NASA CR-132741

MCR-74-423
NAS1-13300

Volume I

Final Report

November 1974

Formulation Manual

Six-Degree-of-Freedom Program to Optimize Simulated Trajectories (6D POST)

(NASA-CR-132741) SIX-DEGREE-OF-FREEDOM PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES (6D POST). VOLUME 1: FORMULATION MANUAL Final Report (Martin Marietta Corp.) 138 p HC $6.00 CSCL 22C G3/13

N76-13133

Unclas
03910

MARTIN MARIETTA

MCR-74-423
NAS1-13300

Volume I

Final
Report

November 1974

FORMULATION MANUAL

SIX-DEGREE-OF-FREEDOM PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES (6D POST)

Prepared by

G. L. Brauer, A. R. Habeger
and R. Stevenson

Approved

D. E. Cornick

D. E. Cornick
Program Manager

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

FOREWORD

This final report describing the formulation of the Six-Degree-of-Freedom Program to Optimize Simulated Trajectories (6D POST) is provided in accordance with Part 3.0 of NASA Contract NAS1-13300. The report is presented in three volumes as follows:

Volume I - 6D POST - Formulation Manual; NASA CR-132741

Volume II - 6D POST - Utilization Manual; NASA CR-132742

Volume III - 6D POST - Programmer's Manual. NASA CR-132743

This work was conducted under the direction of Mr. Howard Stone and Mr. Richard Powell of the Space Systems Division, National Aeronautics and Space Administration, Langley Research Center.

CONTENTS

SUMMARY

This report documents the basic equations and models used in the six-degree-of-freedom version of the program to optimize simulated trajectories (6D POST).

6D POST, a direct extension of the point mass version of POST, is a general purpose rigid body six-degree-of-freedom program. The program can be used to solve a wide variety of atmospheric flight mechanics and orbital transfer problems for powered or unpowered vehicles operating near a rotating oblate planet. The principal features of 6D POST are: an easy to use NAMELIST type input procedure, an integrated set of Flight Control System (FCS) modules, and a general-purpose discrete parameter targeting and optimization capability.

6D POST is written in FORTRAN IV for the CDC 6000 series 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 - Programmers Manual - Documents the program structure and logic, subroutine descriptions, and other pertinent programming information.

Tables

I. INTRODUCTION

The six (6)-degree-of-freedom program to optimize simulated trajectories is a general purpose FORTRAN program for simulating rigid body trajectories of aerospace type vehicles. The program can be used to solve a wide variety of performance, guidance, and flight control problems for atmospheric and orbital vehicles. For example, typical applications of 6D POST include:

1) Guidance and flight control system simulation and analysis;

2) Loads and dispersion type analyses;

3) General-purpose 6D simulation of controlled and uncontrolled vehicles;

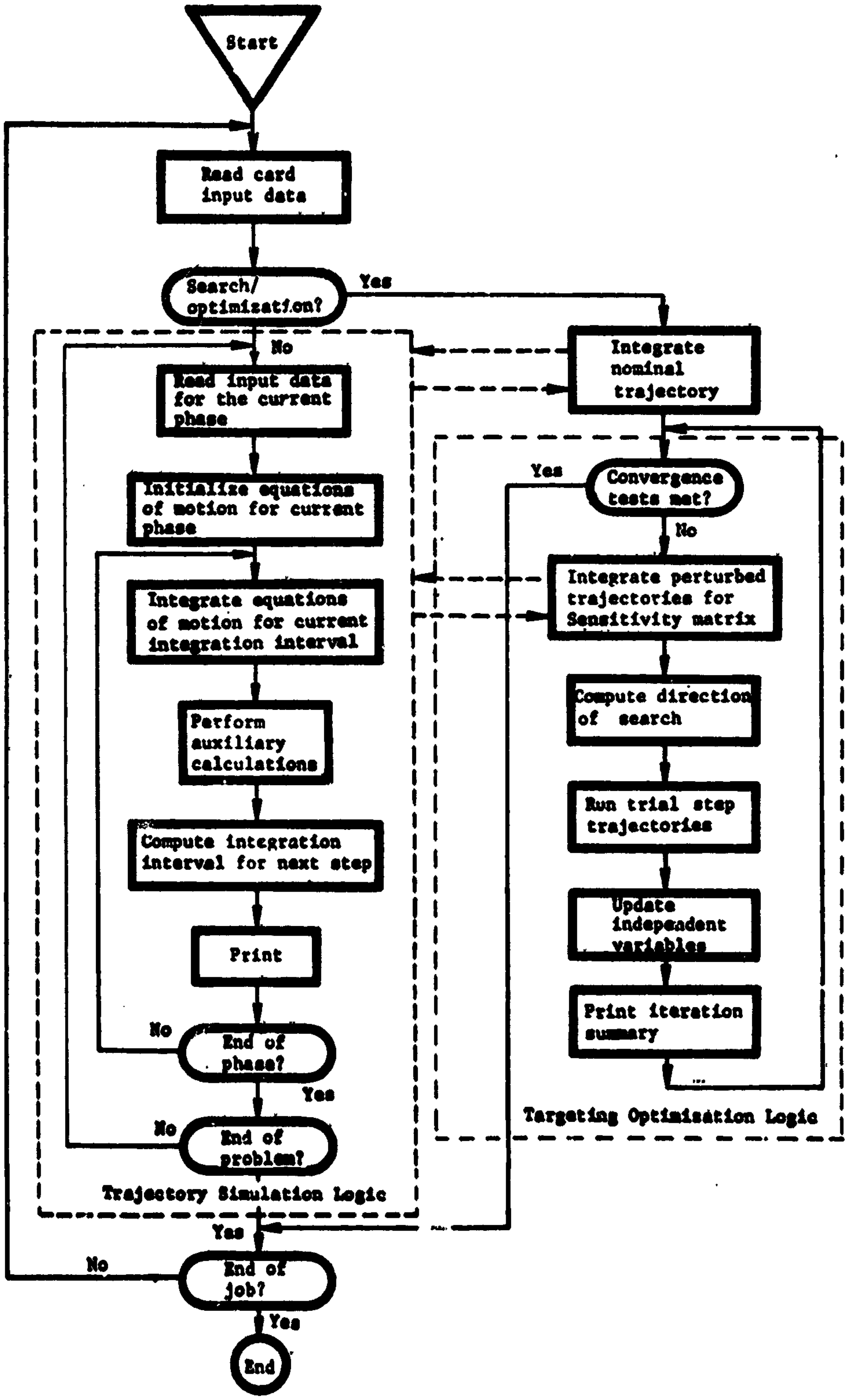
4) 6D performance validation.

One of the key features of 6D POST is an easy to use NAMELIST-type input procedure. This feature significantly reduces input deck set-up time (and costs) for 6D studies that require the normal large amount of input data. In addition, the general applicability of 6D 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 impact of the control system design on the performance characteristics of aerospace vehicles.

The [illegible]ic 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, an airframe model, a navigation and guidance model, and a flight control system 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 unconstrained 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.

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



ORIGINAL PAGE IS OF POOR QUALITY

Figure I-1.- Program Macrologic

II. LIST OF SYMBOLS AND ABBREVIATIONS

Math symbol	Internal Fortran symbol	Definition
a	SEMJAX	semimajor axis, m (ft)
$\underline{A}_{AB} = (A_{AXB}, A_{AYB}, A_{AZB})$	---	aerodynamic acceleration in the body frame, mps^2 (fps^2)
[AB]	AB(I)	matrix transformation from the A-frame to the B-frame
A_E	AR	nozzle exit area of each rocket engine, m^2 (ft^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)
$\underline{A}_{SB} = (A_{SXB}, A_{SYB}, A_{SZB})$	AXB, AYB, AZB	total sensed acceleration in the body frame, mps^2 (fps^2)
$\underline{A}_{SI} = (A_{SXI}, A_{SYI}, A_{SZI})$	ASXI, ASYI, ASZI	total sensed acceleration in the inertial frame, mps^2 (fps^2)
$\underline{A}_{TB} = (A_{TXB}, A_{TYB}, A_{TZB})$	---	thrust acceleration in the body frame, 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)

Math symbol	Internal Fortran symbol	Definition
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 i^{th} constraint
B_n	B(I)	Davidon deflection matrix
B(R)	---	boundary of region R
$B(\underline{\hat{u}})$	---	local boundary hypersurface
C_A, C_Y, C_N	CA, CY, CN	axial, side force, and normal aerodynamic force coefficients
C_{A_0}, C_{Y_0}, C_{N_0}	---	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_0}, C_{L_0}	---	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
$\underline{G}_I = (G_{XI}, G_{YI}, G_{ZI})$	GXI, GYI, GZI	total gravitational acceleration in the ECI-frame, mps^2 (fps^2)

Math symbol	Internal Fortran symbol	Definition
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, [illegible])
$\underline{h} = (h_{XI}, h_{YI}, h_{ZI})$	ANGMØM	[illegible] momen[illegible], mps^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_O	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
$\underline{k} = (k_1, k_2, k_3, k_4)$	---	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)
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)
$\underline{s}$	S(I)	direction of search
$\underline{s}^c$	---	direction of search to satisfy the constraints
$S_{\ell oss_i}$	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
$T^n(y)$	---	denotes n^{th} 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
$\underline{u}_{VI}$	---	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)
$\underline{V}_R$	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
$\underline{V}_{WI} = (V_{WXI}, V_{WYI}, V_{WZI})$	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 n^{th} 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
γ_I, γ_R, γ_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)

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)
$\dot{\theta}_{max}$	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	---	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 n^{th} 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

Math symbol	Internal Fortran symbol	Definition
$(\ )^+$		denotes occurrence at the positive side of an event
$(\ )^-$		denotes occurrence at the negative side of an event
ϵ		is a member of
$\cap$		intersection of
$\cup$		union of
$\{A:B\}$		set
$\ni$		such that
$\oplus$		addition operator

The following symbols have been added to the six-degree-of-freedom portion of the program.

Math symbol	Internal Fortran symbol	Definition
$C_{A\delta a}$, $C_{A\delta e}$, $C_{A\delta R}$, $C_{A\delta f_i}$	CADA, CADE, CADR, CAF(I), I=1,3	Incremental axial force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector
$C_{D\delta a}$, $C_{D\delta e}$, $C_{D\delta R}$, $C_{D\delta f_i}$	CDDA, CDDE, CDDR, CDF(I), I=1,3	Incremental drag force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector
$C_{\ell\delta a}$, $C_{\ell\delta e}$, $C_{\ell\delta R}$, $C_{\ell\delta f_i}$	CLLDA, CLLDE, CLLDR CLLF(I), I=1,3	Incremental rolling moment force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector

Math symbol	Internal Fortran symbol	Definition
$C_{L\delta a}$, $C_{L\delta e}$, $C_{L\delta R}$, $C_{L\delta f_i}$	CLDA, CLDE, CLDR, CLF(I), I=1,3	Incremental lift force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector
$C_{m\delta a}$, $C_{m\delta e}$, $C_{m\delta R}$, $C_{m\delta f_i}$	CMDA, CMDE, CMDR, CMF(I), I=1,3	Incremental pitching moment force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector
$C_{n\delta a}$, $C_{n\delta e}$, $C_{n\delta R}$, $C_{n\delta f_i}$	CWDA, CWDE, CWDR, CWF(I), I=1,3	Incremental yawing moment force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector
$C_{N\delta a}$, $C_{N\delta e}$, $C_{N\delta R}$, $C_{N\delta f_i}$	CNDA, CNDE, CNDR, CNF(I), I=1,3	Incremental normal force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector
$C_{Y\delta a}$, $C_{Y\delta e}$, $C_{Y\delta r}$, $C_{Y\delta f_i}$	CYDA, CYDE, CYDR, CYF(I), I=1,3	Incremental side force coefficient per rad (deg) for the aileron, elevator, rudder, and general deflector
d_R, d_P, d_Y	DREFR, DREFP, DREFY	Reference lengths for roll, pitch, and yaw aerodynamic moment coefficients m (ft)
$\underline{F}_B$	FTTXB(I), I=1,3	Total of all nongravitational forces calculated in the body frame, N (lb)
$\underline{F}_{RCS}$	RCSFXB(I), I=1,3	Force of the RCS engines in the body frame, N (lb)
$\underline{F}_{TB}$	FTXB(I), I=1,3	Total force due to the non-RCS engines calculated in the body frame, N (lb)

Math symbol	Internal Fortran symbol	Definition
I_{XX}, I_{YY}, I_{ZZ}	IXX, IYY, IZZ	Moments of inertia about the body axis system
$\dot{I}_{XX}$, $\dot{I}_{YY}$, $\dot{I}_{ZZ}$, $\dot{I}_{XY}$, $\dot{I}_{YZ}$, $\dot{I}_{XZ}$	IXXD, IYYD, IZZD, IXYD, IYZD, IXZD	Time derivations of the moments and products of inertia
I_{XY}, I_{XZ}, I_{YZ}	IXY, IXZ, IYZ	Products of inertia about the body axis system
[K]	KRDP(I), KPDP(I), KRDY(I), KYDY(I)	Roll nozzle deflection matrix
$K_{P\delta a}$, $K_{P\delta e}$, $K_{P\delta r}$	KPDA, KPDE, KPDR	Mixing logic gains for the aileron, elevator, and rudder about the y-body (pitch) axis
$K_{R\delta a}$, $K_{R\delta e}$, $K_{R\delta r}$	KRDA, KRDE, KRDR	Mixing logic gains for the aileron, elevator, and rudder about the x-body (roll) axis
$K_{Y\delta a}$, $K_{Y\delta e}$, $K_{Y\delta r}$	KYDA, KYDE, KYDR	Mixing logic gains for the z-body (yaw) axis
[M]	---	Matrix representation of the mixing gains
$\underline{M}_{AB}$	AMXB(I), I=1,3	External moment due to the aerodynamic forces, N-m (ft-lb)
$\underline{M}_{B}$	TTMXB(I), I=1,3	Total external moment due to thrust, RCS, and aerodynamic forces, N-m (ft-lb)
$\underline{M}_{RCS}$	RCSMXB(I), I=1,3	Net moment due to the RCS forces, N-m (ft-lb)
$\underline{M}_{TB}$	TMXB(I), I=1,3	External moment due to thrust forces, N-m (ft-lb)

Math symbol	Internal Fortran symbol	Definition
p', q', r'	PND, QND, RND	Nondimensional roll, pitch, and yaw body rates
$\alpha_I, \beta_I, \sigma_I$	ALPHI, BETAI, BANKI	Attitude reference angles measured in the same sense and order as the aerodynamic angles but with respect to the inertial velocity vector, rad (deg)
$\underline{\delta}, \underline{\delta}_0$	---	General vector representing an engine or aerodynamic control surface deflection, rad (deg). the subscript denotes the null value.
$\delta a, \delta e, \delta_r, \delta f_i$	DELA, DELE, DELR DELF(I), I-1,3	Deflection angles for the aileron, elevator, rudder, and general aerodynamic control surfaces rad (deg)
$\delta e_{p_i}, \delta e_{y_i}$	DEP(I), DEY(I)	Pitch and yaw gimbal angles for the i-th engine, rad (deg)
$\delta\underline{\theta} = (\delta\psi, \delta\theta, \delta\phi)$	YAWAC, PITAC, RØLLAC	Yaw, pitch, and roll autopilot commands, rad (deg)
$\Delta\underline{R}_{AB}$	DXR(I), I=1,3	Vector difference between the center of gravity and the reference point for the aerodynamic forces (usually the aerodynamic center of pressure), m (ft)
$\Delta\underline{R}_{BI_i}$	DXP(I), DYP(I), DZP(I)	Vector difference between the center of gravity and the engine gimball point for the i-th engine, m (ft)
x_{gp}, y_{gp}, z_{gp}	GXP, GYP, GZP	Location of engine gimbal in body reference system

III. COORDINATE SYSTEMS

6D 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

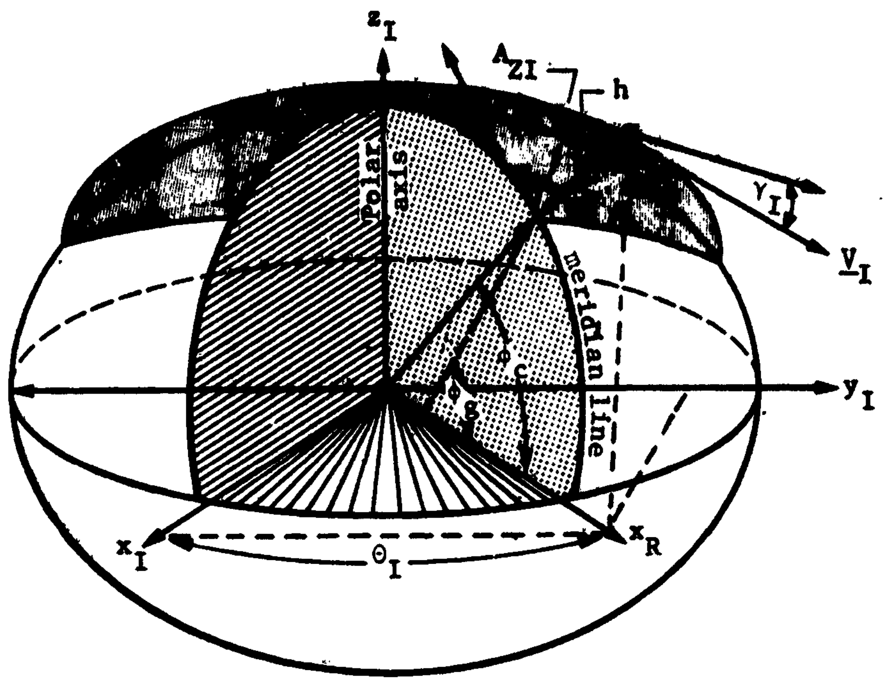


Figure III-1.- Coordinate Systems

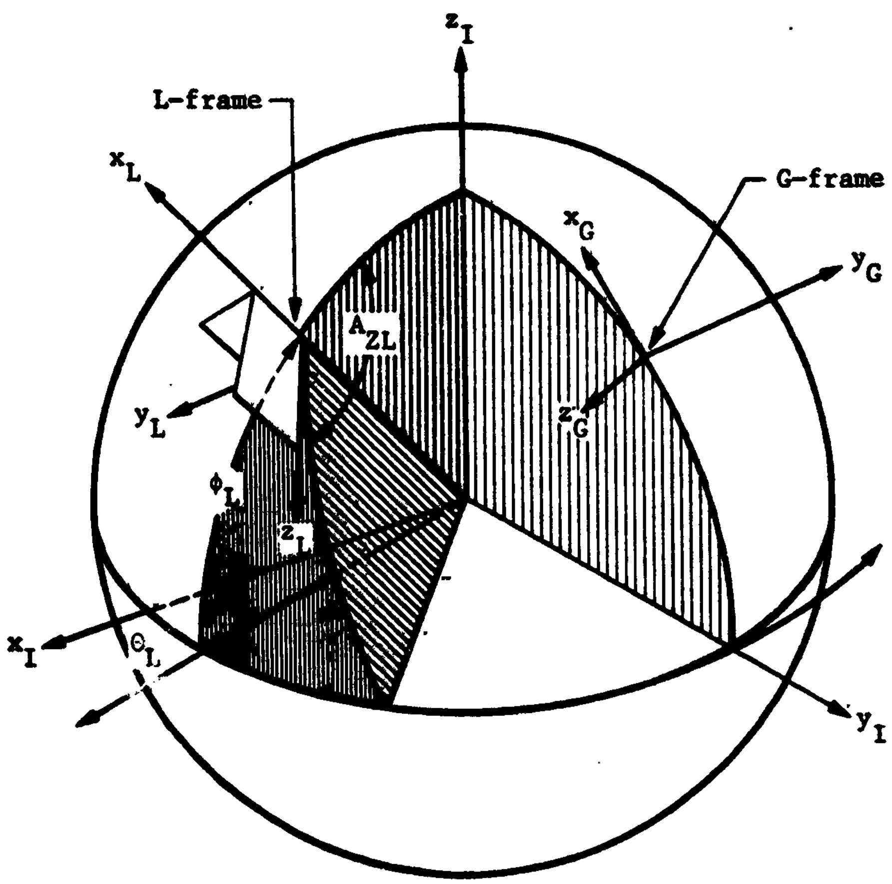


Figure III-2.- Launch Frame

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)

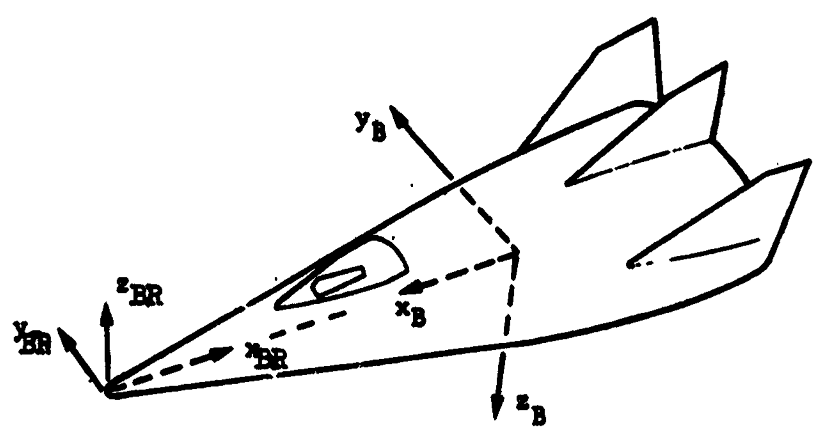


Figure III-3.- Body Frames

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 (h_a, h_p, i, Ω, θ, ω).- 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).

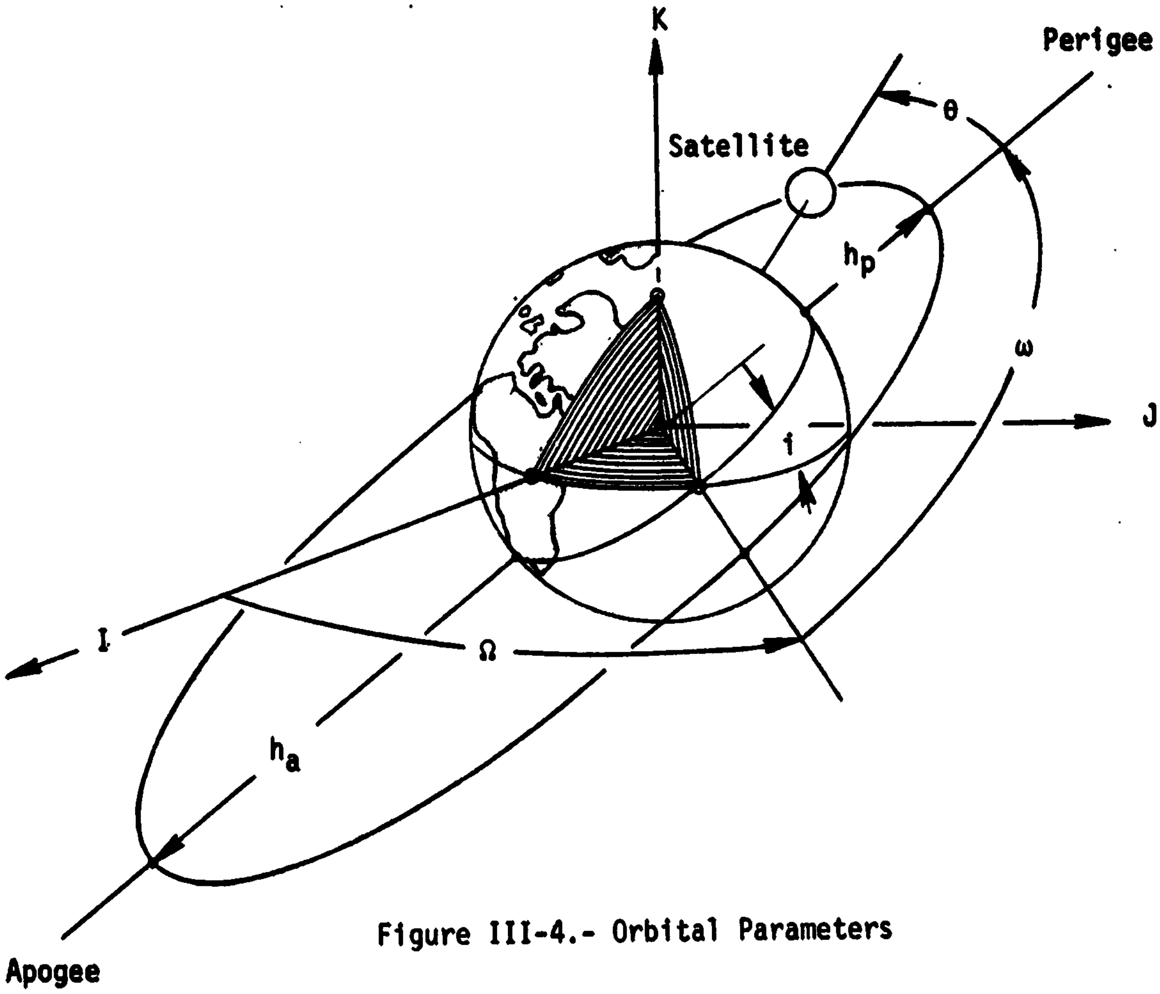


Figure III-4.- Orbital Parameters

Attitude Angles

The program contains the following standard attitude reference 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);

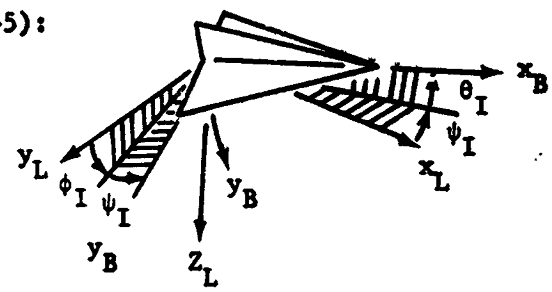


Figure III-5.- Inertial Euler Angles

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).

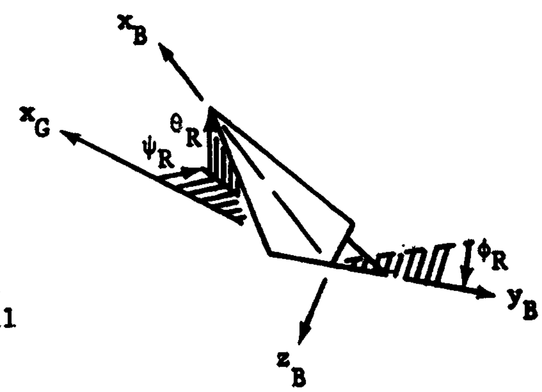


Figure III-6.- Relative Euler Angles

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);

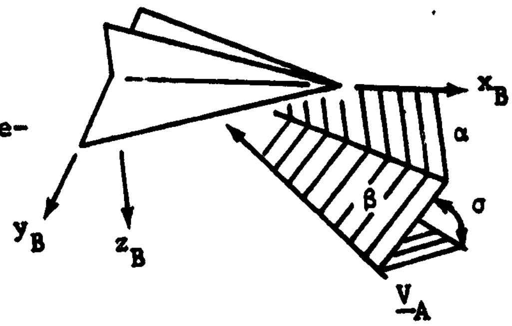


Figure III-7.- Aerodynamic Angles

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);

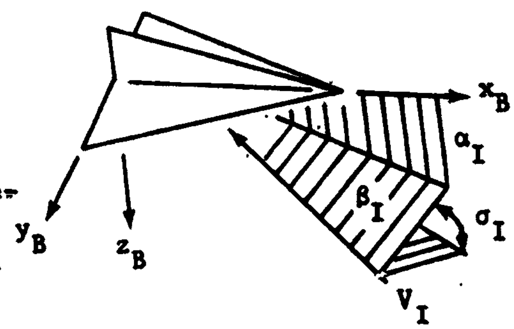


Figure III-8.- Inertial Aerodynamic Angles

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.

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

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.

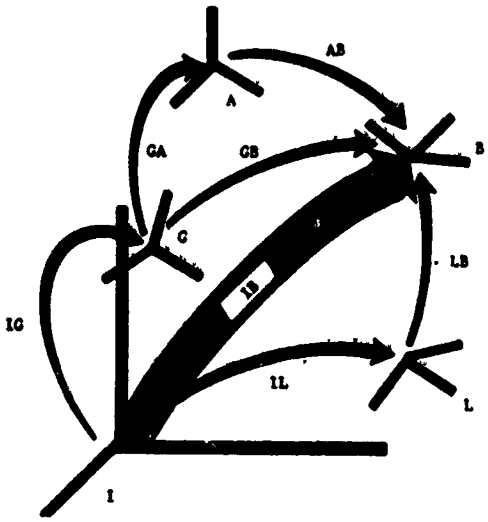


Figure III-9.- Matrix Transformations

A summary of these matrices is given below. The symbols c and s 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

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

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

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

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

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

$\underline{[AB], \text{ ARVS to body}}$.- 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 [IP] matrix, are presented below.

$\underline{[IP], \text{ inertial to planet relative}}$.- 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

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

$\underline{[RB], \text{ body reference to body}}$.- 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. 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

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

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.

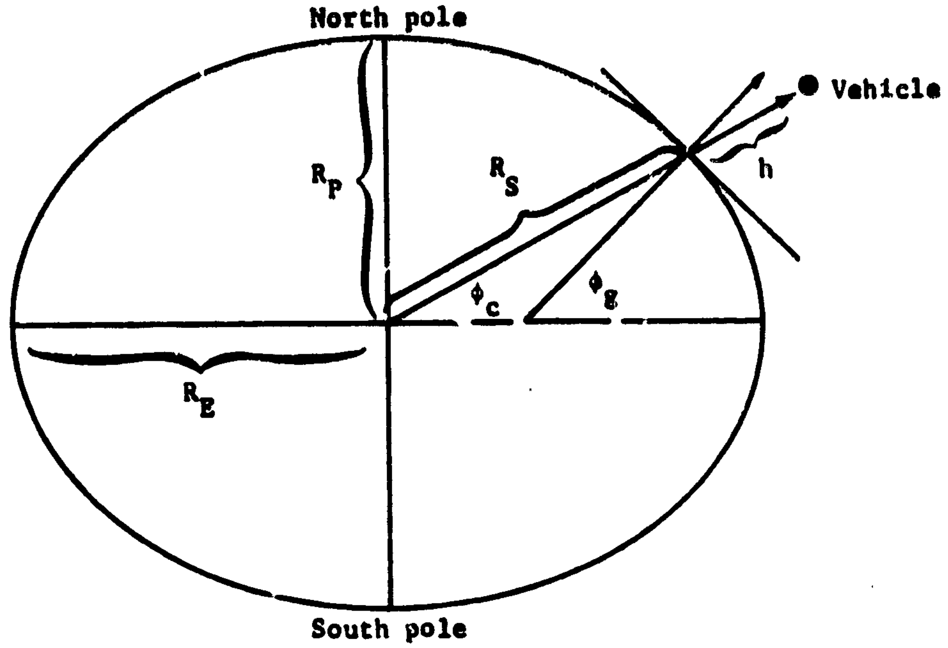


Figure IV-1.- Oblate Planet

Gravitational Model

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

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

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 polynominals. 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

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

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 $\left(H_g\right)$, which is computed as

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

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:

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

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:

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

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:

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

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.

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

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 polynominals.- 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:

$$
\left.
\begin{array}{l}
\text{1) Altitude region = 0 to 28 000 meters:} \\
P = P_1 \exp\left(A + A_1 h + A_2 h^2 + A_3 h^3 + A_4 h^4 + A_5 h^5\right) \\
\qquad \text{where } P_1 = 10.0 \text{ Newtons/cm}^2; \\
\text{2) Altitude region = 28 000 to 83 004 meters:} \\
P = g_0 \times 10^{-4} \exp\left(A + A_1 h + A_2 h^2 + A_3 h^3 + A_4 h^4 + A_5 h^5\right); \\
\text{3) Altitude region = 83 004 to 90 000 meters:} \\
P = P_B \exp\left(\dfrac{-1.373301523 \times 10^{12} \quad h - h_B}{T_B \; (6344860 + h) \; \left(6344860 + h_B\right)}\right); \\
\text{4) Altitude region = 90 000 to 700 000 meters:} \\
L_n \, (P) = L_n \left(P_B\right) + \dfrac{1.373301523 \times 10^{12}}{L_m \; (6344860 + h) \; \left(6344860 + h_B\right)} \\
\qquad\qquad L_n \left(\dfrac{T_{M_B}}{T_{M_B} + L_m \left(h - h_B\right)}\right).
\end{array}
\right\} \quad \text{(IV-11)}
$$

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

$$T = T_B = 180.65°\mathrm{K};$$

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

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

(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:

$$\rho = (34.83676) \frac{P}{T}.$$

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

TABLE IV-2.- DERIVED COEFFICIENTS FOR THE 1963 PATRICK AFB ATMOSPHERE MODEL - CONCLUDED

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 (V_W), the wind azimuth (A_{ZW}), and the wind azimuth bias (A_{ZWB}). 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

$$\underline{V}_{WI} = [IG]^{-1} \underline{V}_{WG} \tag{IV-17}$$

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

$$\underline{V}_{AI} = \underline{V}_{I} - \underline{\omega}_{p} \times \underline{r}_{I} - \underline{V}_{WI} \tag{IV-18}$$

and its magnitude is given by

$$V_A = \sqrt{\underline{V}_{AI} \cdot \underline{V}_{AI}} \tag{IV-19}$$

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.

The composite inertia matrix is input with respect to the body axis system which is located at the instantaneous composite center of gravity of the vehicle.* In 6D POST, the moments and products of inertia are defined as the integrals

$$I_{xx} = \int y^2 + z^2 \, dv \qquad I_{xy} = \int xy \, dv$$

$$I_{yy} = \int x^2 + z^2 \, dv \qquad I_{xz} = \int xz \, dv \qquad \text{(V-5)}$$

$$I_{zz} = \int x^2 + y^2 \, dv \qquad I_{yz} = \int yz \, dv. \qquad \text{(V-5)}$$

The inertia matrix is then given by

$$[I] = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}, \qquad \text{(V-6)}$$

and is generally input as a function of the weight of the vehicle.

The composite center of gravity is referenced with respect to the vehicle reference frame, and the components (x_{cg}, y_{cg}, z_{cg}) are generally input as a function of vehicle weight.

Propulsion Calculations

6D POST can simulate both rocket and jet engines. The program can simulate up to 15 engines in either mode.

Rocket engines.- 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

$$T_{R_i} = \eta \, T_{vac_i} - A_{E_i} \, p(h), \qquad \text{(V-7)}$$

*In 6D POST the center of gravity is assumed to coincide with the center of mass.

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

$$T_R = \sum_{i=1}^{N_{eng}} T_{R_i}, \tag{V-8}$$

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

The weight flowrate in the rocket mode is given by

$$\dot{W} = \begin{cases} -\eta \sum_{i=1}^{N_{eng}} \left(\dot{W}_p^{max}{}_i\right) \\ -\eta \sum_{i=1}^{N_{eng}} \left(T_{vac}/I_{sp_{vac}}\right)_i \end{cases} \tag{V-9}$$

<u>Jet engines</u>.- In the jet engine mode the net jet thrust per engine is given by

$$\left(\frac{T_J}{\delta}\right)_i = f(M, \eta), \tag{V-10}$$

where

$$\delta = p(h) / p_{SL}$$

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

$$T_J = \sum_{i=1}^{N_{eng}} p(h)\Big/\left(P_{SL}\right) \left(T_J/\delta_i\right). \tag{V-11}$$

The weight flowrate in the jet engine mode is

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

The thrust equation for each engine is given by

$$\underline{F}_{TB_i} = T_{B_i} \begin{bmatrix} \cos \delta e_{p_i} \cos \delta e_{y_i} \\ \sin \delta e_{y_i} \\ -\cos \delta e_{y_i} \sin \delta e_{p_i} \end{bmatrix}, \tag{V-13}$$

where the pitch and yaw engine deflection angles δe_p and δe_y are defined in Figure V-1.

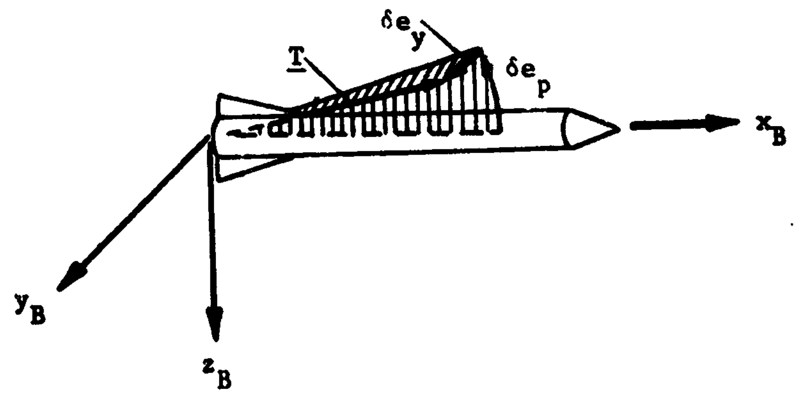


Figure V-1.- Engine Gimbal Angles

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.

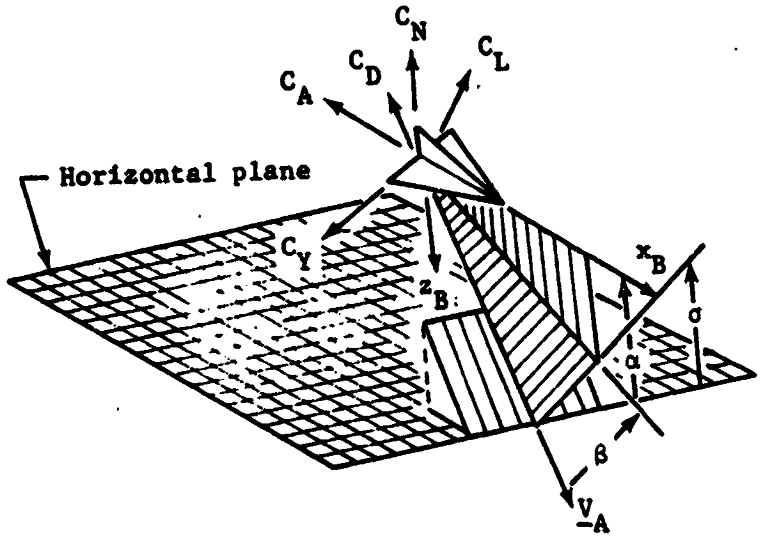


Figure V-2.- Aerodynamic Angles

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 obtained by summing the individual contributions as follows:

$$C_A = C_{A_0} + C_A(\alpha,M) + C_{A_{\delta a}}\,\delta a + C_{A_{\delta e}}\,\delta e + C_{A_{\delta r}}\,\delta_r + \sum_{i=1}^{3} C_{A_{\delta f_i}}\,\delta f_i, \quad \text{(V-15)}$$

$$C_N = C_{N_0} + C_N(\alpha,M) + C_{N_{\delta a}}\,\delta a + C_{N_{\delta e}}\,\delta e + C_{N_{\delta r}}\,\delta r + \sum_{i=1}^{3} C_{N_{\delta f_i}}\,\delta f_i, \quad \text{(V-16)}$$

or optionally

$$C_D = C_{D_0} + C_D(\alpha,M) + C_{D_{\delta a}}\,\delta a + C_{D_{\delta e}}\,\delta e + C_{D_{\delta r}}\,\delta r + \sum_{i=1}^{3} C_{D_{\delta f_i}}\,\delta f_i, \quad \text{(V-17)}$$

$$C_L = C_{L_0} + C_L(\alpha,M) + C_{L_{\delta a}}\,\delta a + C_{L_{\delta e}}\,\delta e + C_{L_{\delta r}}\,\delta r + \sum_{i=1}^{3} C_{L_{\delta f_i}}\,\delta f_i, \quad \text{(V-18)}$$

and

$$C_Y = C_{Y_0} + C_Y(\beta,M) + C_{Y_{\delta a}}\,\delta a + C_{Y_{\delta e}}\,\delta e + C_{Y_{\delta r}}\,\delta r + \sum_{i=1}^{3} C_{Y_{\delta f_i}}\,\delta f_i. \quad \text{(V-19)}$$

The aerodynamic moment coefficients are given by:

$$C_\ell = C_{\ell_o} + C_\ell(\beta,M) + C_{\ell_{\delta a}}\,\delta a + C_{\ell_{\delta e}}\,\delta e + C_{\ell_{\delta r}}\,\delta r$$

$$+ \sum_{i=1}^{3} C_{\ell_{\delta f_i}}\,\delta f_i + C_{\ell_{r'}}\,r' + C_{\ell_{p'}}\,p' \quad \text{(V-20)}$$

$$C_m = C_{m_o} + C_m(\alpha,M) + C_{m_{\delta a}}\,\delta a + C_{m_{\delta e}}\,\delta e + C_{m_{\delta r}}\,\delta r$$

$$+ \sum_{i=1}^{3} C_{m_{\delta f_i}}\,\delta f_i + C_{m_{q'}}\,q', \tag{V-21}$$

$$C_n = C_{n_o} + C_n(\beta,M) + C_{n_{\delta a}}\,\delta a + C_{n_{\delta e}}\,\delta e + C_{n_{\delta r}}\,\delta r$$

$$+ \sum_{i=1}^{3} C_{n_{\delta f_i}}\,\delta f_i + C_{n_{p'}}\,p' + C_{n_{r'}}\,r', \tag{V-22}$$

where δf_i, $i = 1,2,3$, are arbitrary user defined deflection angles; and

$$p' = pd_R/2V_A = \omega_x d_R/2V_A$$

$$q' = qd_P/2V_A = \omega_y d_P/2V_A$$

$$r' = rd_Y/2V_A = \omega_z d_Y/2V_A.$$

The Mach number and dynamic pressure are given by:

$$\left.\begin{aligned} M &= \frac{V_A}{C_S} \\ q &= \frac{1}{2}\,\rho\,V_A^2, \end{aligned}\right\} \tag{V-23}$$

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.,

$$\begin{aligned} \rho &= \rho(h) \\ C_S &= C_S(h) \\ p &= p(h) \\ T &= T(h). \end{aligned} \tag{V-24}$$

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

$$\alpha = \tan^{-1}\left[\frac{\sin\alpha}{\cos\alpha}\right]$$

$$\beta = \tan^{-1}\left[\frac{\sin\beta}{\cos\beta}\right],$$

$$\sin\alpha = \frac{w_B}{\sqrt{u_B^2 + w_B^2}}$$

$$\cos\alpha = \frac{u_B}{\sqrt{u_B^2 + w_B^2}} \qquad \text{(V-25)}$$

$$\sin\beta = \frac{v_B}{V_A}$$

$$\cos\beta = \frac{\sqrt{u_B^2 + w_B^2}}{V_A}$$

The total angle of attack is

$$\alpha_T = \cos\left(V_{AB_x}/V_A\right). \qquad \text{(V-26)}$$

The aerodynamic forces in the body frame are

$$\underline{F}_{AB} = q\ S\begin{bmatrix} -C_A \\ C_Y \\ -C_N \end{bmatrix}, \qquad \text{(V-27)}$$

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

Aeroheating Calculations

6D 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

$$\dot{Q} = \frac{17\ 600}{\sqrt{R_N}} \left(\frac{\rho}{\rho_{SL}}\right)^{1/2} \left(\frac{V_R}{V_C}\right)^{3.15}, \qquad \text{(V-28)}$$

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

$$\dot{Q} = Q_t\ (x_1,\ x_2,\ x_3), \qquad \text{(V-29)}$$

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

$$\dot{Q} = Q_t\ (x_1,\ x_2,\ x_3)\ Q_c, \qquad \text{(V-30)}$$

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

$$\dot{Q} = \dot{Q}_t\ (x_1,\ x_2,\ x_3) \left[1500 \left(\frac{\rho}{\rho_{SL}}\right)^{0.8} \left(\frac{V_A}{10^4}\right)^{3.18}\right]. \qquad \text{(V-31)}$$

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

a) Altitude-velocity correction:

$$\left.\begin{aligned} \Delta h = 10^5 \Big[& 1.06112 - 6.16586\ V_A/10^4 \\ & + 51.12090 \left(V_A/10^4\right)^4 - 20.66258 \left(V_A/10^4\right)^5 \Big] \\ & + 22.52598 \left(V_A/10^4\right)^2 - 48.28080 \left(V_A/10^4\right)^3 \\ h_{ref} = h + \Delta h. & \end{aligned}\right\} \text{(V-32)}$$

b) Maximum centerline heat rate at reference conditions:

$$\left.\begin{aligned}
&\text{-- if } h_{ref} \geq 103\ 600 \text{ m:}\\
&\dot{q}_{ref} = 10^2\Big[277.93332 + 134.55760\ h_{ref}/10^5 - 807.75941\ \left(h_{ref}/10^5\right)^2\\
&\qquad + 2.90536\ \left(h_{ref}/10^5\right)^3 + 722.36896\ \left(h_{ref}/10^5\right)^4 - 311.40176\ \left(h_{ref}/10^5\right)^5\Big];\\
&\text{-- if } h_{ref} \leq 103\ 600 \text{ m;}\\
&\dot{q}_{ref} = 10^4\Big[7115.39692 - 34\ 881.13588\ h_{ref}/10^5 + 69\ 844.23141\ \left(h_{ref}/10^5\right)^2\\
&\qquad - 71\ 534.98453\ \left(h_{ref}/10^5\right)^3 + 37\ 506.13054\ \left(h_{ref}/10^5\right)^4 - 8048.55112\ \left(h_{ref}/10^5\right)^5\Big].
\end{aligned}\right\} \quad \text{(V-33)}$$

c) Angle of attack correction:

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

where

$$x = 10^2\ [0.01136 + 0.01343\ \alpha/10^2 + 1.42672\ (\alpha/10^2)^4 - 0.75623\ (\alpha/10^2)^5] + 0.30535\ (\alpha/10^2)^2 - 1.06269\ (\alpha/10^2)^3. \quad \text{(V-34)}$$

d) Maximum centerline heat rate:

$$\dot{q}_{max} = \left(\dot{q}_{max,\alpha}\right)\Big/\left(\dot{q}_{max,\alpha=50°}\right)\ \left(\dot{q}_{ref}\right). \quad \text{(V-35)}$$

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

$$\dot{Q} = q\ V_A. \quad \text{(V-36)}$$

The aerodynamic heating indicator for zero total angle of attack is

$$Q = \int_0^t \dot{Q}\, dt. \tag{V-37}$$

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

$$Q' = \int_0^t f(\alpha', M)\, \dot{Q}\, dt, \tag{V-38}$$

where

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

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

and

$$\left.\begin{array}{l}
\left.\begin{array}{ll} \alpha' & = \alpha \\ Q'_T & = Q' \end{array}\right\} \text{for } \alpha < 0° \\[2ex]
\left.\begin{array}{ll} \alpha' & = \alpha \\ Q'_B & = Q' \end{array}\right\} \text{for } \alpha > 0° \\[2ex]
\left.\begin{array}{ll} \alpha' & = \beta \\ Q'_L & = Q' \end{array}\right\} \text{for } \beta < 0° \\[2ex]
\left.\begin{array}{ll} \alpha' & = \beta \\ Q'_R & = Q' \end{array}\right\} \text{for } \beta > 0°.
\end{array}\right\} \tag{V-39}$$

The heating indicator for laminar flow is calculated as

$$Q_{lam} = \int_0^t 17\ 600\ K_{\alpha_T} \left(\frac{\rho}{\rho_0}\right)^{1/2} \left(\frac{V_A}{26\ 000}\right)^{3.15} dt, \qquad \text{(V-40)}$$

where

$$K_{\alpha_T} = f\ (\alpha_T). \qquad \text{(V-41)}$$

The heating indicator for turbulent flow is calculated as

$$Q_{turb} = \int 1500\ K_{\alpha_T} \left(\frac{\rho}{\rho_0}\right)^{0.8} \left(\frac{V_A}{10\ 000}\right)^{3.18} dt \qquad \text{(V-42)}$$

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

$$Q_i = H_{R_i}\ Q, \qquad \text{(V-43)}$$

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:

$$W_P = \sum_{i=1}^{10} W_{uA_i}\ A_i \qquad \text{(V-44)}$$

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

Sensor Module

The sensor module computes information that describes the behavior of the sensing elements of the vehicle's navigation system. Thus, the primary functional responsibility of the module is that of simulating hardware characteristics of sensors. For example, the behavior of an inertial measurement unit (IMU) can be described by a mathematical model of the platform and the accelerometers. Frequently this module is used for error analysis purposes.

Sensor models called by this module are necessarily vehicle and subsystem dependent. As a consequence, the sensor model must be designed and implemented for each particular application.

There are many applications of the program that do not require a specific simulation of the sensors. Therefore, for convenience, a "perfect" sensor model is coded into this routine. This "perfect" sensor model sets the sensed program variables equal to their actual values as calculated in the simulation models.

Navigation Module

The function of the navigation model is to estimate the state (position, velocity, etc) of the vehicle based on the sensor outputs. Clearly, this module is also vehicle and subsystem dependent and must be designed and implemented for each specific application. This version of the program contains no navigation models. As a consequence, the estimated state is set equal to the actual state. This is equivalent to simulation of perfect navigation.

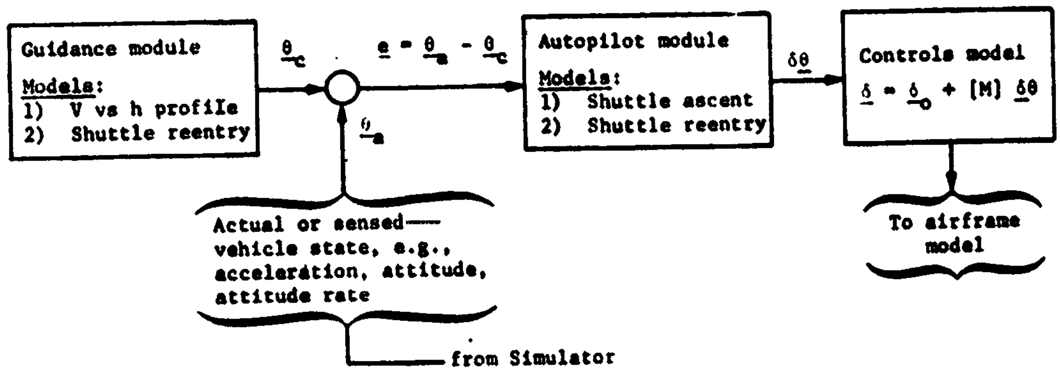
Guidance Module

The guidance module takes the output of the navigation model and computes a guidance command. Typically, the guidance command represents a desired change in the current attitude of the vehicle. This command is computed on the basis of meeting some specified trajectory condition, such as, inject conditions or landing conditions. The autopilot is designed to remove the errors between the commanded values of the guidance variables and their actual (or sensed) values. This is accomplished by deflecting engines, control surfaces, and/or firing RCS jets.

The current version of 6D POST contains three preloaded guidance options: (1) an open-loop profile steering; (2) a closed loop v-h profile ascent algorithm; and (3) the constant drag Space Shuttle reentry scheme (ref. H-1). If these methods are inadequate, the user may implement his own guidance algorithm into this module.

Autopilot Module

The function of the autopilot module is the generation of a command, which, when implemented through the deflection equations contained in the controls module, causes the vehicle to respond as prescribed by the guidance module. This functional responsibility is depicted in Figure V-3.



Nomenclature:

$\underline{\theta}_c$ - Guidance command

$\underline{\theta}_a$ - Actual or sensed values of guidance variables

e - Generalize error signal

$\delta\underline{\theta}$ - Pitch, yaw, and roll autopilot command

$\underline{\delta}$ - Deflection (engine or aerodynamic surfaces) angles

Figure V-3.- Functional Flow

The autopilot module calculates only autopilot commands based on the input guidance commands, and does not calculate engine or control surface deflections. The engine and control surface deflections are computed in the controls module as a linear function of the autopilot commands. The autopilot commands $\delta\theta$, $\delta\phi$, $\delta\psi$ represent changes in vehicle attitude. The mixing equations determine the engine and control surface deflections that create the control forces and moments.

Currently, there are two Space Shuttle autopilot models available in 6D POST. One autopilot is for ascent and the other for reentry. The ascent autopilot is somewhat standard and could be used on most ascent problems with little or no modification. The basic inputs to this model are: attitude commands from the guidance, inertial attitude angles, body rotational rates, translational accelerations, and preloaded engine deflection commands. The outputs are pitch, yaw, and roll autopilot commands, which are sent to the controls module to determine the engine deflection angles. The reentry autopilot is Space Shuttle oriented and is probably not applicable to other vehicle configurations. This model is intended to provide attitude control for Space Shuttle beginning at approximately 400,000-ft altitude and ending in the high subsonic flight regime. The control logic makes use of both aerodynamic control surface torques and reaction control jets. A complete description of this model is presented in ref. H-2.

Controls Model

The controls model converts pitch, yaw, and roll autopilot commands into aerodynamic control surface deflection angles and/or engine gimbal angles. The conversion of the autopilot commands into deflection angles is implemented through the matrix mixing logic given by the equation

$$\underline{\delta} = \underline{\delta}_0 + [M]\ \delta\underline{\theta}, \qquad \text{(V-45)}$$

where $\underline{\delta}$ denotes a general deflection angle with a null position of $\underline{\delta}_0$, $[M]$ the mixing gains, and $\delta\underline{\theta}$ the autopilot commands. The gains contained in the mixing matrix, $[M]$, and the null deflections, $\underline{\delta}_0$, are specified by user input.

The standard aerodynamic surface deflection mixing equations used in the program are

$$\delta\dot{a} = \delta\dot{a}_o + KR_{\delta a}\,\delta\dot{\phi} + KP_{\delta a}\,\delta\dot{\theta} + KY_{\delta a}\,\delta\dot{\psi}, \qquad \text{(V-46)}$$

$$\delta\dot{e} = \delta\dot{e}_o + KR_{\delta o}\,\delta\dot{\phi} + KP_{\delta e}\,\delta\dot{\theta} + KY_{\delta o}\,\delta\dot{\psi} \qquad \text{(V-47)}$$

$$\delta\dot{R} = \delta\dot{R}_o + KR_{\delta r}\,\delta\dot{\phi} + KP_{\delta r}\,\delta\dot{\theta} + KY_{\delta r}\,\delta\dot{\psi}, \qquad \text{(V-48)}$$

and the standard engine deflection angle mixing equations are similarly

$$\delta e_{p_i} = \delta e_{p_{io}} + KP_{\delta p}\,\delta\theta + KR_{\delta p}\,\delta\phi, \qquad \text{(V-49)}$$

$$\delta e_{y_i} = \delta e_{y_{io}} + KY_{\delta y}\,\delta\psi + KR_{\delta y}\,\delta\phi, \qquad \text{(V-50)}$$

where $\delta\dot{\theta}$, $\delta\dot{\psi}$, and $\delta\dot{\phi}$ are the pitch, yaw, and roll autopilot commands.

Airframe Model

The airframe model computes the total thrust and moments acting on the vehicle. The forces and moments are computed from the engine and aerodynamic deflection angles and the RCS thrust and moments.

The nongravitational force acting on the vehicle is computed in the body frame as

$$\underline{F}_B = \underline{F}_{TB} + \underline{F}_{AB} + \underline{F}_{RCS}, \qquad \text{(V-51)}$$

where $\underline{F}_{TB}$ is the total force due to the engines, $\underline{F}_{AB}$ is the total force due to aerodynamic effects, and $\underline{F}_{RCS}$ is the total force resulting from the reaction-control system. Similarly, the total moment acting on the vehicle is computed as

$$\underline{M}_B = \underline{M}_{TB} + \underline{M}_{AB} + \underline{M}_{RCS}. \qquad \text{(V-52)}$$

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 engine gimbal angles δe_{p_i} and δe_{y_i}. The total thrust force in the body frame is given by:

$$\underline{F}_{TB} = \sum_{i=1}^{N_{eng}} \underline{F}_{TB_i}, \tag{V-53}$$

where the individual engine components, $\underline{F}_{TB_i}$, are given by

$$\underline{F}_{TB_i} = T_{R_i} \begin{bmatrix} \cos \delta e_{p_i} \cos \delta e_{y_i} \\ \sin \delta e_{y_i} \\ -\cos \delta c_{y_i} \sin \delta e_{p_i} \end{bmatrix}. \tag{V-54}$$

For roll nozzles, the thrust vector is given by

$$\underline{F}_{TB_i} = [K]\ T_{R_i} \begin{bmatrix} \cos \delta e_{p_i} \cos \delta e_{y_i} \\ \sin \delta e_{y_i} \\ -\cos \delta e_{y_i} \sin \delta e_{p_i} \end{bmatrix} \tag{V-55}$$

where for a roll nozzle the deflection matrix $[K]$ is given by

$$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi_i & s\phi_i \\ 0 & -s\phi_i & c\phi_i \end{bmatrix} \tag{V-56}$$

where

$$\phi_i = \text{input value}$$

or

$$\phi_i = \tan^{-1} \left(z_{gp_i} / y_{gp_i} \right) \text{ if not input.}$$

The thrust moments are obtained by summing the thrust moments for each engine as follows:

$$\underline{M}_{TB} = \sum_{i=1}^{N_{eng}} \underline{F}_{TB_i} \times \Delta\underline{R}_{BT_i} \tag{V-57}$$

where

$$\Delta\underline{R}_{BT_i} = \begin{bmatrix} -\left(x_{gp_i} - x_{cg}\right) \\ \left(y_{gp_i} - y_{cg}\right) \\ -\left(z_{gp_i} - z_{cg}\right) \end{bmatrix}. \tag{V-58}$$

The aerodynamic forces and moments are given by:

$$\underline{F}_{AB} = \begin{bmatrix} F_{AXB} \\ F_{AYB} \\ F_{AZB} \end{bmatrix} = qS \begin{bmatrix} -C_A \\ C_Y \\ -C_N \end{bmatrix}, \tag{V-59}$$

and

$$\underline{M}_{AB} = qS \begin{bmatrix} d_R \; C_\ell \\ d_P \; C_m \\ d_Y \; C_n \end{bmatrix} - \underline{F}_{AB} \times \Delta\underline{R}_{AB} \tag{V-60}$$

where

$$\Delta\underline{R}_{AB} = \begin{bmatrix} -\left(x_{ref} - x_{cg}\right) \\ \left(y_{ref} - y_{cg}\right) \\ -\left(z_{ref} - z_{cg}\right) \end{bmatrix}. \tag{V-61}$$

The aerodynamic reference point $\left(x_{ref}, y_{ref}, z_{ref}\right)$ is calculated from tabular input.

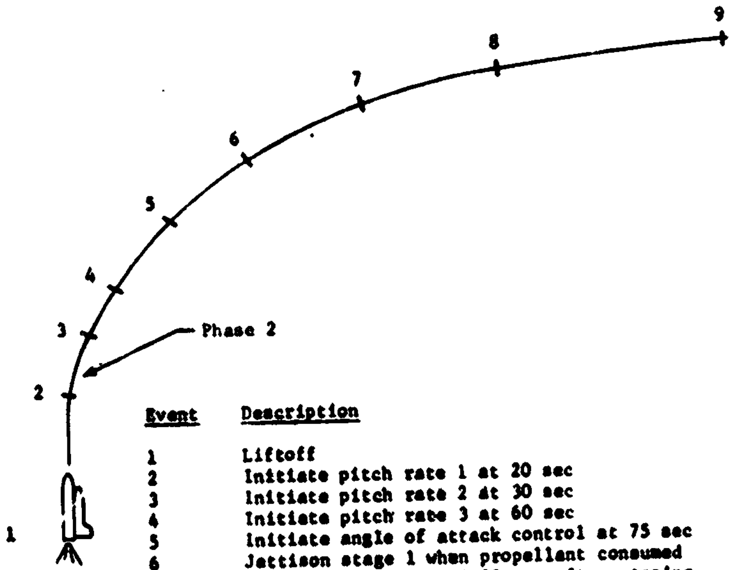
The RCS forces and moments are computed in the autopilot model and are merely added to obtain the resultant force and moment vectors.

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.

Three 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.

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 criterion being monitored at each integration step. If none of the criterion 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

$$\Delta t^* = - y^2(t)/(y(t + \Delta t) - y(t)) \quad \text{(VI-1)}$$

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.

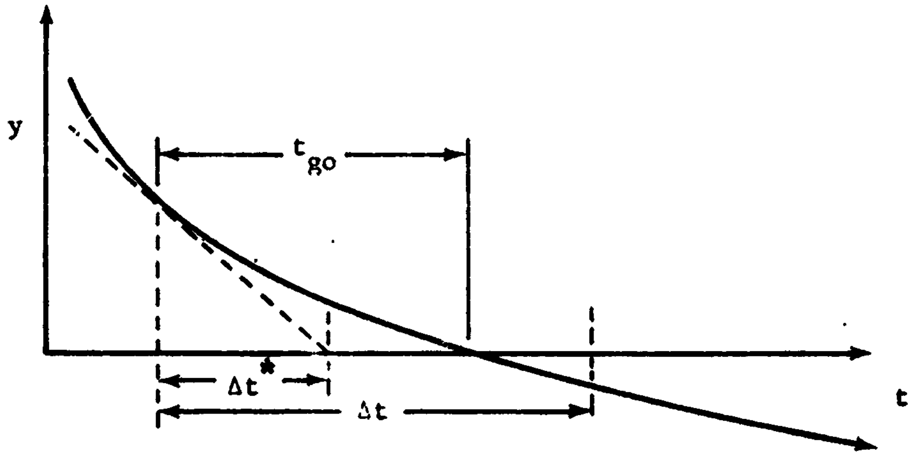


Figure VI-2.- Illustration of Time-to-Go Logic

Translational Equations

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

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

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

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 $(\theta_I$ or θ, ϕ_c or ϕ_g, h or r$)$.- In this option the equations vary and the sequence of calculation varies according to the choice of input. However, the basic equations used are:

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

and

$$\underline{r}_I = r_I \begin{bmatrix} \cos\phi_c \cos\theta_I \\ \cos\phi_c \sin\theta_I \\ \sin\phi_c \end{bmatrix} \tag{VI-5}$$

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

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

and then transformed to the ECI frame by

$$\underline{V}_I = [IG]^{-1} \underline{V}_{IG}. \tag{VI-7}$$

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

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

and then transformed to the ECI frame by

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

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

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

and then transformed to the ECI frame by

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

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

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

and

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

Rotational Equations

The rotational equations of motion are solved in the body-centered coordinate system. These equations are

$$\dot{\underline{e}} = \frac{1}{2} \, [E] \, \underline{\omega}_B \tag{VI-14}$$

$$\dot{\underline{\omega}}_B = [I]^{-1} [\underline{M}_B - [\dot{I}] \, \underline{\omega}_B - \underline{\omega}_B x [I] \, \underline{\omega}_B], \tag{VI-15}$$

$$\underline{M}_B = \underline{M}_{TB} + \underline{M}_{AB} + \underline{M}_{RCS}$$

where $\underline{e}$ is a four dimensional vector of quaternion parameters, $[E]$ is the quaternion matrix, $\underline{\omega}_B$ is the inertial angular velocity expressed in the body frame; $\underline{M}_B$ is the total external moment acting in the vehicle as a result of the thrust, the RCS, and the aerodynamic forces, and $[I]$ is the inertia matrix for the composite vehicle. The $[I]$ and $[E]$ matrices are given by

$$[I] = \begin{bmatrix} I_{XX} & -I_{XY} & -I_{XZ} \\ -I_{XY} & I_{YY} & -I_{YZ} \\ -I_{XZ} & -I_{YZ} & I_{ZZ} \end{bmatrix} \tag{VI-16}$$

$$[E] = \begin{bmatrix} -e_1 & e_2 & e_3 \\ e_0 & e_2 & -e_3 \\ e_0 & -e_1 & e_3 \\ e_0 & e_1 & -e_3 \end{bmatrix} \tag{VI-17}$$

The body rates are defined below and illustrated in Figure VI-3.

ω_x - Roll body rate. The angular rate about the x_B-axis in deg/sec,

ω_y - Pitch body rate. The angular rate about the y_B-axis in deg/sec,

ω_z - Yaw body rate. The angular rate about the z_B-axis in deg/sec.

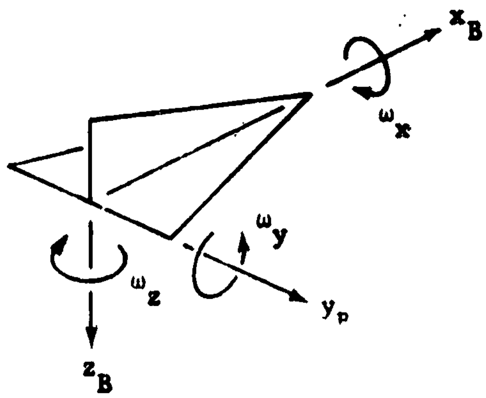


Figure VI-3.- Body Rates

Initialization.- The rotational equations of motion are initialized by defining both the attitude angles and the attitude rates. There are three options for initializing the attitude: (1) inertial Euler angles; (2) relative Euler angles; and (3) aerodynamic angles. There are three options for initializing the attitude rates: (1) body rates; (2) inertial Euler angle rates and (3) relative Euler angles rates. The attitude angles are used to compute initial values of the quaternions in order to initialize equation VI-14. The rates are used to initialize the moment equations. The equations for these options are presented below.

When inertial Euler angles are input the initial quaternion vector is given

$$\underline{e}_0 = \underline{e}\left(\phi_I\right) * \underline{e}\left(\psi_I\right) * \underline{e}\left(\theta_I\right) \tag{VI-18}$$

where the asterisk denotes quaternion multiplication and where

$$\left.\begin{aligned} \underline{e}\left(\phi_I\right) &= \cos\left(0.5\ \phi_I\right) + \sin\left(0.5\ \phi_I\right) i \\ \underline{e}\left(\psi_I\right) &= \cos\left(0.5\ \psi_I\right) + \sin\left(0.5\ \psi_I\right) k \\ \underline{e}\left(\theta_I\right) &= \cos\left(0.5\ \theta_I\right) + \sin\left(0.5\ \theta_I\right) j. \end{aligned}\right\} \tag{VI-19}$$

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

$$\begin{aligned} \underline{e}_0 = {} & \underline{e}\left(A_{ZL}\right) * \underline{e}(90) * \underline{e}\left(\phi_L\right) * \underline{e}\left(-\theta_L\right) * \underline{e}\left(\theta_I\right) * \underline{e}\left(-\phi_c\right) * \underline{e}(-90) * \\ & \underline{e}\left(\lambda_A\right) * \underline{e}\left(\gamma_A\right) * \underline{e}(\sigma) * \underline{e}(-\beta) * \underline{e}(\alpha), \end{aligned} \tag{VI-20}$$

where

$$\left.\begin{aligned}
\underline{e}\left(A_{ZL}\right) &= \cos\left(0.5\ A_{ZL}\right) - \sin\left(0.5\ A_{ZL}\right)\ k \\
\underline{e}(90) &= \cos\ (45) + \sin\ (45)\ j \\
\underline{e}\left(\phi_L\right) &= \cos\left(0.5\ \phi_L\right) + \sin\left(0.5\ \phi_L\right)\ j \\
\underline{e}\left(-\theta_L\right) &= \cos\left(0.5\ \theta_L\right) - \sin\left(0.5\ \theta_L\right)\ k \\
\underline{e}\left(\theta_I\right) &= \cos\left(0.5\ \theta_I\right) + \sin\left(0.5\ \theta_I\right)\ k \\
\underline{e}\left(\phi_c\right) &= \cos\left(0.5\ \phi_c\right) - \sin\left(0.5\ \phi_c\right)\ j \\
\underline{e}(\sigma) &= \cos\ 0.5\sigma + (\sin\ 0.5\sigma)\ i \\
\underline{e}(-\beta) &= \cos\ 0.5\beta - (\sin\ 0.5\beta)\ k \\
\underline{e}(\alpha) &= \cos\ 0.5\alpha + (\sin\ 0.5\alpha)\ j.
\end{aligned}\right\}\qquad \text{(VI-21)}$$

When relative Euler angles are input, then the initial quaternion vector is given

$$\begin{aligned}
\underline{e}_0 = \underline{e}\left(A_{ZL}\right) * \underline{e}(90) * \underline{e}\left(\phi_L\right) * \underline{e}\left(-\theta_L\right) * \underline{e}\left(\theta_I\right) * \underline{e}\left(-\phi_c\right) * \underline{e}(-90) * \\
* \ \underline{e}(\psi) * \underline{e}(\theta) * \underline{e}(\phi),
\end{aligned}\qquad \text{(VI-22)}$$

where

$$\left.\begin{aligned}
\underline{e}\left(A_{ZL}\right) &= \cos\left(0.5\ A_{ZL}\right) - \sin\left(0.5\ A_{ZL}\right)\ k \\
\underline{e}(90) &= \cos\ (45) + \sin\ (45)\ j \\
\underline{e}\left(\phi_L\right) &= \cos\left(0.5\ \phi_L\right) + \sin\left(0.5\ \phi_L\right)\ j \\
\underline{e}\left(-\theta_L\right) &= \cos\left(0.5\ \theta_L\right) - \sin\left(0.5\ \theta_L\right)\ k \\
\underline{e}\left(\theta_I\right) &= \cos\left(0.5\ \theta_I\right) + \sin\left(0.5\ \theta_I\right)\ k \\
\underline{e}\left(\phi_c\right) &= \cos\left(0.5\ \phi_c\right) - \sin\left(0.5\ \phi_c\right)\ j \\
\underline{e}(\psi) &= \cos\ 0.5\psi + (\sin\ 0.5\psi)\ k \\
\underline{e}(\theta) &= \cos\ 0.5\theta + (\sin\ 0.5\theta)\ j \\
\underline{e}(\phi) &= \cos\ 0.5\phi + (\sin\ 0.5\phi)\ i
\end{aligned}\right\}\qquad \text{(VI-23)}$$

The available options for initializing the moment equation are:

1) Input ω_x, ω_y, ω_z directly

2) Input the inertial Euler angle rates $\dot{\phi}_I$, $\dot{\psi}_I$, $\dot{\theta}_I$ and calculate ω_x, ω_y, ω_z via

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi}_I \cos\psi_I \cos\theta_I - \dot{\psi}_I \sin\theta_I \\ \dot{\theta}_I - \dot{\phi}_I \sin\psi_I \\ \dot{\phi}_I \cos\psi_I \sin\theta_I + \dot{\psi}_I \cos\theta_I \end{bmatrix} \qquad \text{(VI-24)}$$

3) Input the relative Euler angle rates $\dot{\psi}_R$, $\dot{\theta}_R$, $\dot{\phi}_R$ and calculate ω_x, ω_y, ωz via

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \underline{a} + \begin{bmatrix} \dot{\phi}_R - \sin\theta_R \dot{\psi}_R \\ \cos\phi_R \dot{\theta}_R + \sin\phi_R \cos\theta_R \psi_R \\ \cos\phi_R \cos\theta_R \psi_R - \sin\phi_R \theta_R \end{bmatrix}, \qquad \text{(VI-25)}$$

where

$$\underline{a} = [GB] \begin{bmatrix} \frac{v}{r_I} \\ \frac{-u}{r_I} \\ \frac{-v}{r_I} \tan\phi_c \end{bmatrix}$$

and

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

Integration Variables

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 40 integrals can be computed per phase.

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, $|\underline{r}_I \times \underline{V}_I|$

p semilatus rectum, h^2/μ

e eccentricity, $\sqrt{1 - p/a}$

ΔV velocity required to circularize orbit, $\sqrt{\Delta \underline{V} \cdot \Delta \underline{V}}$, where

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

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

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

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

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

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

$$\underline{u}_\Omega = \hat{\underline{z}}_I \times \underline{h} / |\hat{\underline{z}}_I \times \underline{h}|$$

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, $\tan^{-1}\left(\frac{u_{r_\infty}(2)}{u_{r_\infty}(3)}\right)$

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.

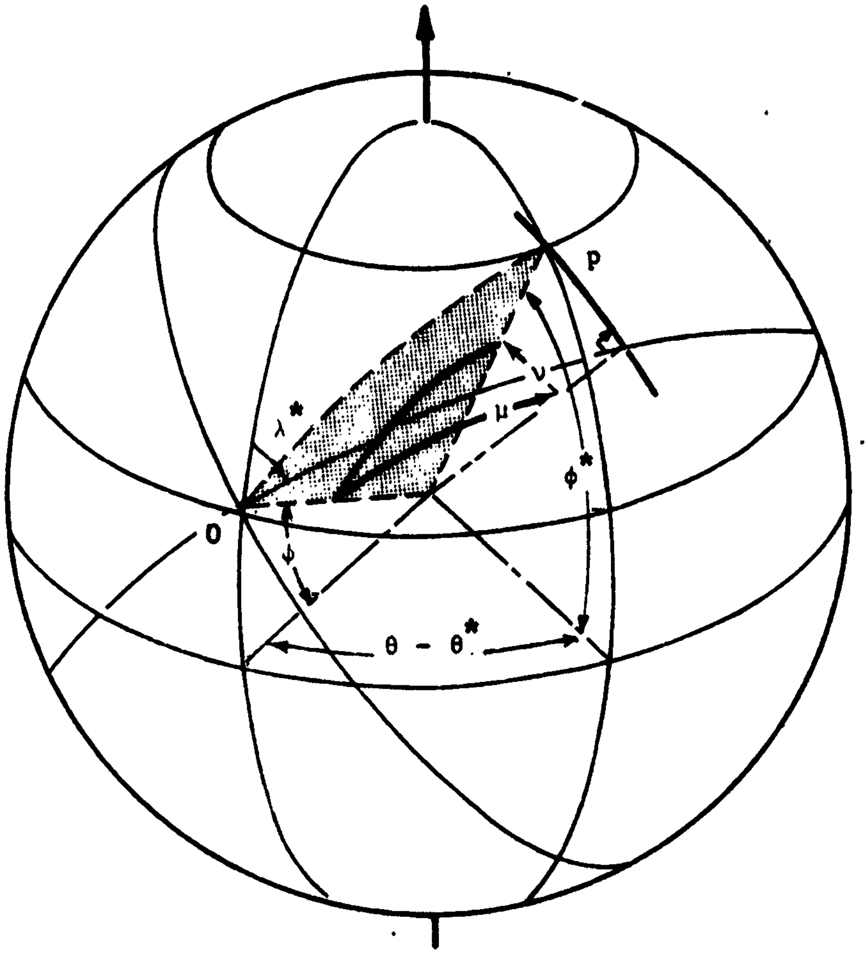
<u>Dot product downrange</u>.- The relative range angle, measured from the vehicle's initial position to its current position, is given by

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

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

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

<u>Crossrange and downrange via orbital plane reference</u>.- 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 0, and are inclined to the meridian by λ^*. From analytical geometry, ν and μ can be expressed as

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

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

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

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

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

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

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

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

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.

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

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

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

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:

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

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:

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

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.

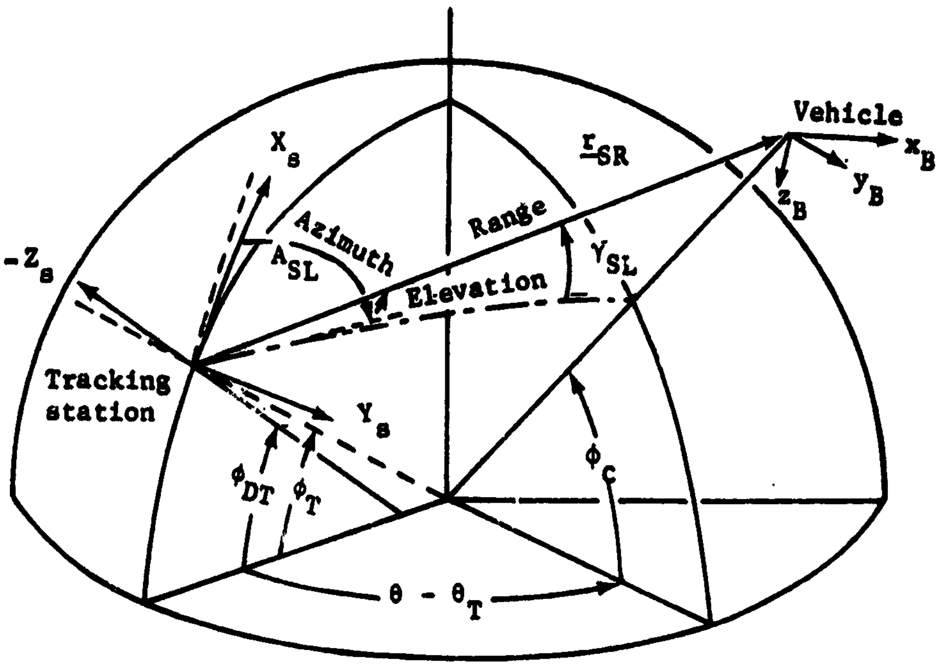


Figure VII-2.- Radar Tracking Schematic

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

$$\underline{r}_{TR} = \left(R_s + h_T\right) \begin{bmatrix} \cos \phi_T \cos \theta_T \\ \cos \phi_T \sin \theta_T \\ \sin \phi_T \end{bmatrix} , \tag{VII-11}$$

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

$$\underline{r}_{SR} = \underline{r}_I - [IP]^{-1} \underline{r}_{TR}, \tag{VII-12}$$

and the slant range is then computed as

$$r_{SR} = \sqrt{\underline{r}_{SR} \cdot \underline{r}_{SR}}. \tag{VII-13}$$

The elevation angle can then be computed as

$$\gamma_T = \sin^{-1} \left(\underline{u}_{TR} \cdot \underline{r}_{SR} / r_{SR}\right), \tag{VII-14}$$

where

$$\underline{u}_{TR} = [IP]^{-1} \underline{r}_{TR} / \left|[IP]^{-1} \underline{r}_{TR}\right|. \tag{VII-15}$$

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

$$\underline{r}_{SRG} = [IG] \underline{r}_{SR}, \tag{VII-16}$$

and thus the tracker's azimuth is given by

$$A_{ZT} = \tan^{-1} \left(y_{SRG} / x_{SRG}\right). \tag{VII-17}$$

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

$$\underline{r}_{SRB} = [IB] \underline{r}_{SR}. \tag{VII-18}$$

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

$$\psi_T = \cos^{-1}\left(x_{SRB}/r_{SR}\right), \tag{VII-19}$$

and the clock angle is given by

$$\alpha_c = \tan^{-1}\left(y_{SRB}/z_{SRB}\right). \tag{VII-20}$$

Space losses are calculated for the tracking stations as follows:

$$\left.\begin{aligned} SL_1 &= 36.56 + 20 \, Log_{10}\left(R_{SLM} \cdot FR_1\right) \\ SL_2 &= 36.56 + 20 \, Log_{10}\left(R_{SLM} \cdot FR_2\right) \\ SL_3 &= 36.56 + 20 \, Log_{10}\left(R_{SLM} \cdot FR_3\right), \end{aligned}\right\} \quad \text{(VII-21)}$$

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

$$r_I(E) = R_s\left(\phi_c\right) + h_{ip} \qquad \text{(VII-22)}$$

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

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

$$f(E) = \left(\cos\left(E - E_0\right) - e\cos E_0\right) \Big/ \left(1 - e\cos E_0\right) \qquad \text{(VII-23)}$$

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

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

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

$$\phi_{g_{ip}} = \tan^{-1}\left(kz_{ip} \Big/ \sqrt{x_{ip}^2 + y_{ip}^2}\right) \qquad \text{(VII-24)}$$

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

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 methods 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)

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

C 2

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 versatile 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

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

If $\underline{x}_n$ denotes the above state variables of the system being simulated at the n^{th} 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 n^{th} 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

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

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 i^{th} constraint is similarly represented.

The Jacobian matrix of the constraint vector function with respect to the independent variable is a matrix whose i^{th} 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 $\hat{\underline{u}}$ 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

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

The *feasible region* for the i^{th} inequality constraint is the half-space in the independent-variable space defined by the set

$$R_i = \left\{ \underline{u}; c_i(\underline{u}) \geq 0 \right\}, \tag{VIII-18}$$

while the complete feasible region for all of the constraints is

$$R = \bigcap_{i=1}^{n} R_i. \tag{VIII-19}$$

The boundary of the complete feasible region must be

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

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

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

$$B(\hat{\underline{u}}) = \bigcap_{i \in K(\hat{\underline{u}})} B_i \tag{VIII-21}$$

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

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

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

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

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

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

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

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

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

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

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

where

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

In particular,

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

and

$$P = I - \tilde{P}. \tag{VIII-29}$$

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

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

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 $\tilde{\underline{c}}$

h_c 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.

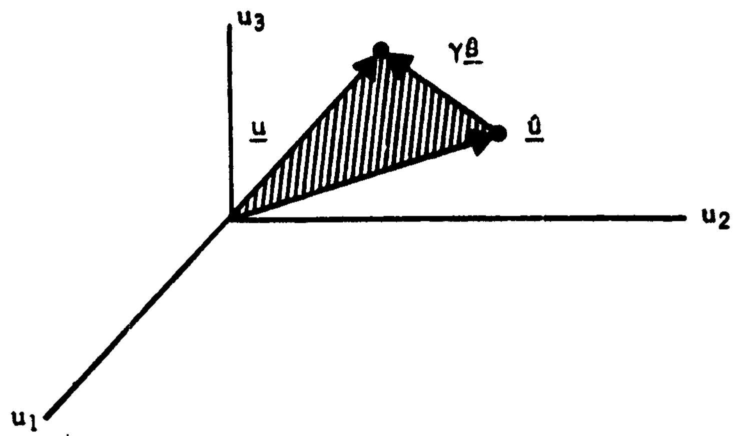


Figure VIII-1.- Direction of Search in the Independent-Variable Space

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.

<u>Constraint direction</u>.- 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,

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

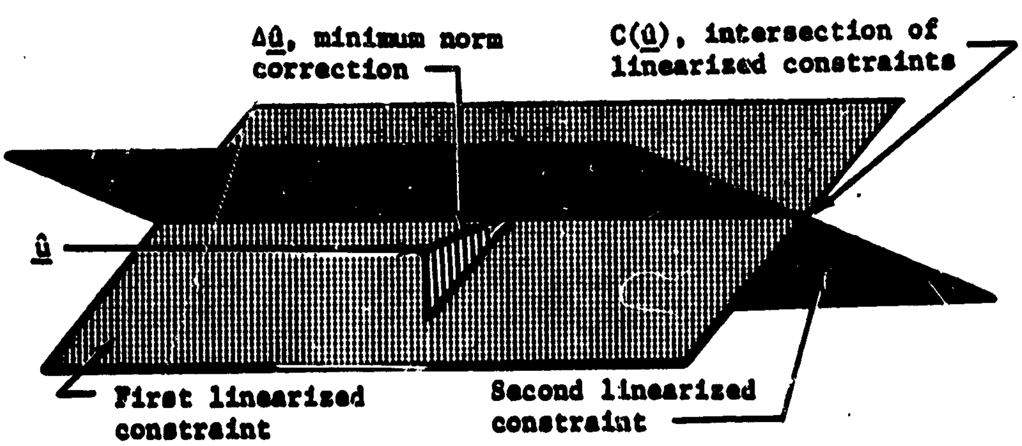


Figure VIII-2.- Illustration of Minimum-Norm Constraint, Direction for $\hat{n}_a = 2 < m = 3$

2) If $\hat{n}_a = m$, then the linearized local boundary $C(\hat{\underline{u}})$ 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

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

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

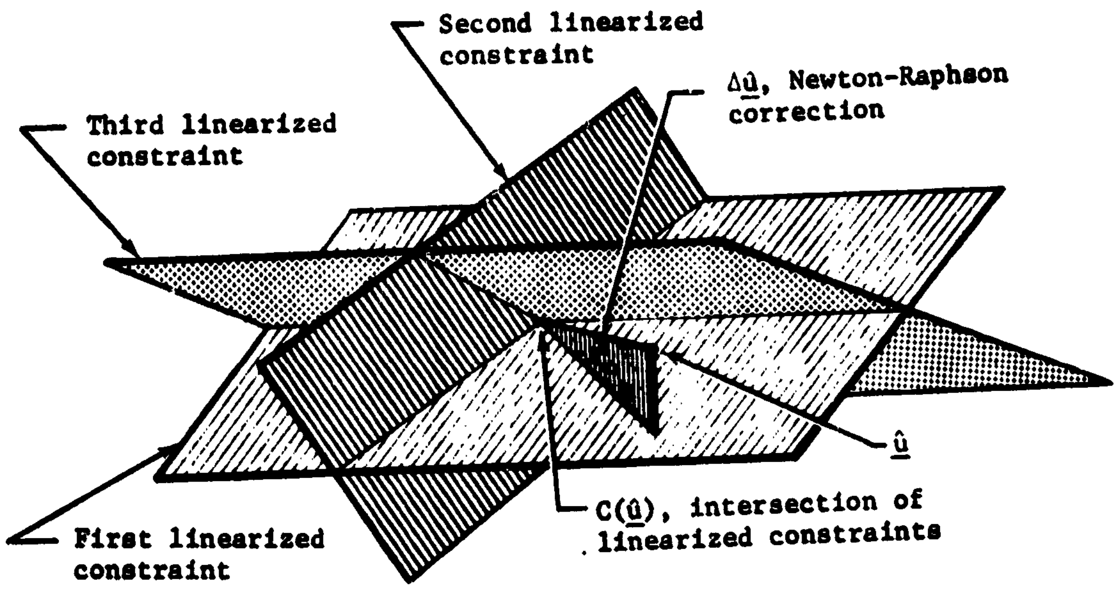


Figure VIII-3.- Illustration of Newton-Raphson Constraint, Direction for $\hat{n}_a = m = 3$

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

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

3) If $\hat{n}_a > m$, then $C(\hat{\underline{u}})$ 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 $\Delta\hat{\underline{u}}$ that minimizes the sum of the squares of the deviations from the linearized constraints. Thus, the function

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

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

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

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,

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

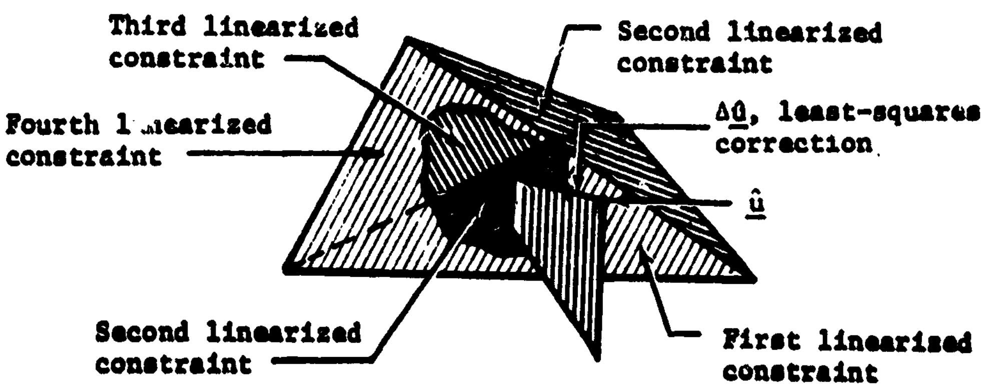


Figure VIII-4.- Illustration of Least-Squares Constraint, Direct for $n_a = 4 > m = 3$

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,

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

The direction of search for the accelerated projected gradient method is

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

where

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

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.

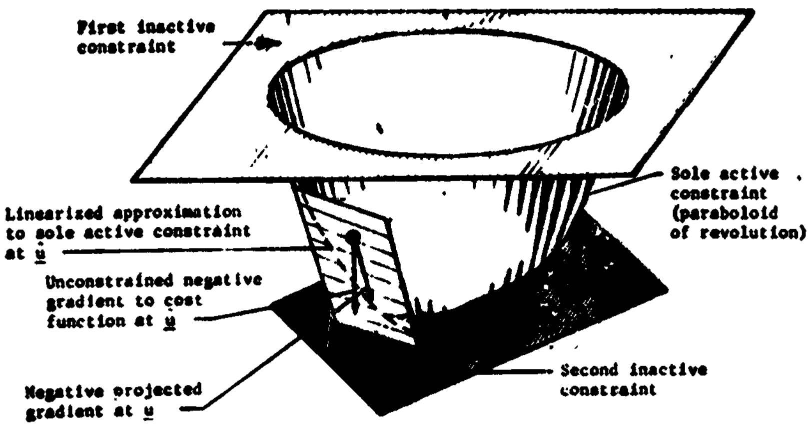


Figure VIII-5.- Direction of Negative-Projected Gradient for $n_a = 1$ and $m = 3$ (Feasible region is that region inside paraboloid, above lower plane, and below upper plane; cost-function is vertical height)

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

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

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

$$P_2 = F + W \, \underline{e}'\underline{e} \tag{VIII-50}$$

are:

1) Steepest-descent method

$$\underline{s}^o(\underline{\hat{u}}) = -\underline{\nabla P}_2(\underline{\hat{u}});$$

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)

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

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 $\hat{\underline{u}}$ 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 $\hat{\underline{u}}$ and the direction ofsearch $\hat{\underline{s}}$ 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, $\hat{\underline{s}}^c$, is the sum of the squares of the constraint violations; namely

$$h_c(\gamma) = \left|\underline{e}\left(\hat{\underline{u}} + \gamma\hat{\underline{s}}^c\right)\right|^2. \qquad \text{(VIII-51)}$$

Clearly

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

Differentiation via the chain rule yields

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

Recall that the search direction $\hat{s}^c$ was obtained as an independent-variable correction either satisfying all the linearized constraint equations if $\hat{n}_a \le m$, 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

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

Clearly

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

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

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

These properties of h_o are illustrated in figure 25.

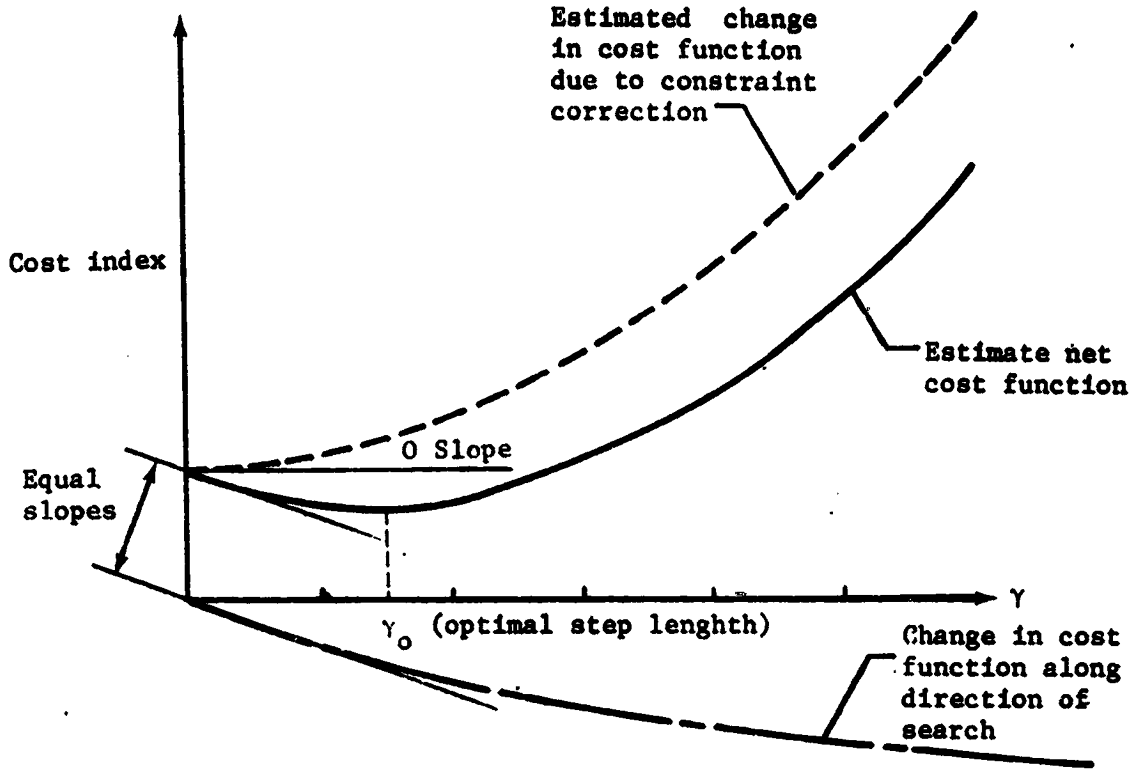


Figure VIII-6.- Properties of Estimated Net Cost Function

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 wit' 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

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

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

$$\lambda(k) = \begin{cases} -c_k(\underline{\hat{u}})/r_k & \text{if } r_k < 0 \\ \\ R & \text{if } r_k \geq 0 \end{cases} \tag{VIII-58}$$

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 k^{th} constraint. Hence a resonable upper bound for the step length is

$$\lambda = \min_{k \epsilon K} [\lambda(k)]. \tag{VIII-59}$$

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,

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

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

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

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

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

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

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) $f(\gamma_3)$.

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:

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

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,

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

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

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

where γ_o is the optimum step size.

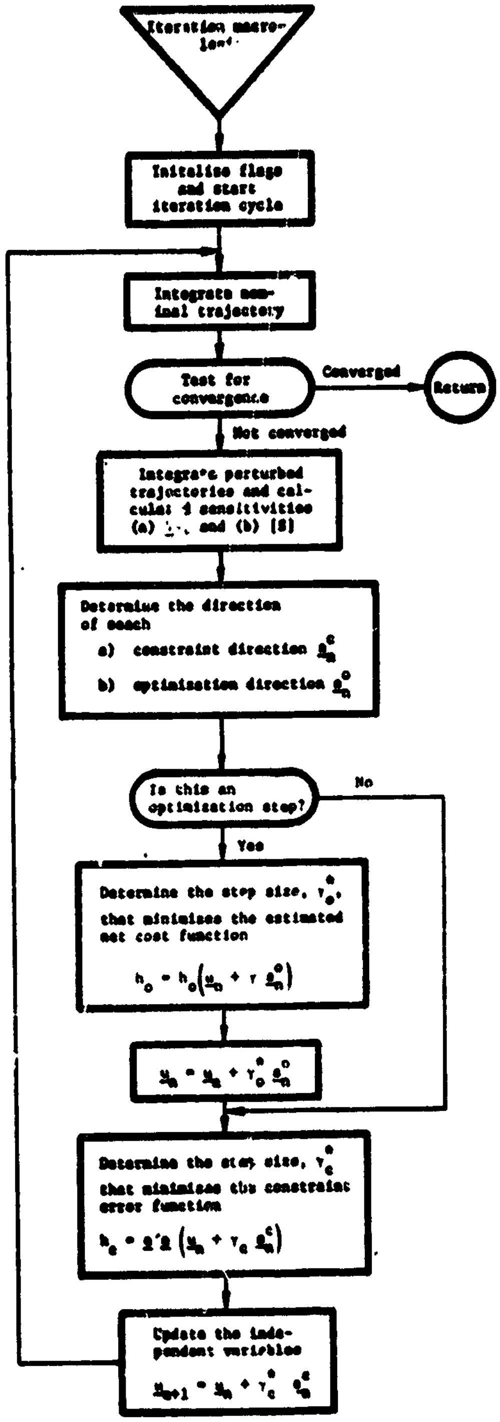


Figure VIII-7.- Macrologic of Projected Gradient Algorithm

The direction of the constraint-correction search emanates from $\underset{\sim}{u}_n^o$; 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 $\underset{\sim}{u}_n$ is used in conjunction with the error at $\underset{\sim}{u}_n^o$. Thus,

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

It can be shown by direct computation that

$$\tilde{P}\left(\underset{\sim}{u}_n\right)\underset{\sim}{s}_n^c = \underset{\sim}{s}_n^c, \tag{VIII-72}$$

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

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

$$\left(\underset{\sim}{s}_n^o\right)' \underset{\sim}{s}_n^c = 0. \tag{VIII-73}$$

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

$$\underset{\sim}{u}_{n+1} = \underset{\sim}{u}_n^o + \gamma_c \underset{\sim}{s}_n^c. \tag{VIII-74}$$

Figure VIII-8 geometrically illustrates a complete PGA iteration.

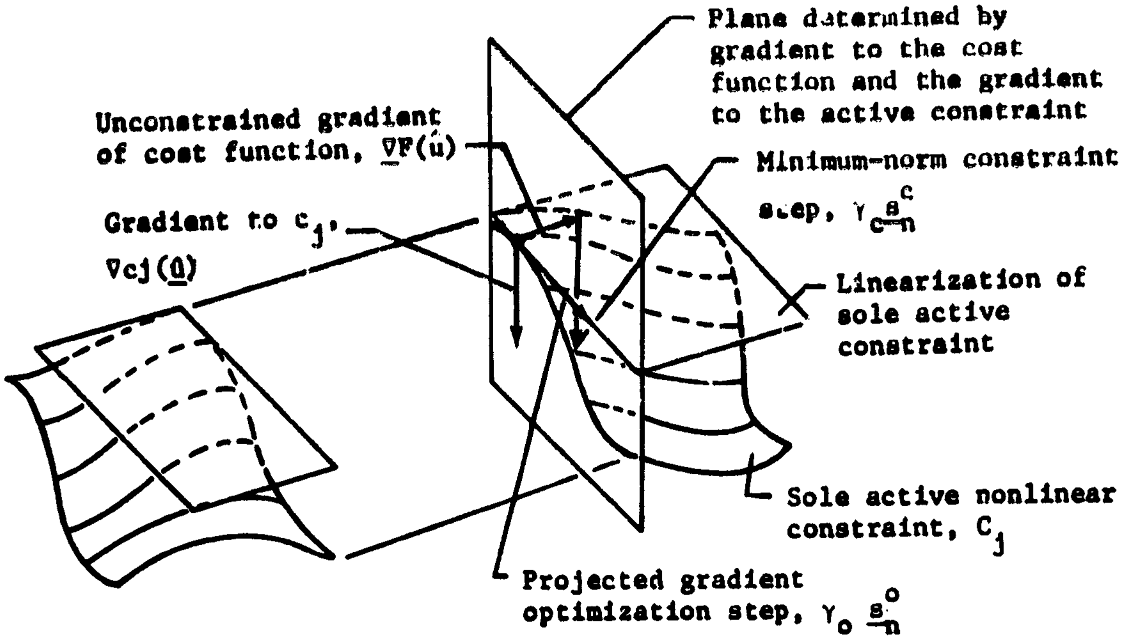


Figure VIII-8.- Complete PGA Iteration, Consisting of Optimization Step Followed by Constraint Step for $\hat{n}_a = 1$ and $m = 3$ (Feasible region is the unbounded region below the indicated nonlinear constraint manifold)

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

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

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

IX. REFERENCES

1. K. J. Cox: Space Shuttle Guidance, Navigation, and Control Design Equations. Vol. III Guidance Systems Analysis Branch, Guidance and Control Division, Johnson Space Center, Houston, Texas, 1973.

2. K. J. Cox: Digital Flight Control Software Design Requirements. JSC-07759, Revision A. Prepared by the Guidance and Control Systems Branch Avionics Systems Engineering Division, Johnson Space Center, Houston, Texas, 1973.

3. J. B. Rosen: The Gradient Projection Method for Nonlinear Programming. Part I - Linear Constraints. J. Soc, Ind. Appl. Math., No. 8, 1967, pp. 181-217.

4. J. B. Rosen: The Gradient Projection Method for Nonlinear Programming. Part II - Nonlinear Constraints. J. Soc, Ind. Appl. Math., No. 8, 1961, pp. 514-532.

5. 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.

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

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