E - E_0\right) - e\sin E + e\sin E_0\ .$$

Once the impact eccentric anomaly, E_{ip}, is determined, then the time, latitude, and longitude of impact are calculated as

$$t_{ip} = t_0 + \sqrt{\frac{a^3}{\mu}}\left(E_{ip} - E_0 - e\sin E_{ip} + e\sin E_0\right)$$

$$\phi_{g_{ip}} = \tan^{-1}\left(kz_{ip} \Big/ \sqrt{x_{ip}^2 + y_{ip}^2}\right) \qquad \text{(VII-27)}$$

$$\theta_{ip} = \tan^{-1}\left(\frac{y_{ip}}{x_{ip}}\right) - \Omega_p t_{ip}$$

Velocity Losses

There are two velocity loss options: (1) inertial velocity losses, and (2) atmospheric relative velocity losses. The inertial losses are used for orbital problems while the atmospheric relative losses are used for atmospheric flight problems.

The total change in the magnitude of the velocity is given by:

$$V_f - V_i = \Delta V = \Delta V^* - \Delta V_{TV} - \Delta V_{AERO} - \Delta V_g - \Delta V_{atm} \qquad \text{(VII-28)}$$

where

$\Delta V^* = \int \sum \eta_i T_{vac_i} / m \, dt$	=	ideal velocity
$\Delta V_{TV} = \int \frac{1}{m} \left(T_R - \underline{T}_R \cdot \underline{U} \right) dt$	=	thrust vector loss
$\Delta V_{AERO} = \int - \frac{1}{m} \underline{F}_A \cdot \underline{U}$	=	aerodynamic loss
$\Delta V_g = \int g \sin \gamma \, dt$	=	gravity loss
$\Delta V_{atm} = \int \frac{Pa}{m} \sum A_{Ei} \, dt$	=	atmospheric pressure loss

Inertial losses are computed when $\underline{U} = [IB]\underline{V}_I/V_I$, and similarly atmospheric relative losses are computed when $\underline{U} = [IB]\underline{V}_A/V_A$.

Velocity Margins

The program computes the amount of velocity margin available and the amount required, based on a percentage of the ideal velocity.

The velocity margin is calculated as

$$\Delta V_M = g_0 I_{SP} \ln \left[\left(W_G / W_G - W_p \right) \right] \qquad \text{(VII-29)}$$

The excess velocity is then given by

$$\Delta V_E = \Delta V_M - \Delta V_{\%} \sqrt{\sum V_i^2} \qquad \text{(VII-30)}$$

Sun-Shadow Calculations

The program computes several sun-shadow variables. These variables are used to calculate the sun-vehicle orientation angles, the sun-shadow conditions, and the position and velocity of the vehicle in the vernal equinox system. These auxilary variables are based upon the Greenwich hour angle (GHA), the Greenwich hour angle of the Sun (GHAS), the declination of the Sun δ_S, and the time of reference past midnight (TRPM). The Greenwich hour angle, the right ascension and declination of the Sun remain constant during the simulation.

The vehicle position and velocity vectors in the vernal equinox system are given by

$$\underline{r}_{VE} = \left[IV\right] \underline{r}_I \tag{VII-31}$$

and

$$\underline{V}_{VE} = \left[IV\right] \underline{V}_I . \tag{VII-32}$$

The Sun unit vector in the (VE) system is

$$\underline{u}_s = \begin{bmatrix} c\delta_s & c\Omega_s \\ c\delta_s & s\Omega_s \\ & s\delta_s \end{bmatrix} \tag{VII-33}$$

where

$$\Omega_s = GHA + GHAS.$$

The Sun unit vector in the ECI system is calculated as

$$\underline{u}_{sI} = [IV]' \underline{u}_s .$$

The cone and clock angles of the Sun vector in the body system are given as

$$\left.\begin{aligned} \delta_{cone} &= \cos^{-1}\left(\underline{u}_B \cdot \underline{u}_{sI}\right) \\ \delta_{clock} &= \tan^{-1}\left(y_{sB}/z_{sB}\right) \end{aligned}\right\} \tag{VII-34}$$

where $\underline{u}_B$ is a unit vector in the direction of the x_B axis, and

$$\begin{bmatrix} x_{sB} \\ y_{sB} \\ z_{sB} \end{bmatrix} = \left[IB\right] \underline{u}_{sI}. \qquad \text{(VII-35)}$$

The program also computes a shadow function, which is used to determine when the vehicle is in or out of the Earth's shadow. This function is based upon a cylindrical shadow model and is given by

$$S = \begin{cases} A - \tfrac{1}{2}\left(R_E + R_p\right) & : \text{ if } \underline{r}_I \cdot \underline{u}_s < 0. \\ -A + \tfrac{1}{2}\left(R_E + R_p\right) & : \text{ if } \underline{r}_I \cdot \underline{u}_{sI} \geq 0, \end{cases} \qquad \text{(VII-36)}$$

where

$$A = \sqrt{\underline{A} \cdot \underline{A}}$$

$$\underline{A} = \underline{r}_I - \left(\underline{r}_I \cdot \underline{u}_{sI}\right) \underline{u}_{sI}$$

If $S<0$, then the vehicle is in the shadow of the Earth. If $S \geq 0$, then the vehicle is in the sunlight. The vehicle is entering the shadow if $\dot{S} \leq 0$, and exiting if $\dot{S}>0$.

Multiple Vehicle

The program has the capability to simultaneously simulate the motion of two independent vehicles. One of the vehicles is active in the sense that it can be controlled using propulsion and/or aerodynamic forces. The other vehicle is passive in the sense that it cannot be controlled and is assumed to be out of the atmosphere and nonthrusting. As a result, the active vehicle is referred to as the pursuer, and the passive vehicle the target. The relative geometry between these two vehicles is defined in Figure VII-3.

A large number of output variables are calculated for the target vehicle. These variables are computed using equations that are identical to those used for the active vehicle. A complete list of these equations is given in Volume II - Utilization Manual. Only the key equations are in this section.

The increment in position, velocity, and acceleration between the active and the target vehicle are given by

$$\Delta \underline{r} = \underline{r}_I - \underline{r}_{It} = \Delta \underline{r}_o + \int \Delta \underline{V}\, dt$$

$$\Delta \underline{V} = \underline{V}_I - \underline{V}_{It} = \Delta \underline{V}_0 + \int \Delta \underline{a}\, dt \qquad \text{(VII-37)}$$

$$\Delta \underline{a} = \underline{a}_I - \underline{a}_{It}$$

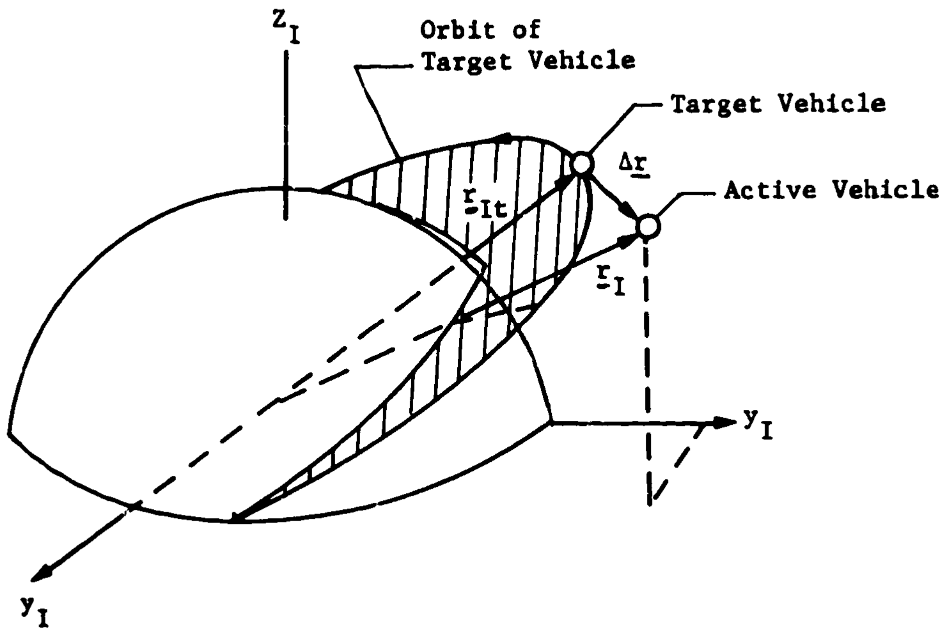


Figure VII-3.- Relative Geometry between Active and Target Vehicles

VIII. TARGETING AND OPTIMIZATION

POST uses an accelerated projected gradient algorithm (PGA) as the basic targeting/optimization technique. PGA is a combination of Rosen's projection method for nonlinear programming (refs. 3, 4, and 5) and Davidon's variable metric method for unconstrained optimization (ref. 6). The program also contains backup single-penalty function methods that use steepest descent, conjugate gradients, and/or the Davidon method. These standard gradient method are well documented in references 6 and 7 and are only briefly described in the following discussion.

The projected gradient algorithm is an iterative technique designed to solve a general class of nonlinear programming problems. PGA employs cost-function and constraint gradient information to replace the multidimensional optimization problem by an equivalent sequence of one-dimensional searches. In this manner, it solves a difficult multidimensional problem by solving a sequence of simpler problems. In general, at the initiation of the iteration sequence, PGA is primarily a constraint-satisfication algorithm. As the iteration process proceeds, the emphasis changes from constraint satisfaction to cost-function reduction. The logic used to effect this changeover process will be discussed below.

Since numerous analytical developments of this technique are available (see refs 3, 4, and 5), this presentation will primarily emphasize the geometrical aspects of the algorithm. This geometric interpretation clearly motivates the equations and logic contained in PGA, and a basic understanding of these concepts is usually sufficient to enable the user to efficiently use the algorithm.

Problem Formulation

The projected gradient method solves the following nonlinear programming problem:

Determine the values of the independent variables, $\underline{u}$, that minimize the cost function (optimization variable)

$$F(\underline{u}), \qquad \text{(VIII-1)}$$

subject to the constraints (dependent variables)

$$\underline{c}(\underline{u}) \geq \underline{0}, \qquad \text{(VIII-2)}$$

where $\underline{u} \in R^m$; $\underline{c}$ is a vector-valued function, i.e., $\underline{c}:R^m \rightarrow R^n$; and F is a scalar-valued function, i.e., $F:R^m \rightarrow R^1$.

The algorithm is actually more ver[illegible]tile than this simple formulation might indicate. In order to maximize any particular function, say $W(\underline{u})$, all that is required is to define $F(\underline{u}) = -W(\underline{u})$ and determine the minimum of $F(\underline{u})$. The equality constraint case is also contained within the above formulation since constraint equations of the form

$$c_j(\underline{u}) = 0 \qquad \text{(VIII-3)}$$

are special cases of Eq (VIII-2).

In the trajectory optimization, the cost function and the constraints are not explicitly a function of the independent variables, but rather depend explicitly on the state variables $\underline{r}_I$, $\underline{V}_I$, m, and Q. The explicity equations relating the state (dependent) variables to the independent variables are the integrals

$$\left.\begin{aligned} \underline{r}_I &= \underline{r}_{I_o} + \int \underline{V}_I \, dt \\ \underline{V}_I &= \underline{V}_{I_o} + \int \left[[IB]^{-1}[\underline{A}_{TB} + \underline{A}_{AB}] + \underline{G}_I\right] dt \\ m &= m_o + \int \dot{m} \, dt \\ Q &= \int \dot{Q} \, dt. \end{aligned}\right\} \text{(VIII-4)}$$

If $\underline{x}_n$ denotes the above state variables of the system being simulated at the nth event, and $\underline{x}_n^+$ and $\underline{x}_n^-$ denote the value of $\underline{x}_n$ on the plus and minus sides of that event, then

$$\underline{x}_{n+1}^- = \mathcal{T}_n\left[\underline{x}_n^+, \underline{u}_n\right], \tag{VIII-5}$$

where $\underline{u}_n$ are the independent variables in phase n, and $\mathcal{T}_n$ represent the solution of the state differential equations over phase n. The values of the state variables on the positive side of event n are then

$$\underline{x}_n^+ = \underline{x}_n^- + \Delta\underline{x}_n \tag{VIII-6}$$

where $\Delta\underline{x}_n$ represents the discontinuity in state (e.g., velocity impulse at the nth event).

The cost function and the trajectory constraints are computed at the positive side of the specified events, and are therefore given by

$$F(\underline{u}) = f\left(\underline{x}_{v_f}^+\right) \tag{VIII-7}$$

and

$$\underline{c}(\underline{u}) = \begin{bmatrix} \underline{c}_{v_1}\left(\underline{x}_{v_1}^+\right) \\ \cdot \\ \cdot \\ \cdot \\ \underline{c}_{v_j}\left(\underline{x}_{v_j}^+\right) \end{bmatrix}, \tag{VIII-8}$$

where v_f denotes the event at which the optimization variable is specified and v_j denotes the events at which the dependent variables are specified. This generality enables the program to solve problems in which intermediate constraints are defined, as well as problems where the cost function is not specified at the final event.

The trajectory propagator, T_n, can represent either numerical integration or analytical Keplerian equations.

Fundamental Concepts and Nomenclature

To facilitate the discussion of the projected gradient algorithm, the following nomenclature and basic concepts will be introduced.

A real k-dimensional Euclidean vector space is denoted by R^k, and $\underline{x}$ denotes a column matrix whose elements are x_i, where $i = 1, 2, \ldots, k$. The vector inequality $\underline{x} \geq 0$ implies $x_i \geq 0$ for each i, and A' denotes the transpose of the real matrix A.

The cost gradient is an m-vector of partial derivatives denoted as $\underline{\nabla F}$ or $\partial F/\partial \underline{u}$, and is defined as

$$(\underline{\nabla F})_j = \frac{\partial F}{\partial u_j}. \qquad \text{(VIII-9)}$$

The gradient to the ith constraint is similarly represented.

The Jacobian matrix of the constraint vector function with respect to the independent variable is a matrix whose ith row is the gradient vector $\underline{\nabla c}_i$. This matrix is denoted as

$$J(\underline{u}) = \frac{\partial \underline{c}}{\partial \underline{u}} \qquad \text{(VIII-10)}$$

and contains n rows and m columns. Clearly,

$$J_{ij} = \frac{\partial c_i}{\partial u_j}. \qquad \text{(VIII-11)}$$

The j^{th} constraint is said to be active at $\underline{u}$ if and only if

a) $$c_j(\hat{\underline{u}}) < 0, \qquad \text{(VIII-12)}$$

An active constraint is said to be unconstraining if and only if

b) $$c_j(\hat{\underline{u}}) = 0 \text{ and } r_j = \left[(SS')^{-1} S\underline{g}\right]_j \leq 0. \qquad \text{(VIII-13)}$$

Condition a) implies that the j^{th} constraint is either violated at $\hat{u}$, while b) indicates that the negative of the cost function gradient "points" outside the feasible region.

The *sensitivity matrix* is that matrix whose rows are the gradients to the active constraints, and is denoted by

$$S(\underline{u}) = \frac{\partial \underline{e}}{\partial \underline{u}}, \qquad \text{(VIII-14)}$$

where $\underline{e}$ is the n_a-vector of active constraints. Equality constraints are always active and thus are loaded into the upper elements of the $\underline{e}$. Thus, $\underline{e}$ is essentially the *error vector* for the active constraints. The *error function* is defined to be

$$E(\underline{u}) = \underline{e}'\underline{e}. \qquad \text{(VIII-15)}$$

The sensitivity matrix, S, is obtained from the Jacobian matrix, J, simply by deleting those rows that correspond to inactive constraints.

Corresponding to each constraint function $c_i(\underline{u})$ is a *boundary hypersurface*, B_i, defined by

$$B_i = \left\{\underline{u}: c_i(\underline{u}) = 0\right\}. \qquad \text{(VIII-16)}$$

Clearly, B_i is an m-1 dimensional nonlinear manifold. It can, however, be approximated at each point $\hat{\underline{u}}$ in R^m by an m-1 dimensional linear manifold

$$C_i(\hat{\underline{u}}) = \left\{\underline{u}: \nabla c_i'(\hat{\underline{u}})\ (\underline{u} - \hat{\underline{u}}) + c_i(\hat{\underline{u}}) = \underline{0}\right\}. \qquad \text{(VIII-17)}$$

The *feasible region* for the ith inequality constraint is the half-space in the independent-variable space defined by the set

$$R_i = \{\underline{u}; c_i(\underline{u}) \geq 0\}, \qquad \text{(VIII-18)}$$

while the complete feasible region for all of the constraints is

$$R = \bigcap_{i=1}^{n} R_i. \qquad \text{(VIII-19)}$$

The boundary of the complete feasible region must be

$$B(R) = \bigcup_{i=1}^{n} (B_i \cap R) \qquad \text{(VIII-20)}$$

The intersection in the preceding equation is required to select from the unbounded boundary, B_i, of the feasible region of the ith constraint that portion which is adjacent to the feasible region, R, for all of the constraints.

At any particular $\underline{\hat{u}} \epsilon R^m$ it is useful to define the *local boundary* hypersurface, $\underline{B}(\underline{\hat{u}})$, to the complete feasible region as the intersection of the active constraints at $\underline{\hat{u}}$. Let $N(\underline{\hat{u}})$ denote the set of indices of the $\hat{n}_a$ tight constraints at $\underline{\hat{u}}$. Then, symbolically,

$$B(\underline{\hat{u}}) = \bigcap_{i \epsilon K(\underline{\hat{u}})} B_i \qquad \text{(VIII-21)}$$

Clearly $B(\underline{\hat{u}})$ is an $m-k_a$ dimensional nonlinear manifold in the m-dimensional independent variable space.

An $m-k_a$ dimensional linear manifold $C(\underline{\hat{u}})$ approximating $B(\underline{\hat{u}})$ is the intersection of the active linearized constraints at $\underline{\hat{u}}$; that is,

$$C(\underline{\hat{u}}) = \bigcap_{i \varepsilon K(\underline{\hat{u}})} C_i(\underline{u}) \qquad \text{(VIII-22)}$$

$$= \left\{ \underline{u} : S(\underline{\hat{u}})(\underline{u} - \underline{\hat{u}}) + \underline{e}(\underline{\hat{u}}) = \underline{0} \right\} \qquad \text{(VIII-23)}$$

Now let $\tilde{Q}(\underline{\hat{u}})$ denote the linear space spanned by the gradients to the active constraints; that is,

$$\tilde{Q}(\underline{\hat{u}}) = \left\{ \underline{u} : \exists \ \alpha_1, \ldots, \alpha_{n_a} \text{ for which } \underline{u} = \sum_{i=1}^{k_a} \alpha_j \, S_{ij}(\underline{\hat{u}}) \ , \right\} \qquad \text{(VIII-24)}$$

and let $Q(\underline{\hat{u}})$ denote the orthogonal complement to $\tilde{Q}(\underline{\hat{u}})$; that is,

$$R^m = Q(\underline{\hat{u}}) \oplus \tilde{Q}(\underline{\hat{u}}). \qquad \text{(VIII-25)}$$

It can be shown that $Q(\underline{\hat{u}})$ is the unique linear space that can be translated to obtain the linear manifold $C(\underline{\hat{u}})$.

Furthermore there exist unique orthogonal projection operators $P(\underline{\hat{u}})$ and $\tilde{P}(\underline{\hat{u}})$ that resolve any vector in the independent-variable space into its corresponding components in $Q(\underline{\hat{u}})$ and $\tilde{Q}(\underline{\hat{u}})$, respectively; that is, for any $\underline{u} \varepsilon R^m$

$$\underline{u} = P(\underline{\hat{u}})\underline{u} + \tilde{P}(\underline{\hat{u}})\underline{u}, \qquad \text{(VIII-26)}$$

where

$$P(\underline{\hat{u}})\underline{u} \ \varepsilon \ Q(\underline{\hat{u}}) \ \text{ and } \ \tilde{P}(\underline{\hat{u}})\underline{u} \ \varepsilon \ \tilde{Q}(\underline{\hat{u}}). \qquad \text{(VIII-27)}$$

In particular,

$$\tilde{P} = S'(SS')^{-1} S \qquad \text{(VIII-28)}$$

and

$$P = I - \tilde{P}. \qquad \text{(VIII-29)}$$

An additional concept is the idea of problem scaling. The purpose of problem scaling is to increase the efficiency of the targeting/optimization algorithms by transforming the original problem into an equivalent problem that is numerically easier to solve.

To numerically scale a problem, two general types of scaling are required: (1) independent-variable scaling, and (2) dependent-variable scaling. Independent-variable scaling is accomplished by defining a positive diagonal scaling matrix, W_u, such that, the *weighted independent variables* are given by

$$\tilde{\underline{u}} = \left[W_u\right]\underline{u}. \qquad \text{(VIII-30)}$$

Simularly, dependent-variable weighting is accomplished by defining an optimization-variable scale factor, W_F, and a positive, diagonal, dependent-variable scaling matrix, W_e, such that the *weighted optimization variable* is

$$P_1 = W_F F(\underline{u}) \qquad \text{(VIII-31)}$$

and the *weighted dependent variables* are given by

$$\tilde{\underline{c}}(\underline{u}) = \left[W_e\right]\underline{c}\left(W_u^{-1}\,\tilde{\underline{u}}\right), \qquad \text{(VIII-32)}$$

yielding a *weighted error function*

$$P_2 = \tilde{\underline{e}}'\tilde{\underline{e}}\,(\gamma). \qquad \text{(VIII-33)}$$

The program contains several options for computing the independent-variable weighting matrix. However, the option most often used is the percentage scaling matrix

$$\left[W_u\right]_{ii} = \frac{1}{u_i}. \qquad \text{(VIII-34)}$$

The dependent-variable weighting matrix is always computed as the reciprocal of the constraint tolerances, and is given by

$$\left[W_e\right]_{ii} = \frac{1}{\varepsilon_i} , \qquad \text{(VIII-35)}$$

where ε_i is the tolerance for the i^{th} constraint. The optimization scale factor is merely input so that P_2 is approximately equal to one.

For simplicity, the following discussion of the algorithm assumes an appropriately scaled problem. However, the scaled equations can be obtained by making the following simple substitutions:

$\underline{u}$ replaced by $\tilde{\underline{u}}$

F replaced by P_1

$\underline{c}$ replaced by $\tilde{\underline{c}}$

P replaced by P_2

S replaced by $\left[W_e\right][S]\left[W_u\right]^{-1}$

$\underline{\nabla}F$ replaced by $W_F\, W_u^{-1}\, \underline{\nabla}F$.

The final key concept employed by PGA is the idea of a direction of search. Heuristically, the direction of search is nothing more than a particular line in the independent-variable space along which the constraint error is reduced, or along which the cost-function is decreased. In a more precise sense, the direction of search at $\underline{\hat{u}}$ is a half-ray emanating from $\underline{\hat{u}}$. Thus, for any positive scalar, γ, the equation

$$\underline{u} = \underline{\hat{u}} + \gamma\underline{\hat{s}} \qquad \text{(VIII-36)}$$

sets the limits of this half-ray and represents "movement" in the direction $\underline{\hat{s}}$ from $\underline{\hat{u}}$. This is illustrated in figure VIII-1.

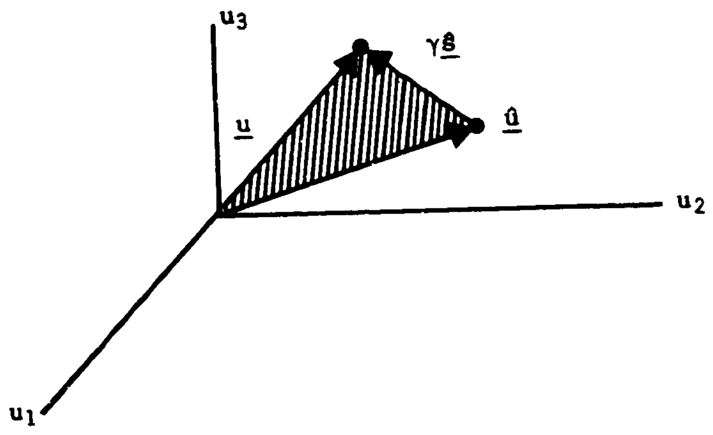


Figure VIII-1.- Direction of Search in the Independent-Variable Space

If $\underline{\hat{s}}$ is a unit vector, then γ represents the actual distance "moved" in the direction $\underline{\hat{s}}$. This concept of direction-of-search is particularly important since it enables the m-dimensional nonlinear programming problem to be replaced by a sequence (hopefully finite) of one-dimensional minimizations. What remains to be explained then is: (1) how to select the direction-of-search; and (2) how to determine the step size in that direction. All "direct" optimization methods employ this concept and, hence, differ only in their answers to the two preceding questions. The technique by which $\underline{s}_n$ and γ_n are selected by PGA will be described in subsequent sections.

Direction of Search

The projected gradient method uses two basic search directions. For this discussion, they will be termed the constraint and optimization directions, respectively. PGA proceeds by taking successive steps in one or the other of these two directions. The computation of each of these search directions is described below at a particular point $\underline{\hat{u}}$ in the independent-variable space where $\hat{n}_a$ of the constraints are active.

Constraint direction.- The constraint direction depends critically on the number of active constraints. Three cases are distinguished below:

1) Case 1.- If $\hat{n}_a < m$, then that unique control correction $\Delta\hat{u}$ is sought, which solves the linearized constraint equation

$$S(\underline{\hat{u}})\ \Delta\underline{u} + \underline{e}(\underline{\hat{u}}) = \underline{0} \tag{VIII-37}$$

and minimizes the length of $\Delta\underline{u}$. The solutions to the preceding vector equations define the $m-\hat{n}_a$ dimensional linear manifold $C(\underline{\hat{u}})$, which approximates the local boundary at $\underline{\hat{u}}$ as described in detail in the preceding section. The desired minimum norm correction, $\Delta\underline{\hat{u}}$, is then the vector of minimum length in the indepdendent-variable space from $\underline{\hat{u}}$ to the linear manifold $C(\underline{\hat{u}})$. Analytically, it is given as

$$\Delta\underline{\hat{u}} = -S'[SS']^{-1}\underline{e}(\underline{\hat{u}}). \tag{VIII-38}$$

This correction is illustrated in figure VIII-2.

The direction of search then is simply taken to be this minimum-norm correction to the locally active linearized constraints; that is,

$$\underline{s}^{c}(\underline{\hat{u}}) = \Delta\underline{\hat{u}}. \tag{VIII-39}$$

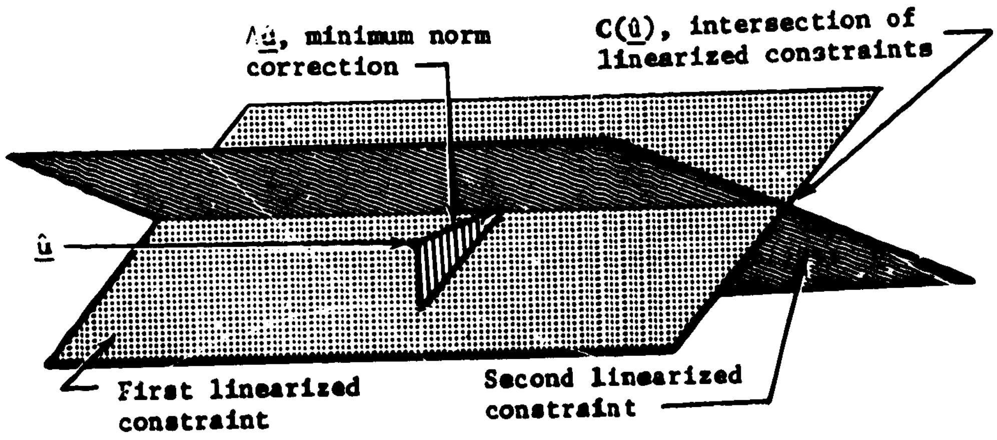

$\Delta\underline{\hat{u}}$, minimum norm correction
$C(\underline{\hat{u}})$, intersection of linearized constraints
$\underline{\hat{u}}$
First linearized constraint
Second linearized constraint

Figure VIII-2.- Illustration of Minimum-Norm Constraint, Direction for $\hat{n}_a = 2 < m = 3$

2) If $\hat{n}_a = m$, then the linearized local boundary $C(\underline{\hat{u}})$ reduces to a single point. Thus, there is a unique solution to the linearized constraint equations without the additional requirement that the length of the independent-variable correction be minimized. The minimum-norm correction formula then reduces to the familar Newton-Raphson formula for solving m equations in m unknowns; namely

$$\Delta\underline{\hat{u}} = - S^{-1}\, \underline{e}(\underline{\hat{u}}). \tag{VIII-40}$$

The Newton-Raphson correction is illustrated geometrically in figure VIII-3.

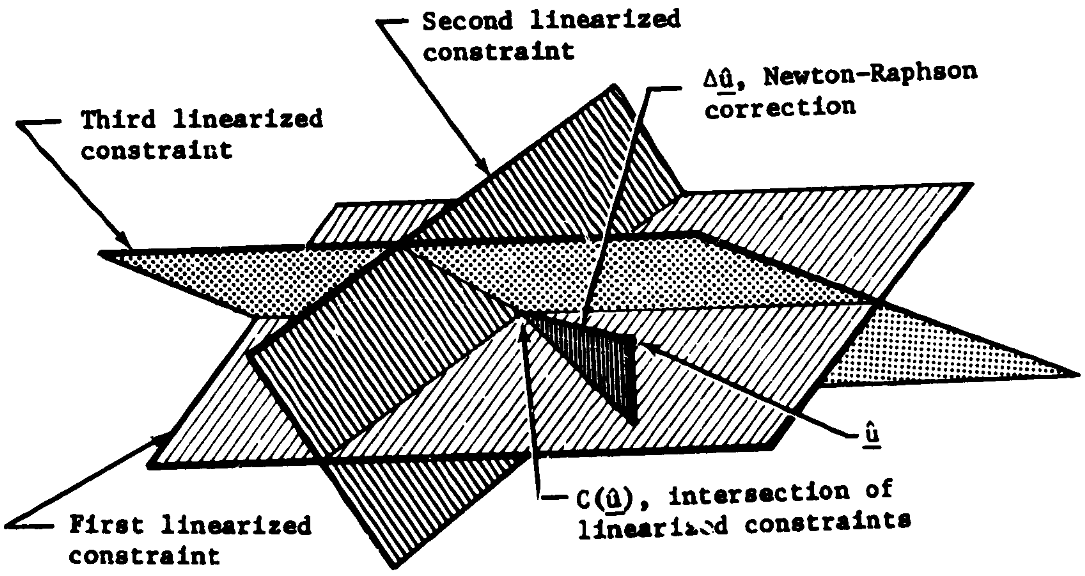

Second linearized constraint
$\Delta\underline{\hat{u}}$, Newton-Raphson correction
Third linearized constraint
$\underline{\hat{u}}$
$C(\underline{\hat{u}})$, intersection of linearized constraints
First linearized constraint

Figure VIII-3.- Illustration of Newton-Raphson Constraint, Direction for $\hat{n}_a = m = 3$

The direction of search is taken to be this unique correction vector satisfying the linearized constraints; that is,

$$\underline{s}^c(\underline{\hat{u}}) = \Delta\underline{\hat{u}}. \tag{VIII-41}$$

3) If $\hat{n}_a > m$, then $C(\underline{\hat{u}})$ is empty, since a simultaneous solution of all of the linearized constraint equations does not exist. Hence, an entirely new method for choosing the search direction must be devised. PGA deals with this problem by seeking the unique independent-variable correction $\Delta\underline{\hat{u}}$ that minimizes the sum of the squares of the deviations from the linearized constraints. Thus, the function

$$f(\Delta\underline{u}) = |S(\underline{\hat{u}})\ \Delta\underline{u} + \underline{e}(\underline{\hat{u}})|^2 \qquad \text{(VIII-42)}$$

is minimized with respect to $\Delta\underline{u}$. Gauss demonstrated that the formula for this "least squares" correction is

$$\Delta\underline{\hat{u}} = -(S'S)^{-1}\ S'\underline{e}(\underline{\hat{u}}). \qquad \text{(VIII-43)}$$

Figure VIII-4 illustrates the least-squares correction pictorially. As in the preceding two cases, the search direction is then taken to be this optimal correction; that is,

$$\underline{s}^c(\underline{\hat{u}}) = \Delta\underline{\hat{u}}. \qquad \text{(VIII-44)}$$

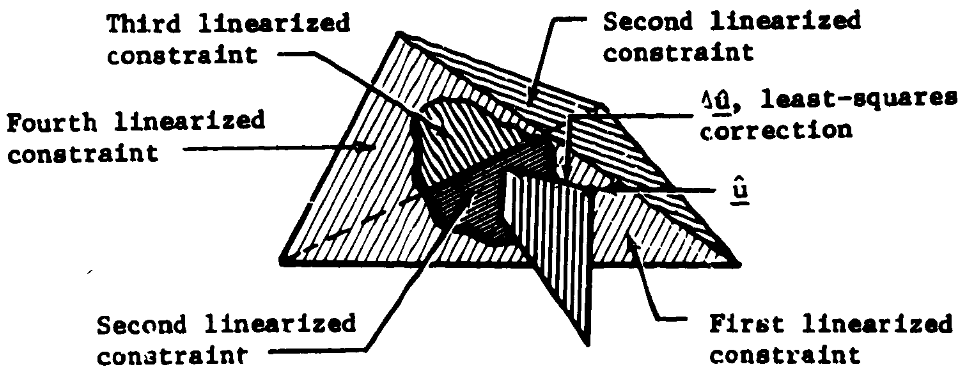

Third linearized constraint
Second linearized constraint
$\Delta\underline{\hat{u}}$, least-squares correction
Fourth linearized constraint
$\underline{\hat{u}}$
Second linearized constraint
First linearized constraint

Figure VIII-4.- Illustration of Least-Squares Constraint, Direct for $n_a = 4 > m = 3$

Optimization direction.- When the number of active constraints is less than the number of independent variables, it is then possible to reduce the nonminimal cost-function. Obviously the steepest descent direction, $-\underline{\nabla}F(\underline{\hat{u}})$, would be the best local search direction for reducing the cost function. Such a direction, however, would generally produce unacceptable constraint violations. To avoid this difficulty PGA orthogonally projects the unconstrained negative gradient, $-\underline{\nabla}F(\underline{\hat{u}})$, into a direction parallel to the local linearized constraint boundary $C(\underline{\hat{u}})$. By searching in the direction of this negative-projected gradient the algorithm can guarantee that there is no further constraint violation than that of $\underline{\hat{u}}$ for the case of linear constraints. To calculate this direction, it is only necessary to apply to the unconstrained negative gradient the projection operator $P(\underline{\hat{u}})$, which maps any vector in the independent-variable space into its component in $Q(\underline{\hat{u}})$, the unique linear space that can be translated into coincidence with the linear manifold $C(\underline{\hat{u}})$. Thus,

$$\left.\begin{aligned} \underline{s}^{o}(\underline{\hat{u}}) &= -P(\hat{u})\ \underline{\nabla}F(\underline{\hat{u}}) \\ &= -[I - \tilde{P}(\underline{\hat{u}})]\ \underline{\nabla}F(\underline{\hat{u}}) \\ &= -[I - S'\ (SS')^{-1}\ S(\underline{\hat{u}})]\ \underline{\nabla}F(\underline{\hat{u}}) \end{aligned}\right\} \quad \text{(VIII-45)}$$

The direction of search for the accelerated projected gradient method is

$$\underline{s}_n^{o}(\underline{\hat{u}}) = -H_n\ P\ \underline{\nabla}F(\underline{\hat{u}}) \qquad \text{(VIII-46)}$$

where

$$H_0 = I \qquad \text{(VIII-47)}$$

and

$$\left.\begin{aligned} H_n &= H_{n-1} + A_n + B_n, \quad \text{where} \quad n = 2 \\ A_n &= \left[\Delta\underline{x}_n\ \Delta\underline{x}_n'\right] \Big/ \Delta\underline{x}_n'\ \underline{g}_n, \\ B_n &= -\left[H_{n-1}\ \underline{g}_n\underline{g}_n'H_{n-1}'\right] \Big/ \underline{g}_n'H_{n-1}\underline{g}_n, \\ \Delta\underline{x}_n &= \underline{u}_n - \underline{u}_{n-1}, \\ \underline{g}_n &= \underline{\nabla}F\left(\underline{u}_n\right) - \underline{\nabla}F\left(\underline{u}_{n-1}\right). \end{aligned}\right\} \quad \text{(VIII-48)}$$

Figure VIII-5 illustrates the direction of the negative-projected gradient for the case of a single active constraint.

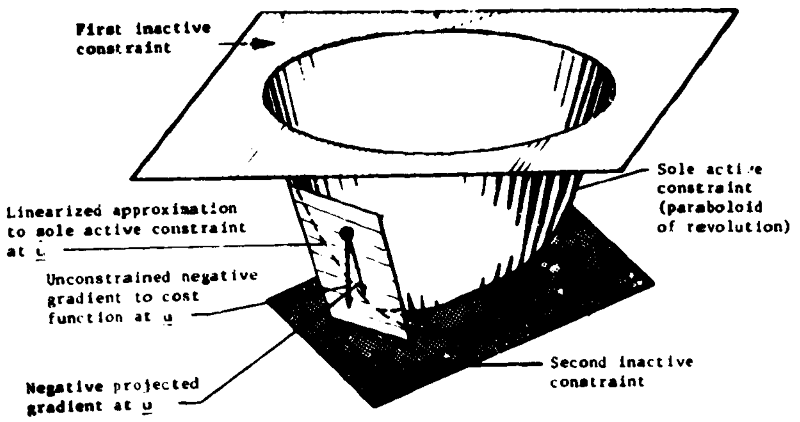

First inactive constraint
Linearized approximation to sole active constraint at $\underline{u}$
Unconstrained negative gradient to cost function at $\underline{u}$
Negative projected gradient at $\underline{u}$
Sole active constraint (paraboloid of revolution)
Second inactive constraint

Figure VIII-5.- Direction of Negative-Projected Gradient for $n_a = 1$ and $m = 3$ (Feasible region is that region inside paraboloid, above lower plane, and below upper plane; cost-function is vertical height)

If there are no equality constraints, and if all the inequality constraints are inactive, then S is the zero matrix and the direction of search becomes the standard deflected gradient direction

$$\underline{s}^o(\hat{\underline{u}}) = -H_n \underline{\nabla F}(\hat{\underline{u}}). \qquad \text{(VIII-49)}$$

Similarly, if the single-penalty-function methods are used, then the directions of search that minimize

$$P_2 = F + W \underline{e}'\underline{e} \qquad \text{(VIII-50)}$$

are:

1) Steepest-descent method

$$\underline{s}^o(\hat{\underline{u}}) = -\underline{\nabla}P_2(\hat{\underline{u}});$$

2) Conjugate gradient method (steepest-descent starter)

$$\underline{s}_n^o = -\underline{\nabla P}_2\, \underline{u}_n + \left[\frac{\underline{\nabla P}_2'\left(\underline{u}_n\right)\underline{\nabla P}_2\left(\underline{u}_n\right)}{\underline{\nabla P}_2'\left(\underline{u}_{n-1}\right)\underline{\nabla P}_2\left(\underline{u}_{n-1}\right)}\right]\underline{s}_{n-1}, \text{ where } n > 2,$$

3) Davidon's method (steepest-descent starter)

$$\underline{s}_n^o = -H_n\, \underline{\nabla P}_2\left(\underline{u}_n\right), \quad \text{where } n \geq 2$$

and

$$H_n = H_{n-1} + A_n + B_n,$$

where A_n and B_n have the same definitions as in the accelerated projected gradient mode.

Step-Size Calculation

At any particular point $\underline{\acute{u}}$ in the independent-variable space, the PGA algorithm proceeds by reducing the multidimensional problem to a one-dimensional search along the constraint direction to minimize the sum of the squares of the constraint violations, or along the optimization direction to minimize the estimated net cost-function. In either case, once the initial point $\underline{u}$ and the direction ofsearch $\underline{\hat{s}}$ are specified, the problem reduces to the numerical minimization of a function of a single variable--namely, the step size. PGA performs this numerical minimization via polynominal interpolation, based on function values along the search ray and the function's value and slope at the starting point. Consider then, in detail, the calculation of this latter pair of quantities for the respective functions associated with the constraint and optimization directions.

Constraint direction.- The function to be minimized along the constraint direction, $\underline{s}^c$, is the sum of the squares of the constraint violations; namely

$$h_c(\tau) = \underline{e}\left(\underline{\dot{u}} + \tau\,\underline{s}^c\right) \text{[illegible]}. \qquad \text{(VIII-51)}$$

Clearly

$$h_c(0) = |\underline{e}(\hat{\underline{u}})|^2. \qquad \text{(VIII-52)}$$

Differentiation via the chain rule yields

$$h_c'(0) = 2\underline{e}'(\hat{\underline{u}})S(\hat{\underline{u}})\hat{\underline{s}}^c. \qquad \text{(VIII-53)}$$

Recall that the search direction $\hat{s}^c$ was obtained as an independent-variable correction either satisfying all the linearized constraint equations if $\hat{n}_a \leq m$, or minimizing their violation if $m < \hat{n}_a$. Thus, if the constraints are reasonably linear, a good initial estimate for the γ minimizing h_c is one.

Optimization direction.- The function to be minimized along the optimization direction, $\hat{\underline{s}}^o$, is the estimated net cost-function which is defined as

$$h_o(\gamma) = \underbrace{F\left(\hat{\underline{u}} + \gamma\hat{\underline{s}}^o\right) - F(\hat{\underline{u}})}_{\substack{\text{change in cost-}\\ \text{function produced}\\ \text{by step of length}\\ \gamma \text{ along } \underline{s}^o}} + \underbrace{\underline{\nabla}'F(\hat{\underline{u}})\left[-S'(SS')^{-1}\,\underline{e}\left(\hat{\underline{u}} + \gamma\hat{s}^o\right)\right]}_{\substack{\text{linearized approximation to}\\ \text{change in cost-function re-}\\ \text{quired to perform minimum-}\\ \text{norm correction back to the}\\ \text{feasible region}}}. \qquad \text{(VIII-54)}$$

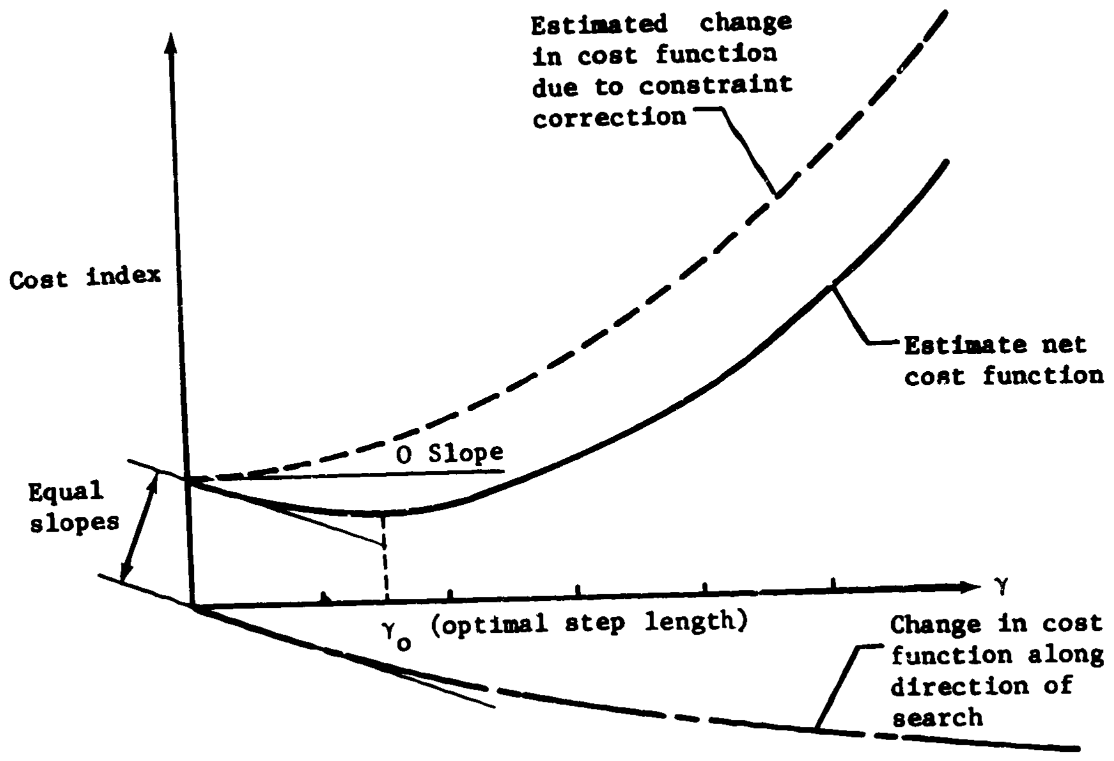
Clearly

$$h_o(0) = -\underline{\nabla}'F(\hat{\underline{u}})\,S'(SS')^{-1}(\hat{\underline{u}})\underline{e}(\hat{\underline{u}}). \qquad \text{(VIII-55)}$$

By expanding h_o in a Taylor series in γ about $\gamma = 0$, and by making use of the fact that $\tilde{P}\hat{\underline{s}}^o = \underline{0}$ since $\hat{\underline{s}}^o$ lies in $Q(\hat{\underline{u}})$, it can be shown that

$$h_o'(0) = \underline{\nabla}'F(\hat{\underline{u}})\,\hat{\underline{s}}^o. \qquad \text{(VIII-56)}$$

These properties of h_o are illustrated in figure 25.



Estimated change in cost function due to constraint correction
Cost index
Estimate net cost function
0 Slope
Equal slopes
γ
γ_o (optimal step length)
Change in cost function along direction of search

Figure VIII-6.- Properties of Estimated Net Cost Function

Both the constraint and optimization directions are based on a sensitivity matrix that depends critically on which constraints are active. Hence, for searches in either direction, it is important to limit the step size so that the set of active constraints does not grow. Such a limit can be obtained based on linear approximation and suffices to deal with inactive constraints becoming active.

The reverse situation--of active constraints becoming inactive--poses no difficulty. To see this, note that because of our treatment of the active constraints as linear manifolds, a first-order approximation of the distance to a particular active constraint boundary would not change along the optimization direction. Furthermore, along the constraint direction any change in the status of an active constraint will be appropriately treated by minimizing h_c with respect to the step length.

Let $K(\underline{\hat{u}})$ denote the set of active constraint indices at $\underline{\hat{u}}$, and let

$$r_k = \underline{s}'(\underline{\hat{u}})\underline{\nabla}c_k(\underline{\hat{u}}), \qquad \text{(VIII-57)}$$

where $\underline{s}(\underline{\hat{u}})$ is the search direction at vector $\underline{\hat{u}}$. Then assign to each k in K the number

$$\lambda(k) = \left\{ \begin{array}{ll} -c_k(\underline{\hat{u}})/r_k & \text{if } r_k < 0 \\ & \\ R & \text{if } r_k \geq 0 \end{array} \right\} \qquad \text{(VIII-58)}$$

where R is a very large real number. Then $\lambda(k)$ is a linear approximation to the distance along the search ray from $\underline{\hat{u}}$ to the boundary, B_k, of the kth constraint. Hence a resonable upper bound for the step length is

$$\lambda = \min_{k \varepsilon K} [\lambda(k)]. \qquad \text{(VIII-59)}$$

One-Dimensional Minimization

Monovariant minimization in PGA is performed exclusively by polynominal interpolation. First the actual function, f, to be minimized is fitted with one or more quadratic or cubic polynominals until a sufficiently accurate curve fit, p, is obtained; that is,

$$p(\gamma) = \sum_{i=0}^{n} a_i \gamma^i \approx f(\gamma) \quad \text{for all } \gamma \text{ of interest.} \qquad \text{(VIII-60)}$$

Then the independent variable value, γ^m, that minimizes f is approximated by the value, γ_p^m, which minimizes p. Clearly, γ_p^m can be determined analytically if $n \leq 3$.

The minimization routine makes ingenious use of all the information it accumulates about f to obtain a good curve fit. First, f is fitted with a quadratic polynominal, p_1, based on:

1) $f(0)$

2) $f'(0)$

3) $f\left(\gamma_0^m\right)$, where $\gamma_0^m > 0$ is an initial estimate of the γ value that minimizes f.

The coefficients of this quadratic polynominal are then calculated from the formulas:

$$\left.\begin{aligned} a_0 &= f(0) \\ a_1 &= f'(0) \\ a_2 &= \left[f\left(\gamma_0^m\right) - a_0\right] \Big/ \gamma_0^{m^2} + a_1 \Big/ \gamma_0^m . \end{aligned}\right\} \quad \text{(VIII-61)}$$

The value of the independent variable that minimizes this polynominal is

$$\gamma_1^m = -a_1/2a_2 . \qquad \text{(VIII-62)}$$

If γ_1^m and γ_0^m do not differ significantly, γ^m is taken to be γ_1^m and the minimization procedure is considered complete. Similarly, if $p_1\left(\gamma_1^m\right)$ is not significantly different from $f\left(\gamma_1^m\right)$, then γ^m is taken to be equal to γ_1^m and the process is terminated. Otherwise f is fitted with a cubic polynominal, p_2, based on

1) $f(0)$

2) $f'(0)$

3) $f\left(\gamma_0^m\right)$ and $\gamma_0^m > 0$

4) $f\left(\gamma_1^m\right)$.

If f is fitted using p_2, then coefficients are calculated from the following formulas:

$$\left.\begin{aligned}
a_0 &= f(0)\\
a_1 &= f'(0)\\
\lambda &= \max\left(\gamma_0^m, \gamma_1^m\right)\\
\alpha &= \min\left(\gamma_0^m, \gamma_1^m\right)\Big/\lambda\\
a_2 &= [\lambda a_1 \alpha + a_0 (1 + \alpha) + (\alpha^2 f(\lambda) - f(\alpha\lambda))/(1 - \alpha)]/(\lambda^3\alpha^2)\\
a_3 &= [(f(\alpha\lambda) - \alpha^3 f(\lambda))/(1 - \alpha) - \lambda\alpha(1 + \alpha)a_1 - (1 + \alpha = \alpha^2)a_0]/(\lambda^2\alpha^2)
\end{aligned}\right\} \quad \text{(VIII-63)}$$

The value of the independent variable, λ_2^m,- that minimizes this cubic polynomial is

$$\gamma_2^m = \left(-a_2 + \sqrt{a_2^2 - 3a_3a_1}\right) \Big/ 3a_3. \quad \text{(VIII-64)}$$

If γ_2^m and γ_1^m do not differ significantly, γ^m is taken to be γ_2^m and the minimization is stopped. Similarly, if $p_2\left(\gamma_2^m\right)$ is not significantly different from $f\left(\gamma_2^m\right)$, then γ^m is taken to be equal to γ_2^m and the procedure is terminated.

If none of these stopping conditions is met, a third quadratic curve-fit is attempted. The accumulated set of sample points on f, namely $[0, f(0)]$, $\left[\gamma_0^m, f\left(\gamma_0^m\right)\right]$, $\left[\gamma_1^m, f\left(\gamma_1^m\right)\right]$, and $\left[\gamma_2^m, f\left(\gamma_2^m\right)\right]$, is arranged in the order of their ascending abscissa values. Then the first point whose ordinate value is less than that of the following point is selected.

To simplify the notation in the following pages, relable this point as $[\gamma_2, f(\gamma_2)]$, the preceding point as $[\gamma_1, f(\gamma_1)]$, and the following point as $[\gamma_3, f(\gamma_3)]$.

Another quadratic polynomial, p_3, is then fitted to

1) $f(\gamma_1)$

2) $f(\gamma_2)$

3) $f(\gamma_.)$.

The formulas for these quadratic coefficients are as follows:

$$\left.\begin{aligned}
b_{ij} &= \gamma_i \gamma_j \\
c_{ij} &= \gamma_i + \gamma_j \\
d_{ij} &= \gamma_i - \gamma_j \\
a_0 &= \frac{b_{23}}{d_{12}d_{13}} f(\gamma_1) + \frac{b_{13}}{d_{21}d_{23}} f(\gamma_2) + \frac{b_{12}}{d_{31}d_{32}} f(\gamma_3) \\
a_1 &= \frac{c_{23}}{d_{12}d_{13}} f(\gamma_1) - \frac{c_{13}}{d_{21}d_{23}} f(\gamma_2) - \frac{c_{12}}{d_{31}d_{32}} f(\gamma_3) \\
a_2 &= \frac{1}{d_{12}d_{13}} f(\gamma_1) + \frac{1}{d_{21}d_{23}} f(\gamma_2) + \frac{1}{d_{31}d_{32}} f(\gamma_3).
\end{aligned}\right\} \quad \text{(VIII-65)}$$

The value of the independent variable that minimizes this quadratic is

$$\gamma_3^m = -a_1/2a_2. \qquad \text{(VIII-66)}$$

If γ_3^m and γ_2^m do not differ significantly, γ_m is taken to be γ_3^m and the search is discontinued. On the other hand, if $p_3\left(\gamma_3^m\right)$ is not significantly different from $f\left(\gamma_3^m\right)$, then γ^m is taken to be $\left(\gamma_3^m\right)$ and the process is terminated.

If neither of these stopping conditions is met, then a cubic polynomial is fitted to

1) $f(\gamma_1)$, γ_1

2) $f(\gamma_2)$, γ_2

3) $f(\gamma_3)$, γ_3

4) $f(\gamma_4)$, $\gamma_4 = \gamma_3^m$.

The formulas for these coefficients are as follows:

$$\left.\begin{aligned}
D_1 &= (\gamma_2 - \gamma_1)(\gamma_3 - \gamma_1)(\gamma_4 - \gamma_1) \\
D_2 &= (\gamma_1 - \gamma_2)(\gamma_3 - \gamma_2)(\gamma_4 - \gamma_2) \\
D_3 &= (\gamma_1 - \gamma_3)(\gamma_2 - \gamma_3)(\gamma_4 - \gamma_3) \\
D_4 &= (\gamma_1 - \gamma_4)(\gamma_3 - \gamma_4)(\gamma_3 - \gamma_4) \\
a_0 &= \frac{\gamma_2\gamma_3\gamma_4}{D_1} f(\gamma_1) + \frac{\gamma_1\gamma_3\gamma_4}{D_2} f(\gamma_2) + \frac{\gamma_1\gamma_2\gamma_4}{D_3} f(\gamma_3) + \frac{\gamma_1\gamma_2\gamma_3}{D_4} f(\gamma_4) \\
a_1 &= \frac{\gamma_2\gamma_3 + \gamma_2\gamma_4 + \gamma_3\gamma_4}{D_1} f(\gamma_1) + \frac{(\gamma_1\gamma_3 + \gamma_1\gamma_4 + \gamma_4\gamma_3)}{D_2} f(\gamma_2) \\
&\quad + \frac{(\gamma_1\gamma_2 + \gamma_1\gamma_4 + \gamma_2\gamma_4)}{D_3} f(\gamma_3) + \frac{(\gamma_1\gamma_2 + \gamma_1\gamma_3 + \gamma_2\gamma_3)}{D_4} f(\gamma_4) \\
a_2 &= \frac{(\gamma_2 + \gamma_3 + \gamma_4)}{D_1} f(\gamma_1) + \frac{(\gamma_1 + \gamma_3 + \gamma_4)}{D_2} f(\gamma_2) \\
&\quad + \frac{(\gamma_1 + \gamma_2 + \gamma_4)}{D_3} f(\gamma_3) + \frac{(\gamma_1 + \gamma_2 + \gamma_4)}{D_4} f(\gamma_4) \\
a_3 &= -\frac{1}{D_1} f(\gamma_1) - \frac{1}{D_2} f(\gamma_2) - \frac{1}{D_3} f(\gamma_3) - \frac{1}{D_4} f(\gamma_4).
\end{aligned}\right\} \quad \text{(VIII-67)}$$

The value of the indpendent variable minimizing this fourth cubic polynomial is

$$\gamma_4^m = (-a_2 \pm \sqrt{a_2^2 - 3a_3a_1})/3a_3. \qquad \text{(VIII-68)}$$

If γ_4^m and γ_3^m do not differ significantly, γ^m is taken to be γ_4^m and the minimization is stopped. Similarly, if $P_4\left(\gamma_4^m\right)$ is not significantly different from $f\left(\gamma_4^m\right)$, then γ^m is taken to be equal to γ_4^m and the procedure is terminated.

If none of these stopping conditions is met, the accumulated set of sample points is searched for the point with the minimum ordinate value. The abscissa value of this point is taken to be γ^m, and the minimization is considered complete.

Algorithm Macrologic

After being initialized the projected gradient algorithm proceeds as a sequence of iterations, each consisting of an optimization step followed by a constraint-correction step (see fig. VIII-7). The very first step from the user's initial independent-variable estimate is however, one of constraint correction. Furthermore, the optimization step is also omitted on any iteration for which the constraint-violation function, h_c, was not reduced by the constraint correction step of the preceding iteration.

The optimization search direction that emanates for $\underline{u}_n$ is based on the sensitivity matrix, $S\left(\underline{u}_n\right)$; that is,

$$\underline{s}_n^o = s^o\left(\underline{u}_n\right) = -P\underline{\nabla}F\left(\underline{u}_n\right), \qquad \text{(VIII-69)}$$

as discussed previously. Hence, $\underline{s}_n^o$ lies in the subspace $Q\left(\underline{u}_n\right)$.

The value of the independent-variable vector, $\underline{u}_n^o$, after the optimization is

$$u_n^o = \underline{u}_n + \gamma_o \underline{s}_n^o, \qquad \text{(VIII-70)}$$

where γ_o is the optimum step size.

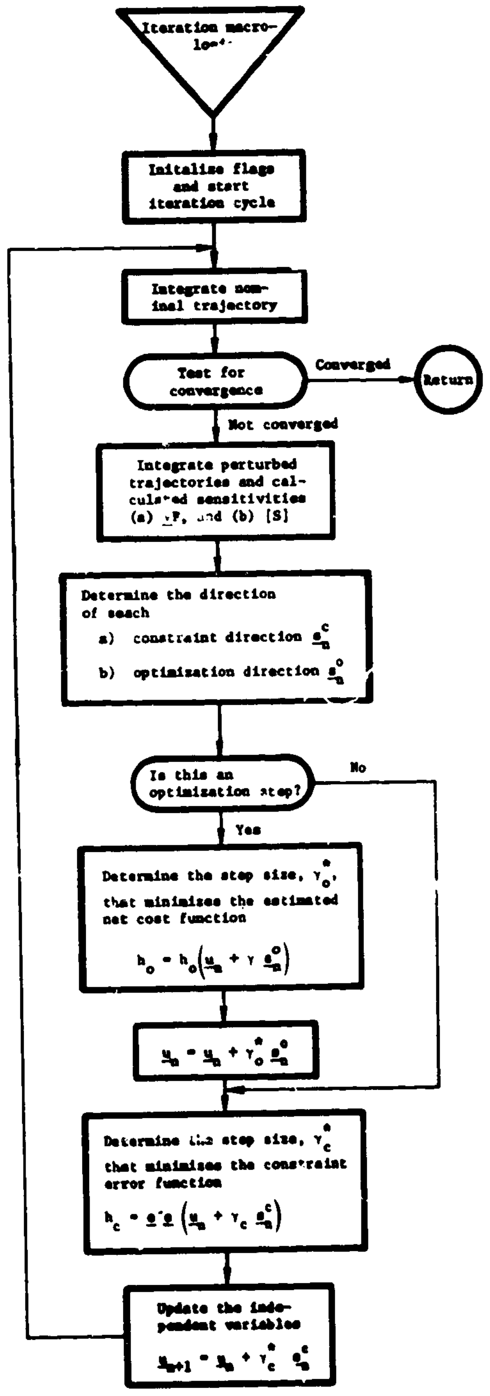

Iteration macro-logic
Initalize flags and start iteration cycle
Integrate nominal trajectory
Test for convergence
Converged
Return
Not converged
Integrate perturbed trajectories and calculated sensitivities (a) ∇F, and (b) [S]
Determine the direction of seach
a) constraint direction $\underline{s}_n^c$
b) optimization direction $\underline{s}_n^o$
Is this an optimization step?
No
Yes
Determine the step size, γ_o^*, that minimizes the estimated net cost function
$h_o = h_o\left(\underline{u}_n + \gamma\,\underline{s}_n^o\right)$
$\underline{u}_n = \underline{u}_n + \gamma_o^*\,\underline{s}_n^o$
Determine the step size, γ_c^* that minimizes the constraint error function
$h_c = \underline{e}'\underline{e}\left(\underline{u}_n + \gamma_c\,\underline{s}_n^c\right)$
Update the independent variables
$\underline{u}_{n+1} = \underline{u}_n + \gamma_c^*\ \underline{s}_n^c$

Figure VIII-7.- Macrologic of Projected Gradient Algorithm

The direction of the constraint-correction search emanates from $\underline{u}_n^o$; however, since generating a new sensitivity matrix is such an expensive calculation, the old Jacobian matrix, J, of the constraints with respect to the controls evaluated at $\underline{u}_n$ is used in conjunction with the error at $\underline{u}_n^o$. Thus,

$$\underline{s}_n^c = -S'(SS')^{-1}\left(\underline{u}_n\right)\underline{e}\left(\underline{u}_n^o\right). \qquad \text{(VIII-71)}$$

It can be shown by direct computation that

$$\tilde{P}\left(\underline{u}_n\right)\underline{s}_n^c = \underline{s}_n^c, \qquad \text{(VIII-72)}$$

where $\tilde{P}\left(\underline{u}_n\right)$ is based on $S\left(\underline{u}_n\right)$. Thus, $\underline{s}_n^c$ lies in the subspace $\tilde{Q}\left(\underline{u}_n\right)$ in the independent-variable space.

Since $Q\left(\underline{u}_n\right)$ and $\tilde{Q}\left(\underline{u}_n\right)$ are orthogonal complements, it follows that the optimization and constraint directions for any iteration are exactly orthogonal; that is,

$$\left(\underline{s}_n^o\right)' \underline{s}_n^c = 0. \qquad \text{(VIII-73)}$$

The result of the constraint correction step is then the independent-variable vector for the next iteration. Thus

$$\underline{u}_{n+1} = \underline{u}_n^o + \gamma_c \underline{s}_n^c. \qquad \text{(VIII-74)}$$

Figure VIII-8 geometrically illustrates a complete PGA iteration.

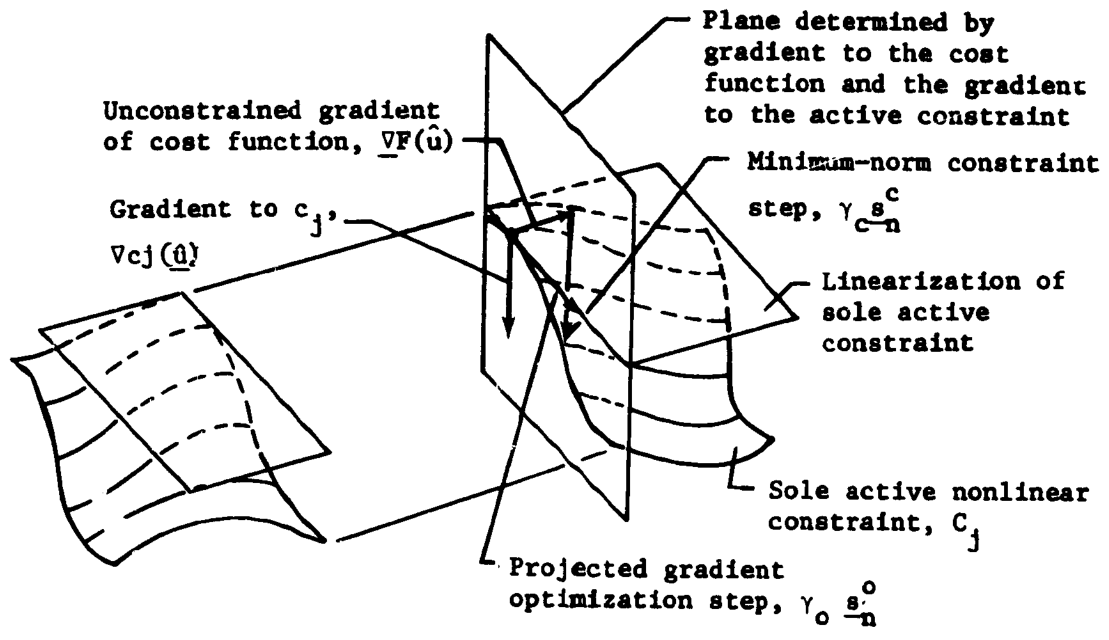

Plane determined by gradient to the cost function and the gradient to the active constraint
Unconstrained gradient of cost function, $\underline{\nabla F}(\hat{u})$
Minimum-norm constraint step, $\gamma_c \underline{s}_n^c$
Gradient to c_j, $\nabla cj(\hat{\underline{u}})$
Linearization of sole active constraint
Sole active nonlinear constraint, C_j
Projected gradient optimization step, $\gamma_o \underline{s}_n^o$

Figure VIII-8.- Complete PGA Iteration, Consisting of Optimization Step Followed by Constraint Step for $\hat{n}_a = 1$ and $m = 3$ (Feasible region is the unbounded region below the indicated nonlinear constraint manifold)

Finally, the algorithm has two stopping conditions. First, the search is stopped if the change in the cost function and the change in the length of the independent-variable vector between two successive iterations fall below their respective input tolerances; that is, if

$$\left.\begin{aligned} \left|F\left(\underline{u}_{n+1}\right) - F\left(\underline{u}_n\right)\right| &< \varepsilon_1 \\ \left|\underline{u}_{n+1} - \underline{u}_n\right| &< \varepsilon_2 . \end{aligned}\right\} \qquad \text{(VIII-75)}$$

Second, the procedure is discontinued if the number of the current iteration equals the maximum permissible number input by the user.

REFERENCES

1. D. E. Cornick, R. Stevenson, G. L. Brauer, and R. T. Steinhoff: Program to Optimize Shuttle Trajectories. MCR-71-731, prepared under Contract NAS1-10811. Martin Marietta Corporation, Denver, Colorado, 1971.

2. G. L. Kessler: Generalized N-Phase Trajectory Program (UD-213). Martin Marietta Corporation, Denver, Colorado, January 1971.

3. W. E. Wagner, and A. C. Serold: Formulation on Statistical Trajectory Estimation Program. NASA CR-1482, January 1970.

4. J. B. Rosen: The Gradient Projection Method for Nonlinear Programming. Part I - Linear Constraints. *J. Soc. Ind. Appl. Math.*, No. 3, 1967, pp 181-217.

5. J. B. Rosen: The Gradient Projection Method for Nonlinear Programming. Part II - Nonlinear Constraints. *J. Soc. Ind. Appl. Math.*, No. 3, 1961, pp 514-532.

6. B. A. Glassman, et al.: A Parameter Optimization Procedure for Multistage Vehicles. Vol. II. *AAS Science and Technology Series*, M. L. Anthony, ed, pp 223-241, 1967.

7. W. C. Davidon: Variable Metric Method for Minimization. Report No. ANL-5990 (Rev). Argonne National Laboratory, Oak Park, Illinois, 1959.

8. R. Fletcher and M. J. D. Powell: A Rapidly Convergent Descent Method for Minimization. *Computer J.*, July 1963.

9. F. T. Krogh: "Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations." TU Doc. CP2308, NPO-11643, Jet Propulsion Laboratory, Pasadena, California, 1969.

10. F. T. Krogh: "An Integrator Design." Technical Memo 33-479, Jet Propulsion Laboratory, Pasadena, California, 1971.

11. F. T. Krogh: "Algorithms for Changing the Step Size." *SIAM Journal on Numerical Analysis*. Vol 10, No. 5, October 1973.

APPENDIX A
DECOMPOSITION BY PARTITIONING INTO FULL-RANK SUBPROBLEMS

Consider a trajectory consisting of s mission segments each of which may consist of one or more phases. Suppose that each segment has its own physical control vector, $\underline{u}^k$, containing m_k components and its own constraint vector $\underline{c}^k$ having n_k components. Let $\underline{v}^k$ be the target value of the constraint vector for segment k. Finally, suppose there are n constraints C_i, which are best associated with the mission as a whole rather than any particular segment. Let V_i denote the target values of these constraints. The problem is then

$$\left.\begin{aligned}
&\text{minimize:}\\
&F\left(\biguplus_{k=1}^{s} \underline{u}^k\right) = \sum_{k=1}^{s} f^k\left(\biguplus_{\ell=1}^{k} \underline{u}^\ell\right)^{1}\\
&\text{subject to:}\\
&v_i^k = c_i^k\left(\biguplus_{\ell=1}^{k} \underline{u}^\ell\right) \quad \text{for } k=1,\dots,s\\
&\qquad\qquad\qquad\qquad\quad \text{for } i=1,\dots,n_k\\
&V_i = C_i\left(\biguplus_{k=1}^{s} \underline{u}^k\right) \quad \text{for } i=1,\dots,n.
\end{aligned}\right\} \qquad [1]$$

When numerous segments are present with varying degrees of influence on the overall objective, F, solution of problem [1] in a single piece by means of any existing equality-constraint minimization procedure becomes impractical if not impossible. By solving coordinated sets of subproblems representing the individual segments, the decomposition procedure is able to solve problem [1] routinely. The decomposition technique thus both imitates and extends the intuitive approach of the experienced trajectory designer.

---

^{1} The notation $\biguplus_{k=1}^{s}$ denotes the ordered union of the indexed quantity immediately to its right.

The process of decomposing a problem by partitioning it into full-rank subproblems is based upon two fundamental ploys. The first is the use in each segment of certain key control variables to satisfy the constraints of that segment. Thus segment k is made into a full-rank subproblem by designating n_k of the physical control vector components as subproblem variables and using them to satisfy the n_k constraints of that segment. The remaining control vector components of segment k are grouped together with similar variables from the other segments. This collection of controls, together with the overall mission objective, F, and constraints, $\underline{C}$, are made into a master problem of minimization subject to equality constraints. This partitioning of the original problem lends itself well to computation. The subproblems, which must be re-solved for each new choice of master problem controls, are solved by the highly efficient Newton-Raphson procedure. The master-problem, which serves to coordinate the subproblem solutions, uses the equally efficient but more time-consuming accelerated projected-gradient algorithm.

The second fundamental ploy is use of constraint target values of the various subproblems as master-problem independent variables. To obtain an optimum composite trajectory from a set of mission segments, the mission analyst typically varies the segment aim points parametrically and chooses the endpoint combination that results in the lowest overall cost. Indeed, in trajectory analysis the decision "where to go" is usually more important than "how to get there." By using subproblem constraint targets as master-problem controls the decomposition procedure automates the analyst's successful design approach.

The successful convergence of the decomposition procedure demands a reasonable partitioning of the original controls and constraints into the master-problem and subproblem categories. To be more precise, the subproblem controls and constraints must be so chosen that each subproblem will have a solution for any set of subproblem constraint target values that might reasonably arise during the master-problem iteration process. Thus, for a given subproblem, the controls that have the most substantial effect on that subproblem's constraint set should be chosen. Similarly if a particular constraint cannot be assigned to any subproblem whose controls can achieve its satisfaction, it should be designated a master-problem constraint. Finally, the number of master-problem constrains should be held to a minimum. Indeed the master-problem should be kept as simple as possible because each of its iterations requires the re-solution of all the subproblems.

The decomposition procedure maintains simulation flexibility in obtaining master-problem control sensitivity information by using numerical differencing. Solution of the master-problem by descent requires constrained derivatives--quotients of dependent perturbations in the master-problem objectives and constraints by independent perturbations in the master-problem controls assuming that the subproblem controls adjust uniquely to keep the subproblem targeted. These derivatives could be approximated by the numerical differencing of master-problem trajectories consisting of iteratively targeted subproblems. This approach, however, must be rejected because it is both susceptible to numerical error and demanding in computational effort. Instead formulas are used that relate the constrained derivatives to the partial derivatives of the master-problem objective and constraints with respect to all of the physical controls of the original problem. These partial derivatives are approximated conveniently by numercial differencing of the master-problem trajectories without subproblem targeting. Thus, the need for deriving variational equations for each simulated trajectory is eliminated for a reasonable computational price. Any trajectory that can be simulated, can be shaped with no additional analytical effort.

To precisely define the procedure, considerable nomenclature must be established. Most of the user supplied parameters have already been defined. Two, however, remain. The first is p_k, the number of subproblem constraint target values from segment k which are to be used as master-problem independent variables. The second is q_k, the number of master-problem constraints arising from segment k.

Consider next the procedure's working variables. To simplify notation, the subproblem and master-problem controls are given distinct literal symbols and resequenced. Indeed, let y_j^k denote the jth subproblem control arising from segment k and z_j^k be the jth master-problem control from that segment. Similarly, the segment constraint target values are assigned new symbols to distinguish those that are to be held fixed from those that are to be used as master-problem independent variables. Indeed, let r_i^k denote the ith constraint target value from segment k that is to be used as a master-problem independent variable, and t_i^k, the ith constraint target value to be held fixed for that segment. Finally, the master-problem constraint vector is resequenced so that its first q_1 components denoted by $\underline{C}^1$ arise from the first subproblem, and the next q_2 components denoted by $\underline{C}^2$ arise from the second subproblem, and so forth. In terms of this new notation, the original problem [1] becomes minimize:

$$F\left[\biguplus_{k=1}^{s}\left(\underline{r}^k \cup z^k\right)\right] = f\left[\biguplus_{k=1}^{s}\left(\underline{y}^k \cup z^k\right)\right]$$

subject to:

$$\underline{c}^\ell\left[\biguplus_{k=1}^{s}\left(\underline{r}^k \cup z^k\right)\right] = \underline{c}^\ell\left[\biguplus_{k=1}\left(\underline{y}^k \cup z^k\right)\right] \qquad \text{for } \ell=1,\ldots,s$$

where:

$\underline{r}^k$ is a master-problem independent variable

$\underline{z}^k$ is a master-problem independent variable

$\underline{y}^k\left[\biguplus_{k=1}^{s}\left(\underline{r}^k \cup \underline{z}^k\right)\right]$ is the unique vector of subproblem independent variables for subproblem k, satisfying that subproblem's constraint set for the current set of master-problem independent variables. [2]

To solve problem [2], a procedure must first be established for solving the subproblems, that is, for determining the unique $\underline{y}^k$ given the $\underline{r}^k$ and $\underline{z}^k$. As noted above, the Newton-Raphson algorithm for solving full-rank systems of nonlinear equations is the technique selected. To start the iterative solution, the user must input a good estimate, $\left(\underline{y}^k\right)_o$, for the physical-control vector of subproblem k, which approximately yield the constraint target values for that subproblem. The procedure, then, successively refines this estimate using the Newton-Raphson recursion formula

$$\left(\underline{y}^k\right)_{\nu+1} = \left[\underline{y}^k - \left(A^{kk}\right)^{-1}\left(\underline{c}^k - \underline{v}^k\right)\right]_\nu \qquad [3]$$

where

$$A^{\ell k} = \frac{\partial \underline{c}^k}{\partial \underline{y}^\ell} \qquad \begin{array}{l}\text{for } \ell=1,\ldots,s\\ \text{for } k=1,\ldots,\ell\end{array} \qquad [4]$$

After the first set of subproblems are solved, the converged subproblem control values from one subproblem set are used as the starting estimates for the next subproblem set. Further, the Jacobian matrix, A^{kk}, is updated from one iteration to the next only if the old Jacobian does not reduce the constraint errors in norm by a user specified fraction, ρ.

The equality constrained minimization that is the master-problem is carried out by the projected gradient algorithm already familiar to POST users. The only new technique involved is the computation of the constrained derivatives in terms of the partial derivatives of the master-problem objective and constraints with respect to all of the physical controls of the original problem. First, the perturbation of the subproblem independent variables caused by perturbations in the master-problem independent variables must be determined. Both master-problem physical controls and subproblem constraint target values must be considered. Once these constrained derivatives of the subproblem independent variables are determined, they can be used to calculate the desired constrained derivatives of the master-problem objective and constraints with respect to all of the master-problem independent variables.

The constrained derivatives of the subproblem controls are all derived from the basic subproblem equation

$$A^{\ell\ell}\,\underline{y}^{\ell} = \underline{r}^{\ell} - \sum_{k=1}^{\ell-1} A^{\ell k}\,\underline{y}^{k} - \sum_{k=1}^{\ell} B^{\ell k}\,\underline{z}^{k} \qquad \text{for } \ell=1,\ldots,s \qquad [5]$$

where

$$B^{\ell k} = \frac{\partial \underline{c}^{\ell}}{\partial \underline{z}^{k}} \qquad \begin{array}{l}\text{for } \ell=1,\ldots,s.\\ \text{for } k=1,\ldots,\ell\end{array} \qquad [6]$$

The constrained derivatives with respect to the master-problem physical controls are given by the equation

$$\frac{\delta \underline{y}^{\ell}}{\delta \underline{z}^{k}} = -\left(A^{\ell\ell}\right)^{-1}\left[B^{\ell k} + \sum_{o=k}^{\ell-1} A^{\ell o}\,\frac{\delta \underline{y}^{o}}{\delta \underline{z}^{k}}\right] \qquad \begin{array}{l}\text{for } \ell=1,\ldots,s\\ \text{for } k=1,\ldots,\ell.\end{array} \qquad [7]$$

The constrained derivatives with respect to the subproblem constraint target values are computed from the two formulas

$$\frac{\delta \underline{y}^k}{\delta \underline{r}^k} = \left(A^{kk}\right)^{-1} E_{n_k}^{p_k} \qquad \text{for } k=1,\ldots,s \tag{8}$$

where $E_{n_k}^{p_k}$ is the matrix consisting of the first p_k columns of the identity matrix of order n_k,

$$\frac{\delta \underline{y}^\ell}{\delta \underline{r}^k} = -\left(A^{\ell\ell}\right)^{-1} \sum_{o=k}^{\ell-1} A^{\ell o} \frac{\delta \underline{y}^o}{\delta \underline{r}^k} \qquad \begin{array}{l}\text{for } \ell=1,\ldots,s.\\ \text{for } k=1,\ldots,\ell-1\end{array} \tag{9}$$

The constrained derivatives of the master-problem objective with respect to both the master-problem physical controls and the subproblem constraint target values follow from the "chain-rule" for differentiation. They are computed as

$$\frac{\delta F}{\delta \underline{z}^k} = \sum_{\ell=k}^{s} \left[b^{\ell k} + \sum_{o=k}^{\ell} a^{\ell o} \frac{\delta \underline{y}^o}{\delta \underline{z}^k} \right] \qquad \text{for } k=1,\ldots,s, \tag{10}$$

and

$$\frac{\delta F}{\delta \underline{r}^k} = \sum_{\ell=k}^{s} \sum_{o=k}^{\ell} a^{\ell o} \frac{\delta \underline{y}^o}{\delta \underline{r}^k} \qquad \text{for } k=1,\ldots,s, \tag{11}$$

where

$$a^{\ell k} = \frac{\partial f^\ell}{\partial \underline{y}^k} \qquad \begin{array}{l}\text{for } \ell=1,\ldots,s\\ \text{for } k=1,\ldots,\ell,\end{array} \tag{12}$$

and

$$b^{\ell k} = \frac{\partial f^\ell}{\partial \underline{z}^k} \qquad \begin{array}{l}\text{for } \ell=1,\ldots,s.\\ \text{for } k=1,\ldots,\ell\end{array} \tag{13}$$

Finally, the constrained derivatives of the master-problem constraints with respect to the master-problem physical controls and the subproblem constraint target values follow from a straightforward application of the "chain rule." They are related to the appropriate partial derivatives by the equations

$$\frac{\delta \underline{C}^{\ell}}{\delta \underline{z}^{k}} = D^{\ell k} + \sum_{o=k}^{\ell} G^{\ell o} \frac{\delta \underline{y}^{o}}{\delta \underline{z}^{k}} \qquad \text{for } \ell=1,\ldots,s \quad \text{for } k=1,\ldots,\ell, \qquad [14]$$

and

$$\frac{\delta \underline{C}^{\ell}}{\delta \underline{r}^{k}} = \sum_{o=k}^{\ell} G^{\ell o} \frac{\delta \underline{y}^{o}}{\delta \underline{r}^{k}} \qquad \text{for } \ell=1,\ldots,s \quad \text{for } k=1,\ldots,\ell, \qquad [15]$$

where

$$D^{\ell k} = \frac{\partial \underline{C}^{\ell}}{\partial \underline{z}^{k}} \qquad \text{for } \ell=1,\ldots,s, \quad \text{for } k+1,\ldots,\ell, \qquad [16]$$

and

$$G^{\ell k} = \frac{\partial \underline{C}^{\ell}}{\partial \underline{y}^{k}} \qquad \text{for } \ell=1,\ldots,s \quad \text{for } k=1,\ldots,\ell. \qquad [17]$$