AN OPTIMAL DESIGN METHODOLOGY FOR A
CLASS OF AIR-BREATHING LAUNCH VEHICLES

by

PHILIP DAVID HATTIS

B.S., Northwestern University
(1973)

M.S., California Institute of Technology
(1974)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1980

Signature of Author ______________________
Department of Aeronautics & Astronautics
June 1980

Certified by ______________________
Thesis Supervisor

Certified by ______________________
Thesis Supervisor

Certified by ______________________
Thesis Supervisor

Certified by ______________________
Thesis Supervisor

Certified by ______________________
Thesis Supervisor

Accepted by ______________________
Chairman, Departmental Graduate Committee

AN OPTIMAL DESIGN METHODOLOGY FOR A CLASS OF AIR-BREATHING LAUNCH VEHICLES

by

Philip David Hattis

Submitted to the Department of

Aeronautics and Astronautics on June, 1980,

in partial fulfillment of the requirements for

the degree of Doctor of Philosophy

ABSTRACT

A generalized two-point boundary value problem methodology applicable to the design of flight vehicles is developed and subsequently applied to a two-staged air-breathing launch vehicle. Methods are developed to simultaneously consider configuration and trajectory by treating geometry, dynamic discontinuities, and time-dependent flight variables all as controls to be optimized with respect to a single mathematical performance measure. In the air-breathing launch vehicle application, inequality constraint bounds are applied to dynamic pressure and specific force, and the effect of their variation is evaluated and discussed.

Thesis Supervisor: Wallace E. Vander Velde
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Steven R. Croopnick
Title: Staff, Flight Control and Dynamics Group, C.S. Draper Laboratory

Thesis Supervisor: Donald C. Fraser
Title: Lecturer, Aeronautics and Astronautics

Thesis Supervisor: Manuel Martinez-Sanchez
Title: Associate Professor of Aeronautics and Astronautics

Thesis Supervisor: Rudrapatna Ramnath
Title: Lecturer, Aeronautics and Astronautics and Mechanical Engineering

ACKNOWLEDGMENT

I wish to express mv sincere gratitude to my thesis committee. This includes Professor allace E. Vander Velde who served both as chairman and frequent technical consultant, Dr. Steven R. Croopnick who helped originate the topic and consistently provided technical support and advice, Dr. Donald C. Fraser whose frequent advice assured practical application and direction for the thesis, Professor Manual Martinez-Sanchez who provided much useful advice and many ideas on modelling and interpreting the behavior of air-breathing vehicles, and Dr. Rudrapatna Ramnath whose mathematical expertise was invaluable.

I also wish to express my deep appreciation to Robert A. Jones, William J. Small, and F. Steven Kirkham of the NASA Langley Research Center who provided much useful technical data and advice.

I would like to thank all the members of The Charles Stark Draper Laboratory Control and Flight Dynamics Division for their comments and suggestions during my doctoral program, especially for the moral support provided by Christopher B. Kirchwey, Harvey L. Malchow, Alexander Penchuk, Edward V. Bergmann, and fellow students Steven D. Ginter and Sean K. Collins.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Contract NAS9-13809 with the Johnson Space Center of the National Aeronautics and Space Administration. Publication of this thesis does not constitute approval by NASA or The Charles Stark Draper Laboratory, Inc., of the findings or conclusions contained herein.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF SYMBOLS

a	the dimension of the state vector (x)
a_d	the bound on the desired acceleration excluding gravity (the specific force bound)
A	(1) aspect ratio (2) a matrix term in the differential equation for Λ defined in Eq. (4-45)
A_{f_s}	fuselage surface area
A_S	scramjet inlet area
A_T	turbojet inlet area
A_P	second stage planform area
A_{P_0}	space shuttle planform area (a reference value)
A_{w_p}	area of first stage wing planform
b	the dimension of the control vector (u)
B	a matrix term in the variation equations defined in Eq. (4-70a)
c	the dimension of the parameter vector (p)
C	a vector used in the variation equations defined in Eq. (4-70b)
C_A	a constant to establish the desired weight given to specific force bound violation penalties
C_D	aerodynamic drag coefficient
C_{D_0}	aerodynamic parasitic drag coefficient

C_J	a coefficient to establish the desired weight given to cost improvement contributions to variation terms
C_L	aerodynamic lift coefficient
C_Ψ	a coefficient to establish the desired weight given to equality constraint violation improvements in variation terms
d	(1) a mathematical symbol for a total derivative (2) the dimension of the switch time vector (t_s)
D	total vehicle drag
D_1	vehicle first stage drag
D_2	vehicle second stage drag
D_w	drag due to first stage wing
f	derivative of state vector with respect to time
f_s	the derivative of the state vector with respect to time evaluated at the switch points (a vector)
f_{s_i}	the derivative of the state vector with respect to time evaluated at the i^{th} switch point
f_p	the partial derivative of the vector f with respect to the vector p (a matrix)
f_x	the partial derivative of the vector f with respect to the vector x (a matrix)
F	a symbol for the matrix f_x
F_g	the gravitational force component
F_r	the radial force component
F_{sp}	the specific force
F_θ	the tangential force component
g	(1) the gravitational acceleration (2) a vector used in the variation equations defined in Eq. (4-56a)

g_0	the gravitational acceleration at the earth's surface
G	(1) the universal gravitation constant
	(2) the partial derivatives of the vector f with respect to the vector u (a matrix)
h_S	the scramjet inlet height
h_T	the turbojet inlet height
h_0	the desired orbital altitude
H	the Hamiltonian defined in Eq. (4-3)
H_p	the partial derivatives of H with respect to the vector p (a vector)
H_u	the partial derivatives of H with respect to the vector u (a vector)
i	a subscript denoting the element of a vector
I_{JJ}	an influence function of cost (a scalar)
I_{sp_R}	rocket specific impulse
I_{sp_S}	scramjet specific impulse
I_{sp_T}	turbojet specific impulse
$I_{\psi J}$	an influence function of constraint and cost (a vector)
$I_{\psi\psi}$	an influence function of constraints (a matrix)
J	the mathematical cost imposed on the system
$\bar{J}$	a mathematical cost of J plus adjoined terms
J_s	a specified cost
k	the dimension of the vector ψ
K	a symbol for the matrix f_p
ℓ	fuselage length

LIST OF SYMBOLS (CONT.)

L	(1) the total vehicle lift
	(2) the distributed mathematical cost term
L_p	the partial derivatives of the distributed cost with respect to the vector p (a vector)
L_u	the partial derivatives of the distributed cost with respect to the vector u (a vector)
L_w	the lift due to the first stage wing
L_x	the partial derivatives of the distributed cost with respect to the vector x (a vector)
L_1	(1) net first stage lift
	(2) penalty term on dynamic pressure
L_2	(1) net second stage lift
	(2) penalty term on specific force
m	the net vehicle mass
m_{d_s}	the second stage dry mass
m_{d_0}	the nominal shuttle dry mass
m_{f_a}	the first stage fuselage mass due to surface area
m_{f_s}	the second stage propellant tank fuel mass capacity
m_{f_t}	the first stage dry mass contribution of the fuel tanks
m_i	the i^{th} component of the three propellant mass states
m_S	the dry mass of the scramjet
m_T	the dry mass of the turbojet
m_w	the dry mass of the first stage wing
m_1	the turbojet fuel mass
m_2	the scramjet fuel mass
m_3	the rocket propellant mass

M	a matrix term used in the variation equations defined in Eq. (4-56c)
M_e	the earth's mass
M_0	the free stream Mach number
M_1	the Mach number immediately following the nose shock (and also at the air-breathing engine inlets)
N	the aerodynamic force normal to the vehicle body x-axis
p	the parameter vector
$\dot{p}_p$	the partial derivatives of the parameter vector time derivatives with respect to the parameter vector (a matrix)
$\dot{p}_x$	the partial derivatives of the parameter vector time derivatives with respect to the state vector (a matrix)
p_1	the first stage nose angle
p_2	the first stage fuselage width
p_3	the scramjet inlet height
p_4	the turbojet inlet height
p_5	the first stage wing span
p_6	the first stage wing delta angle
p_7	the ratio of the turbojet tank volume to total first stage tank volume
p_8	the first stage fuselage length
p_9	the second stage maximum propellant tank mass capacity
p_{10}	the maximum rocket thrust
P	the aerodynamic force parallel to the body x-axis
P_1	the pressure after the first-stage nose shock (and at the air-breathing engine inlets)
q	dynamic pressure
q_d	the bound on the desired dynamic pressure range

q_0 the free stream dynamic pressure

Q a constant to establish the desired weight given to dynamic pressure bound violation penalties

r the distance from the earth's center

r_{f_S} the stoichiometric scramjet fuel/air mass ratio

r_{f_T} the stoichiometric turbojet fuel/air mass ratio

r_S the scramjet fuel tank mass/volume ratio

r_T the turbojet fuel tank mass/volume ratio

R the universal gas constant

R_e the earth's radius

R_f the turbojet fuel tank volume/total first stage fuel tank volume ratio

S step size

S_T advanced turbojet mass scaling factor

t_s the vector of switch times

t_{s_i} the i^{th} component of t_s

T (1) the matrix transpose sign when used as a superscript

(2) the total vehicle thrust

T_{max_s} the second-stage maximum thrust

T_R the rocket thrust

T_S the scramjet thrust

T_T the turbojet thrust

T_0 the free stream temperature

T_1 the post first-stage nose shock temperature (also the temperature at the air-breathing engine inlets)

u	the control vector
u_f	an unbounded fuel flow function
u_0	a unit step function
u_1	air velocity after first-stage nose shock compression (also the velocity at the air-breathing engine inlets)
U	a matrix to weight different elements of the control variation vector
V	a matrix to weight different elements of the parameter variation vector
V_f	the first stage fuselage volume
V_T	the second stage propellant tank volume
V_0	the shuttle nominal volume (a reference value)
w_f	the first stage fuselage width
w_s	the first stage wing span
x	the state vector
x_1	the distance from earth's center
x_2	the radial velocity
x_3	the orbital angle (in polar coordinates—two dimensional)
x_4	the angular velocity
x_5	the turbojet fuel mass
x_6	the scramjet fuel mass
x_7	the rocket fuel mass
z	an intermediate vector in the derivation of the variation equations
α	the angle of attack
α_n	the first-stage nose angle

α_t the first-stage tail angle

β_s the first-stage nose shock angle

γ_0 the free stream specific heat ratio

δ (1) a mathematical symbol for a variation

(2) the vehicle body x-axis angle relative to the local horizontal

δ_a the first stage wing delta angle

η a Lagrange multiplier scalar

θ the orbital angle (in polar coordinates—two dimensional)

θ_f the first stage nose flow deflection angle

λ the costate vector

Λ a matrix influence function of the equality constraints

Λ_0 a coefficient of some terms in the Λ_1 boundary condition

Λ_1 a state equality constraint matrix influence function

Λ_2 a parameter equality constraint matrix influence function

ν a Lagrange multiplier vector

ρ air density

ρ_0 free stream air density

ρ_1 air density after the first-stage nose shock compression (also the density at the air-breathing engine inlets)

τ the terminal time of the trajectory

ϕ the terminal cost terms

ϕ_x the partial derivatives of the terminal cost terms with respect to the states (a vector)

Φ proportion of stoichiometric fuel mixture used (where $\Phi=1$ is assumed as an upper limit)

Ψ	the equality constraint vector
Ψ_p	the partial derivatives of the Ψ vector with respect to parameters (a matrix)
Ψ_x	the partial derivatives of the Ψ vector with respect to states (a matrix)
ω_e	the earth's angular velocity in an inertial frame
ω_0	the desired orbital angular velocity
Ω	the state integration cutoff function
Ω_x	the derivative of Ω with respect to states (a vector)
∂	the partial derivative sign
$\cdot$	when over a symbol, a derivative with respect to time
$+$	as a superscript, indicates a small positive time displacement
$-$	as a superscript, indicates a small negative time displacement

CHAPTER I

INTRODUCTION

At a time when the first reusable space transportation system is in the final developmental stages, considerable attention is being given to new space applications. Investigations of required flight frequencies, orbital payload masses, and associated transportation costs have been performed[1] with the presumption that any one of several proposed new uses of the space environment will occur, resulting in vastly expanded space operations.

With the probability that the next quarter century will lead to many imaginative and unanticipated uses of space, it seems likely that improvements in performance over what can be achieved with the space shuttle will be desirable, and probably necessary. Given the uncertainty of the economic evaluation methods (i.e., trying to establish dollar figures for development and flight costs years before actual flight), it seems reasonable to judge the design requirements of advanced space transportation concepts, where approval is still years in the future, on strict engineering performance requirements. In particular, the propellant mass consumption requirement for a variety of propulsion methods, along with the sensitivity of this quantity to the application of constraints on specific force and dynamic pressure, seems an unambiguous performance measure which will permit a lucid comparison of alternate space transportation concepts.

One class of launch vehicle that promises considerable improvement in the propellant mass performance measure when compared to space shuttle technology is the air-breathing launch system.

Since the end of World War II, much attention has been given to high performance air-breathing engines for both aircraft and launch vehicle propulsion. Most interest has been in the area of ramjet propulsion, due to its simplicity and high performance to beyond Mach 4.[2] In the early 1960's interest developed in the possibility of ramjet designs using supersonic combustion (scramjets) due to the decreased inlet losses expected, and improved specific impulse at high Mach number. The potential of operating to beyond Mach 10 was recognized.[3]

As spaceflight became routine in the mid 1960's, and the potential for vastly increased flight activity was foreseen, consideration was given to the possible economic benefit of space operations similar to the routine commercial aircraft operations.[4] The potential of an orbital craft that used an aircraft type launching platform was recognized.

At about the same time NASA was investigating design concepts for the space shuttle. To support the investigation, work was done at NASA's Langley Research Center on design concepts for a two stage vehicle with an air-breathing first stage, and a rocket-powered second stage that would achieve orbit. Two papers were submitted to the AIAA Advanced Space Transportation meeting in 1970 that gave considerable thought to the necessary technology, the general geometry, the development requirements, the operational considerations, and the expected performance of the air-breathing system as compared to more conventional launch concepts.[5,6]

Due to concern about the technology development time, along with a general conservatism about proven vs. new systems, the air-breathing system was dropped as a contender for the space shuttle design. However, technology development has continued on a low key in the context of hypersonic transport research. The technology and potential for hypersonic transports were assessed at Lockheed in a 1970 report,[7] and at NASA's Langley Research Center propulsion/airframe integration concepts have been assessed along with thermal and structural problems,[8-11] and supersonic hydrogen fuel mixing and combustion.[12]

Recently, the case has been argued for the development of a flight-test platform for experimental hypersonic propulsion, structural, and fuel systems.[13] It seems likely that the technology base for construction of reusable air-breathing launch vehicles will soon exist.

As generally proposed, the air-breathing launch vehicle consists of a two-staged horizontal takeoff configuration with air-breathing capability on the first stage and with the second stage operating as a conventional rocket. The first stage would have turbojet engines for flight to about Mach 3. At some point above Mach 1, dual mode ramjet engines, with both subsonic and supersonic combustion capability are phased into use. At some point between Mach 4 and Mach 12, separation of the stages occurs, and the first stage returns in an aircraft type operation, while the second stage attains orbital velocity on rocket power.

Almost without exception, the first stage is presumed to use liquid hydrogen fuel at the higher Mach numbers because of the capacity of the fuel as a heat sink in active structural cooling, along with its high energy/mass density. At low Mach numbers, however, consideration has been given to the alternative fuels due to the low volumetric energy capacity of hydrogen.

Given the potential of full system reusability, more routine aircraft type operation, safer abort capability, and improved performance values, it seems appropriate at this time to develop a design methodology for the optimization of two-staged air-breathing launch vehicles based on propellant mass performance criteria. (A two-staged concept is considered to prevent the fuel penalty of transporting the dry weight of heavy air-breathing propulsion systems to orbit, since they have little utility in flight near orbit.) Consideration will be given to trajectory shape, aerodynamic design, propulsion system performance, and propulsion system changeover points. Since the overall problem consists of a complicated nonlinear two point boundary value problem, it is essential that efficient methods be devised to find the optimal solutions, to help keep the cost of computation down. It is believed that any efficient solution techniques derived for this problem will represent a contribution to the solution of many similar problems.

Many aspects of flight vehicle optimization have been addressed in the last twenty-five years, though generally individually. The aerospace Research Laboratory supported the development of an optimization technique to evaluate the configuration of a two-dimensional hypersonic cruise vehicle, with the cruise condition specified, using the lift-to-drag ratio as the primary cost consideration.[14] Optimal rocket trajectories, with aerodynamic effects included, have been evaluated for fixed rocket configurations.[15] Minimum time to climb and maximum altitude paths have been evaluated for air-breathing crafts (fighters) with specified geometry and performance models.[16] Recently, attempts have been made to treat trajectory and configuration simultaneously for a single stage to orbit launch vehicle[17] although in this case the method involved iterative optimization of each part separately with intermediate efforts to match the results of the different computations. When the aggregate objectives of these studies are considered, the need for a unified mathematical treatment to simultaneously optimize flight vehicle configuration and trajectory is clear.

A solid data base exists for the general performance characteristics of hypersonic lifting vehicles and propulsion systems. General conceptualization studies have been done for the design of air-breathing launch vehicles. Also, operational requirements of reusable space transportation systems have been established. The need for post-shuttle launch vehicles with improved performance seems likely to develop. Therefore, it appears to be timely to develop an optimal design methodology for a reusable air-breathing launch platform.

In the following material a methodology to establish an optimal configuration and flight path of sophisticated systems with complicated dynamics is developed and subsequently applied to a class of air-breathing launch vehicles. Chapter II specifies a parametric vehicle model, suitable for the optimization methodology, used for demonstration of the technique. Chapter III defines the system dynamics model used. Chapter IV outlines the mathematical development of the optimization algorithm, while Chapter V relates the material in Chapters II and III to the equations in Chapter IV. Chapters VI and VII discuss and interpret the results, and note numerical difficulties to be avoided when implementing the methodology. Conclusions are drawn, future research extensions are suggested and a listing of a computer algorithm is given in Chapter VIII and the Appendix.

CHAPTER II

THE VEHICLE AND ENVIRONMENT MODEL

In the following material, a parametric model of the two-staged air-breathing launch vehicle is developed, and the vehicle thrust and aerodynamic properties are defined, in a form suitable for computer algorithm application.

2.1 An Overview

The class of vehicle to be considered is a two-staged configuration, the second of which is a conventional rocket propelled orbiter with reentry glider configuration resembling the space shuttle. The first stage is propelled by air-breathing propulsion systems. Turbojets operate at low Mach numbers. Convertible subsonic/supersonic combustion ramjets (scramjets) are available to operate at high Mach number. Overlapping operation of the two air-breathing propulsion systems is possible at intermediate Mach numbers. The vehicle is configured for horizontal takeoff. The payload to be delivered to low earth orbit is assumed comparable to the space shuttle.

The vehicle parametric model is a function of aerodynamic geometry, propulsion geometry, fuel capacity, and propellant properties.

A shuttle-derived liquid hydrogen/liquid oxygen system is assumed for the second stage. Due to the heat sink capacity of liquid hydrogen upon conversion to combustible fuel, and the anticipated high heat loading on hypersonic air frames and propulsion systems, liquid hydrogen is assumed as the propellant in the scramjet propulsion mode. For low Mach number, in the turbojet mode, a hydrocarbon fuel is assumed due to its high volumetric energy capacity, its simpler handling and storage requirements, and its ready use by existing and extrapolated turbojet technology.

The propulsion dynamics are modeled to allow for the three thrust magnitude discontinuities which are associated with changes in propulsive mode. The first discontinuity following takeoff represents transition from the turbojet-only mode to mixed turbojet/scramjet operation. The second discontinuity represents transition to the scramjet-only mode. The third discontinuity represents simultaneous staging and initiation of the second-stage rocket thrust.

The atmosphere is modeled as variable with altitude only, and is assumed to rotate with the earth's surface.

Gravity is assumed to depend on altitude only, with variations modeled on a homogeneous sphere representation of the earth.

2.2 The First-Stage Model

A principal objective in the development of a parametric model of the physical characteristics of the air-breathing stage has been to obtain a reasonable representation of vehicle performance while requiring a minimum quantity of independent parameters. The desire to hold down the number of parameters is associated with the strong influence of the dimension of the parameter vector on computation time.

The basic dynamic behavior of the vehicle can be determined by giving consideration to wing, fuselage, and propulsion system dimension, along with component mass models, propulsion system performance data, and aerodynamic characteristics as a function of exterior geometry. If component geometry and aerodynamic models are limited to two dimensions, a further savings in parameter vector dimension is possible with little loss in solution accuracy, since very little cross coupling between down track and cross track motion would normally be expected.

2.2.1 The Wing

A simple body-integrated delta wing is considered, and the resulting configuration is treated as the sole source of aerodynamic lift and drag. The data on the propulsion system, given later, specifies installed performance. Consequently, most of the expected drag not attributable to the wing is incorporated into the propulsion model. Data on body-integrated delta wing lift and drag coefficients is available as a function of Mach number, aspect ratio, and angle of attack, as long as the angle of attack does not stray too far from zero.

The interpolation variables are A, and M_0, where

A = aspect ratio

M_0 = free stream Mach number

One has

$$A = \frac{w_s^2}{A_{w_p}} \tag{2-1a}$$

$$C_L = \frac{L_w}{q_0 A_{w_p}} \tag{2-1b}$$

$$C_D = \frac{D_w}{q_0 A_{w_p}} \tag{2-1c}$$

where

w_s = wing span

A_{w_p} = area of wing planform

q_0 = free stream dynamic pressure

L_w = lift due to wing

D_w = nonparasitic drag due to wing

The resultant lift and drag equations are

$$L_1 = \left[\left(\frac{dC_L}{d\alpha}\right)\alpha\right] q_0 A_{w_p} \tag{2-2a}$$

$$D_1 = \left[\left(\frac{dC_D}{dC_L^2}\right)C_L^2 + C_{D_0}\right] q_0 A_{w_p} \tag{2-2b}$$

where

L_1 = net lift on first stage

D_1 = net drag on first stage

C_{D_0} = parasitic drag term

The data in Table II-1[18] is the basis of an aerodynamic data linear interpolation scheme used to calculate lift and drag coefficients.

The interpolation results are the quantities $dC_L/d\alpha$, and dC_D/dC_L^2, where

C_L = lift coefficient

C_D = drag coefficient

α = angle of attack

Table II-1. Delta wing lift and drag coefficients.

$\frac{dC_L}{d\alpha}$ data →

↓ Aspect Ratio	Mach Number 0.25	0.6	0.7	0.9	1.1	1.2	1.5	2.0	4.0	8.0
0.5	0.0120	0.0130	0.0140	0.0145	0.0155	0.0150	0.0145	0.0140	0.0120	0.0075
1.0	0.0200	0.0230	0.0250	0.0280	0.0320	0.0300	0.0280	0.0240	0.0150	0.0075
2.0	0.0400	0.0440	0.0480	0.0530	0.0680	0.0620	0.0530	0.0380	0.0175	0.0075
3.0	0.0530	0.0590	0.0630	0.0720	0.0870	0.0740	0.0620	0.0420	0.0180	0.0075
4.0	0.0650	0.0700	0.0730	0.0810	0.0980	0.0800	0.0700	0.0440	0.0190	0.0075

$\frac{dC_D}{dC_L^2}$ data →

↓ Aspect Ratio	Mach Number 0.25	0.6	0.7	0.9	1.1	1.2	1.5	2.0	4.0	8.0
0.5	0.77	0.79	0.80	0.86	0.89	0.92	1.00	1.10	1.45	2.00
1.0	0.47	0.48	0.42	0.39	0.41	0.43	0.58	0.69	1.10	1.99
2.0	0.30	0.30	0.27	0.24	0.26	0.27	0.34	0.48	1.00	1.99
3.0	0.25	0.25	0.23	0.21	0.22	0.24	0.31	0.46	1.00	1.99
4.0	0.22	0.22	0.21	0.195	0.21	0.22	0.28	0.45	1.00	1.99

Data for evaluation of C_{D_0} are given in Table II-2[19], and are based on shuttle zero angle of attack drag properties with a scaling factor used to approximate the effect of assumed improved aerodynamic properties of the air-breathing stage. (The shuttle parasitic drag is very high because of the silica thermal protection tile surface properties).

Table II-2. First stage parasitic drag coefficients. (Based on scaled shuttle data—scale factor = 1/3).

Mach Number	C_{D_0}	Mach Number	C_{D_0}
0.25	0.0203	1.30	0.0521
0.60	0.0208	1.50	0.0516
0.80	0.0228	2.00	0.0470
0.90	0.0268	3.00	0.0378
0.95	0.0351	4.00	0.0327
1.05	0.0492	5.00	0.0295
1.10	0.0507	8.00	0.0263
1.20	0.0518	10.00	0.0260

Similar to the parasitic drag scaling factor, a scaling factor is allowed for the net lift and drag computations to approximate aerodynamic refinement of wing design or more complex wing geometries (~0.7 for drag; ~1.7 for lift).

The result of the data format is a requirement for two wing-associated independent parameters, wing span and wing area. However, for a simple delta wing one has

$$A_{w_p} = \frac{w_s^2}{(4 \tan \delta_a)} \qquad (2-3)$$

where

δ_a = delta wing angle, the forward wing edge angle with the fuselage

The quantity δ_a is used in place of the wing planform area as one of the two wing-associated parameters.

Without going into detailed analysis of thermal protection requirements, one can roughly judge wing mass from wing planform area. A useful approximation in terms of slugs mass vs. square feet of wing area is[20]

$$m_w = 0.25\ A_{w_p} \tag{2-4}$$

where

m_w = mass of wing

2.2.2 The Fuselage

For numerical simplicity it is desirable to keep the vehicle aerodynamics model two dimensional. This consideration mandates a fuselage design with constant geometry in a x-z body axis plane cross section as shown in the side view of Figure II-1. Therefore, a representation of the fuselage geometry simply requires a shape for the upper and lower surfaces. The complete characterization of the fuselage physical properties also requires internal propellant tank capacity, and dry mass properties.

For vehicles expected to use air-breathing propulsion at high Mach numbers, the body geometry must incorporate features beneficial to propulsion system performance. The forward surface should provide some compression prior to the propulsion system ingestion of air. The aft surface should serve as an expansion region extending beyond the propulsion system internal expansion nozzle. Good propulsion system behavior requires much attention to the geometry of the aft expansion surface to avoid undesirable effects such as separation. Consequently, to retain simplicity in the parametric model, the aft surface angle is assumed to be fixed, and is incorporated into the installed propulsion system performance data. The vehicle nose angle and the overall fuselage length are left as parameters to be evaluated. The general vehicle geometric configuration is shown in Figure II-1. Note that the propulsion system is assumed to be under the fuselage, permitting a flat upper suface.

The propellant tank capacity is constrained to match the propellant requirements. To allow the volume to meet the constraint, fuselage width is made a parameter.

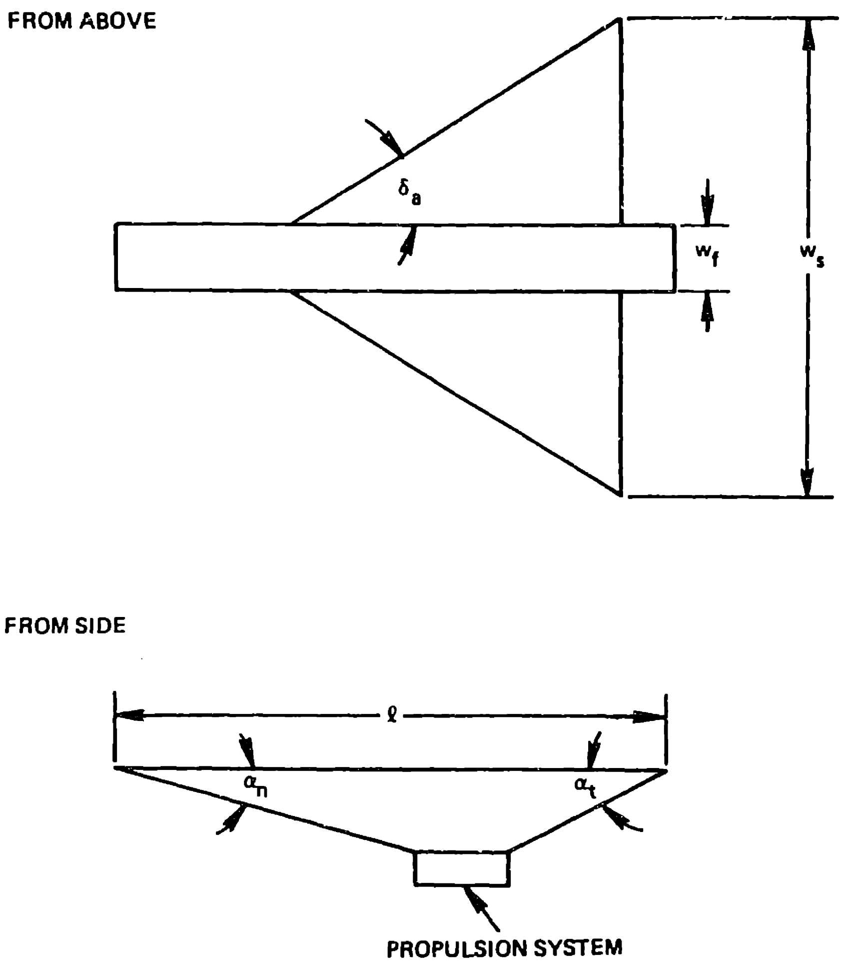


Figure II-1. First stage cross-section.

One has

$$V_f = \frac{w_f \ell^2 \sin(\alpha_n) \sin(\alpha_t)}{2 \sin(\alpha_n + \alpha_t)} \tag{2-5a}$$

$$A_{f_s} = w_f \ell \left(1 + \frac{\sin(\alpha_n) + \sin(\alpha_t)}{\sin(\alpha_n + \alpha_t)}\right) + \frac{2V_f}{w_f} \tag{2-5b}$$

where

V_f = fuselage volume

A_{f_s} = fuselage surface area

w_f = fuselage width

ℓ = fuselage length

α_n = nose angle

α_t = tail angle

The most important aerodynamic property of the fuselage not incorporated into propulsion performance models is the free stream compression by the vehicle nose. On the basis of the already assumed simple vehicle geometry, one can use oblique shock/normal shock/Prandtl-Meyer expansion fan theory for the supersonic compression calculations prior to the propulsion system ingestion of the flow. (Perfect gas behavior assumption is implied). Since the nose angle solution value is likely to be small, and since vehicle rotational dynamics will not be considered, eliminating the need for body pressure distributions, subsonic compression effects are neglected. Thus, one need only consider the nose effect when the free stream Mach number exceeds one. If the nose angle plus the angle of attack exceeds zero, a shock will result, otherwise an expansion will result. The shock can be oblique or normal depending on the turning angle of the free stream, and the Mach number. The following equations apply to normal shocks[21]

$$M_1 = \left(\frac{1 + \frac{\gamma_0 - 1}{2} M_0^2}{\gamma_0 M_0^2 - \frac{\gamma_0 - 1}{2}} \right)^{1/2} \tag{2-6a}$$

$$T_1 = \left(1 + \frac{2(\gamma_0 - 1)}{(\gamma_0 + 1)^2} \frac{(\gamma_0 M_0^2 + 1)}{M_0^2} (M_0^2 - 1) \right) T_0 \tag{2-6b}$$

$$\rho_1 = \left(\frac{(\gamma_0 + 1) M_0^2}{(\gamma_0 - 1) M_0^2 + 2} \right) \rho_0 \tag{2-6c}$$

$$P_1 = \rho_1 R T_1 \tag{2-6d}$$

ıere

M_0 = free stream Mach number

M_1 = post nose compression Mach number

T_0 = free stream temperature

T_1 = post nose compression temperature

ρ_1 = post nose compression density

P_1 = post nose compression pressure

R = universal gas constant

γ_0 = free stream specific heat ratio

The following equations apply to oblique shocks[22]

$$M_1 = \left(\frac{1}{\sin^2 (\beta_s - \theta_f)} \frac{1 + \frac{\gamma_0 - 1}{2} M_0^2 \sin^2 \beta_s}{\gamma_0 M_0^2 \sin^2 \beta_s - \frac{\gamma_0 - 1}{2}} \right)^{1/2} \quad (2\text{-}7a)$$

$$T_1 = \left(1 + \frac{2(\gamma_0 - 1)}{(\gamma_0 + 1)^2} \frac{(\gamma_0 M_0^2 \sin^2 \beta_s + 1)}{M_0^2 \sin^2 \beta_s} (M_0^2 \sin^2 \beta_s - 1) \right) T_0 \quad (2\text{-}7b)$$

$$\rho_1 = \left(\frac{(\gamma_0 + 1) M_0^2 \sin^2 \beta_s}{(\gamma_0 - 1) M_0^2 \sin^2 \beta_s + 2} \right) \rho_0 \quad (2\text{-}7c)$$

$$P_1 = \rho_1 R T_1 \quad (2\text{-}7d)$$

$$\tan \theta_f = 2 \cot \beta_s \frac{M_0^2 \sin^2 \beta_s - 1}{M_0^2 (\gamma_0 + \cos 2\beta_s) + 2} \quad (2\text{-}7e)$$

ɜre

θ_f = flow deflection angle

β_s = shock angle

The following equations apply for expansion[23]

$$\theta_f = \sqrt{\frac{\gamma_0 + 1}{\gamma_0 - 1}}\left(\tan^{-1}\left(\frac{\gamma_0 - 1}{\gamma_0 + 1}(M_0^2 - 1)\right)^{1/2} - \tan^{-1}\left(\frac{\gamma_0 - 1}{\gamma_0 + 1}(M_1^2 - 1)\right)^{1/2}\right)$$

$$- \left(\tan^{-1}(M_0^2 - 1)^{1/2} - \tan^{-1}(M_1^2 - 1)^{1/2}\right) \quad (2\text{-}8a)$$

$$T_1 = \left(\frac{1 + \frac{\gamma_0 - 1}{2} M_0^2}{1 + \frac{\gamma_0 - 1}{2} M_1^2}\right) T_0 \quad (2\text{-}8b)$$

$$\rho_1 = \rho_0\left(\frac{T_1}{T_0}\right)^{\frac{1}{(\gamma_0 - 1)}} \quad (2\text{-}8c)$$

$$P_1 = \rho_1 R T_1 \quad (2\text{-}8d)$$

Also one needs a relation for the flow turning angle

$$\theta_f = \alpha_n + \alpha \quad (2\text{-}9)$$

If θ_f is negative one should use Eq. (2-8) with an iterative solution of Eq. (2-8a) required. If θ_f is not negative, either Eq. (2-7) or (2-6) should be used.

The distinction between use of Eq. (2-7) and (2-6) is dependent on whether a solution to Eq. (2-7) exists. If there is a solution then Eq. (2-7) is used. Otherwise Eq. (2-6) is used. (The principle that the weak shock solution applies when possible is used). To determine if a solution to Eq. (2-7) exists, it is necessary to investigate Eq. (2-7e) in detail. Differentiation, and evaluation where the derivative $d\theta_f/d\beta_s$ vanishes yields a value of β_s for the maximum θ_f that permits a real solution

$$\sin\left(\beta_{s_{max}}\right) = \left[\left(\{(\gamma_0 + 1)M_0^2\} - 4 + \left((\gamma_0 + 1)\left(\{(\gamma_0 + 1)M_0^4\} + \{8(\gamma_0 - 1)M_0^2\} + 16\right)\right)^{1/2}\right)/(4\gamma_0 M_0^2)\right]^{1/2} \quad (2\text{-}10)$$

Substitution of Eq. (2-10) into Eq. (2-7e) yields a maximum value for θ_f for which Eq. (2-7) can be applied.

The fuselage mass can be modeled to be a function of surface area, propellant tank volume, and fuel type within a given tank.

Using the same approximation as is used in Eq. (2-4) one gets

$$m_{f_a} = 0.25\, A_{f_s} \tag{2-11}$$

where

m_{f_a} = mass of the fuselage due to area

The mass contribution of the propellant tanks must be based on the volume available to each tank. Something less than the entire fuselage volume is available for fuel storage due to the space requirement of various nonpropellant systems. A simple approximation is to presume that 80% of the total fuselage volume is available for fluid storage. Of the volume available, allocation of space must be made for both the turbojet and scramjet fuels, each having tanks with different mass properties due to the different fuel handling requirements. A parameter is created to specify the relative space allocation.

One gets

$$m_{f_t} = 0.8(r_T R_f + r_S(1 - R_f))V_f \tag{2-12}$$

where

m_{f_t} = mass of fuselage due to propellant tanks

R_f = turbojet fuel tank volume/total fuel volume ratio

r_T = turbojet tank mass/volume ratio

r_S = scramjet tank mass/volume ratio

r_T and r_S are constants requiring specification. Since the scramjet fuel is cryogenic, with the associated added storage burdens (e.g., insulation), one expects r_S to be larger than r_T. Analysis of material and storage requirements suggest

$$0.01 < r_T < 0.02 \quad \text{in slugs/ft}^3 \tag{2-13a}$$

$$0.03 < r_S < 0.04 \tag{2-13b}$$

2.2.3 The Propulsion Systems

The information necessary to characterize the propulsion system includes the external geometry, the mass, and the operational performance.

The geometry information can be limited to inlet area in the simplest case. On the basis of the two-dimensional aerodymanic approximations already made, the required information can be stated by providing inlet width and height for each propulsion system. Since the entire fuselage width is expected to provide precompression of the free stream, a further simplification is to assume the propulsion system width is equal to that of the fuselage, and all the compressed flow enters the system. One therefore requires two geometric parameters to define the air breathing propulsion system: the turbojet inlet height, and the scramjet inlet height. One gets

$$A_T = h_T w_f \tag{2-14a}$$

$$A_S = h_S w_f \tag{2-14b}$$

where

A_T = turbojet inlet area

A_S = scramjet inlet area

h_T = turbojet inlet height

h_S = scramjet inlet height

Models of propulsion system mass are somewhat speculative due to the experimental nature of scramjets and advanced lightweight turbojets. However, some data has been generated, and can be suitably simplified for use in the present problem.

In the case of the turbojet, one can extrapolate advanced lightweight designs from tabulated data of existing turbojet designs and historical weight reduction trends,[24] to derive an approximate system mass per unit of inlet area. The result yields

$$m_T = (7.5) S_T A_T \quad \text{with: } m_T \text{ in slugs}, \; A_T \text{ in ft}^2 \tag{2-15}$$

where

m_T = mass of turbojet in slugs

S_T = advanced turbojet scale factor $\approx$ 2/3

The scramjet mass model must be extrapolated from data on experimental configurations. Information on modularized designs has been produced[25] and can be converted into a function of inlet area. The models lead to rather heavy systems designed to tolerate heat flux loads expected until well above Mach 6. The result is

$$m_S = ((15.2\ h_S) - (4.6/h_S))w_f \tag{2-16}$$

where

m_S = scramjet mass in slugs if h_S, w_f are in feet

As h_S drops below about 1.25 ft, then the accuracy of Eq. (2-16) is rapidly reduced due to the growing relative contribution of propulsion system support equipment. A lower bound on mass per width can be used to resolve the problem. Analysis of the support system mass requirements suggests the bound[26]

$$m_S \geq (15.0)w_f \quad (m_S \text{ in slugs}) \tag{2-17}$$

The performance data for propulsion systems intended for a body-integrated design application is usually presented in the form of installed specific impulse[27] with the implicit assumption of nearly stoichiometric fuel/air mixture ratios. The data includes losses due to engine cowl effects and nozzle expansion. The data can be made to be a function of the aft fuselage angle, but a desireable angle is usually given as

$$\alpha_t = 12° \tag{2-18}$$

On the basis of approximate curve fits for the Mach number dependence of the specific impulse data one gets

$$I_{SP_T} = 3800 - (300\ M_1) - (100\ M_1^2) \tag{2-19a}$$

$$I_{SP_S} = 15000(M_1^{1.6})e^{-1.73(M_1^{0.52})} \tag{2-19b}$$

where

I_{SP_T} = turbojet installed specific impulse

I_{SP_S} = scramjet installed specific impulse

Accurate computation of system thrust generally requires knowledge of the variation of specific impulse with fuel/air mix. In most cases, however, the specific impulse has little change near the stoichiometric mixture ratio. As an approximation, the impulse is not made a function of the fuel/air mix. This gives

$$T_T = \rho_1 u_1 A_T I_{SP_T} g_0 r_{f_T} \Phi \tag{2-20a}$$

$$T_S = \rho_1 u_1 A_S I_{SP_S} g_0 r_{f_S} \Phi m_{CR} \tag{2-20b}$$

where

T_T = turbojet thrust

T_S = scramjet thrust

ρ_1 = air density after nose compression

u_1 = air velocity after nose compression

g_0 = gravity at earth's surface

r_{f_T} = stoichiometric turbojet fuel/air mass ratio (≈ 0.0633 for octane)

r_{f_S} = stoichiometric scramjet fuel/air mass ratio (≈ 0.027778)

Φ = proportion of stoichiometric fuel mixture used (where $\Phi = 1$ is assumed as an upper limit)

m_{CR} = scramjet mass capture ratio (proportion of inlet flow used in engine)

The quantity Φ is used in both parts of Eq. (2-20) since it is assumed that if the scramjet and turbojet operate together, then they are equally throttled. The mass capture ratio for the scramjet is required to model the spillage effects at low supersonic Mach numbers typical of these engines. (Data is available on the Langley three dimensional propulsion system design and is given in Table II-3).[28]

2.2.4 A Summary of First Stage Parameters

If the independent geometric parameters are assumed to be elements of a vector p, one can define the following:

Table II-3. Scramjet mass capture ratio.

Mach Number	Mass Capture Ratio
2.5	0.35
3.5	0.60
5.0	0.80
5.5	0.85
8.0	0.95

p_1 = nose angle (α_n)

p_2 = fuselage width (w_f)

p_3 = scramjet height (h_S)

p_4 = turbojet height (h_T)

p_5 = wing span (w_s)

p_6 = delta wing angle (δ_a)

p_7 = turbojet tank volume/first-stage tank volume ratio (R_f)

p_8 = fuselage length (ℓ)

2.3 The Second-Stage Model

The assumption is made that the second stage is a scaled version of the space shuttle, somewhat refined to improve aerodynamics. The thrust is assumed to be selectable. The fuel is assumed to be internally carried. Nominal space shuttle physical properties are given in Table II-4.

Table II-4. Space shuttle physical properties.

Mass (with 65,000 lbm payload)	= 7,000 slugs
Planform area	= 4,000 ft^2
Thrust (in vacuum)	= 1,500,000 lbf
Engine mass	= 600 slugs
Vehicle volume	= 59,000 ft^3

2.3.1 The Physical Characteristics

To compute the planform area of the second stage one uses elementary scaling theory. Since area is a function of linear dimension squared, and the volume is a function of linear dimension cubed one expects a 2/3 power growth factor of planform area vs. volume. One gets

$$A_P = A_{P_0}\left(1 + \left(\frac{V_T}{V_0}\right)\right)^{2/3} \qquad (2\text{-}21)$$

where

A_P = second stage planform area

A_{P_0} = shuttle planform area

V_T = volume required for propellant tank

V_0 = shuttle nominal volume

The space shuttle uses an H_2/O_2 fuel flow ratio 1.6 times stoichiometric. This is because the increase in hydrogen flow beyond a stoichiometric ratio can reduce the average molecular weight of exhaust products more rapidly than the preexpansion temperature. The result is a higher exhaust velocity and thus a higher specific impulse. It seems appropriate to assume the same fuel mixture for the second stage being considered. Also, an 80% use of available volume for propellant storage seems likely after structure and insulation have been accommodated. One gets

$$A_P = A_{P_0}\left(1 + \left(0.0000334\ m_{f_s}\right)\right)^{2/3} \text{ in } ft^2$$

where

m_{f_s} = mass capacity in slugs of fuel tanks for the second stage

Mass property calculations require fuel tank mass contributions and rocket propulsion system mass contributions. On the basis of modest improvements in space shuttle external tank and main engine construction technologies one gets[29]

$$m_{d_s} = m_{d_0} + (0.04\ m_{f_s}) + T_{max_s}/3220. \qquad (2\text{-}22)$$

where

m_{d_s} = second-stage dry mass

m_{d_0} = nominal shuttle dry mass

T_{max_s} = second-stage maximum thrust

(Equation (2-22) is derived from scaling shuttle data with a 15% mass reduction assumed possible compared to shuttle orbiter vehicle 102.

The aerodynamic properties of the second stage are assumed to have characteristics identical to the shuttle, though scaled to a degree assumed possible by significant utilization of new aerodynamic technology (~0.7 for drag; ~1.7 for lift.) The baseline shuttle data is in Table II-5.[30]

Using the data in Table II-5 one has

$$L_2 = C_L A_P q_0 \qquad (2\text{-}23a)$$

$$D_2 = C_D A_P q_0 \qquad (2\text{-}23b)$$

where

L_2 = lift due to second stage

D_2 = drag due to second stage

The thrust level of the rocket propulsion has a maximum value, but may be varied to any nonnegative value less than the maximum. The equation is

$$T_R = T_{max_s}\Phi \qquad (2\text{-}24)$$

Table II-5. Space shuttle aerodynamic data.

C_L

Mach Number

Angle of Attack ↓	0.25	0.60	0.80	0.90	0.95	1.05	1.1	1.2	1.3	1.5	2.0	3.0	4.0	5.0	8.0	10.0
-10°	-0.4974	-0.5239	-0.5734	-0.6084	-0.6373	-0.6388	-0.6283	-0.6034	-0.5590	-0.4784	-0.3735	-0.3011	-0.2390	-0.2073	-0.1860	-0.1800
0	-0.0393	-0.0469	-0.0501	-0.0500	-0.0501	-0.0298	-0.0257	-0.0197	-0.0118	-0.0168	-0.0348	-0.0402	-0.0595	-0.0578	-0.0529	-0.0520
10°	0.4281	0.4383	0.4419	0.4521	0.5123	0.5458	0.5417	0.5236	0.4993	0.4336	0.3281	0.2114	0.1674	0.1399	0.1069	0.1010
20°	1.0001	0.9432	0.8966	0.8770	0.9360	1.0163	0.9958	0.9585	0.9269	0.8433	0.6792	0.5118	0.4505	0.4180	0.3733	0.3621
25°	1.1420	1.0618	0.9138	0.9109	0.9560	1.1198	1.1137	1.0934	1.0748	0.9967	0.8401	0.6580	0.6007	0.5621	0.5174	0.5093

C_D

Mach Number

Angle of Attack ↓	0.25	0.60	0.80	0.90	0.95	1.05	1.1	1.2	1.3	1.5	2.0	3.0	4.0	5.0	8.0	10.0
-10°	0.1047	0.1194	0.1533	0.1877	0.2220	0.2674	0.2703	0.2683	0.2621	0.2477	0.2177	0.1839	0.1628	0.1519	0.1414	0.1400
0	0.0610	0.0625	0.0683	0.0805	0.1052	0.1477	0.1522	0.1557	0.1563	0.1547	0.1410	0.1135	0.0982	0.0884	0.0788	0.0784
10°	0.0891	0.0969	0.1271	0.1538	0.1884	0.2421	0.2448	0.2426	0.2364	0.2149	0.1817	0.1347	0.1125	0.0990	0.0849	0.0837
20°	0.2823	0.3578	0.3870	0.4093	0.4518	0.5227	0.5200	0.5051	0.4867	0.4381	0.3576	0.2751	0.2423	0.2259	0.2023	0.1981
25°	0.4753	0.5074	0.4965	0.5274	0.5706	0.6812	0.6630	0.6673	0.6501	0.5928	0.4976	0.3917	0.3587	0.3373	0.3102	0.3063

where

T_R rocket thrust

It is assumed that rocket specific impulse is constant at 450 seconds for mass flow calculations.

2.3.2 A Summary of Second Stage Parameters

Using the notation adopted for the first-stage independent geometric parameters, one gets the following for the second stage

P_9 = maximum fuel capacity (m_{f_s})

P_{10} = maximum rocket thrust (T_{max_s})

2.3.3 The Environment Model

The physical properties influencing vehicle behavior, but independent of vehicle design constitute the environment. In the problem under consideration, the atmosphere and gravity fit this category. Both are assumed functions only of radial distance from the earth's surface.

Table II-6 lists the data modeling the atmosphere, and used as the basis for computing free stream properties.[31] For Mach number and dynamic pressure calculations, the assumption is made that the air mass rotates with the earth's surface uniformly. Between data points, the density is assumed to exponentially decrease with altitude increase as the basis of interpolation. Temperature is linearly interpolated.

Gravitation is modeled as though the earth were a perfect, homogeneous sphere. That is

$$g = \frac{GM_e}{r^2} \qquad (2\text{-}25)$$

where

G = universal gravitation constant

M_e = mass of the earth

r = distance from earth center

g = gravitation acceleration

Table II-6. Atmospheric properties.

Altitude (km)	$\frac{\rho}{\rho_0}$	Temp (°C)	Altitude (km)	$\frac{\rho}{\rho_0}$	Temp (°C)	Altitude (km)	$\frac{\rho}{\rho_0}$	Temp (°C)
0	1.00000	288.150	68	9.3051×10^{-5}	227.529	136	3.714×10^{-9}	642.32
4	0.66885	262.166	72	5.4361×10^{-5}	211.876	140	2.770×10^{-9}	714.22
8	0.42921	236.215	76	3.050×10^{-5}	196.24	144	2.129×10^{-9}	785.87
12	0.25464	216.650	80	1.632×10^{-5}	180.65	148	1.676×10^{-9}	857.24
16	0.13589	216.650	84	7.807×10^{-6}	180.65	152	1.359×10^{-9}	918.94
20	0.072579	216.650	88	3.738×10^{-6}	180.65	156	1.128×10^{-9}	970.88
24	0.038317	220.560	92	1.584×10^{-6}	182.62	160	9.459×10^{-10}	1022.23
28	0.020470	224.527	96	8.229×10^{-7}	198.45	164	8.139×10^{-10}	1054.67
32	0.011065	228.490	100	4.060×10^{-7}	210.02	168	7.040×10^{-10}	1087.01
36	0.0059248	239.282	104	2.034×10^{-7}	229.18	172	6.149×10^{-10}	1115.73
40	0.0032618	250.350	108	1.080×10^{-7}	247.85	176	5.415×10^{-10}	1136.02
44	0.0018440	261.403	112	5.839×10^{-8}	275.85	180	4.782×10^{-10}	1156.12
48	0.0010749	270.650	116	3.294×10^{-8}	313.01	184	4.236×10^{-10}	1176.03
52	6.5389×10^{-4}	270.650	120	1.988×10^{-8}	349.49	188	3.762×10^{-10}	1195.73
56	4.0622×10^{-4}	263.628	124	1.171×10^{-8}	423.90	192	3.359×10^{-10}	1211.66
60	2.4973×10^{-4}	255.772	128	7.533×10^{-9}	497.36	196	3.014×10^{-10}	1223.88
64	1.5377×10^{-4}	243.202	132	5.165×10^{-9}	570.09	200	2.708×10^{-10}	1235.95

CHAPTER III

SYSTEM DYNAMICS

To complete the dynamics model needed for computer analysis of the air-breathing vehicle, the orbital mechanics model is needed.

The time dependent behavior of the air-breathing launch vehicle can be represented by a set of differential equations. The equations provide the necessary information to evaluate system position, velocity, and mass state provided an appropriate reference frame and a set of initial conditions are defined.

The real problem of the general launch vehicle trajectory analysis requires three components each of position, velocity, and force. However, for the sake of keeping computation requirements to a minimum (due to the strong association of computation time and component dimension), a two dimensional orbital dynamics model is used. Specifically, only an equatorial launch to achieve an equatorial orbit is considered.

If an earth-centered set of polar coordinates is used, Newton's laws of motion yield

$$F_r = m\left(\frac{d^2r}{dt^2} - r\left(\frac{d\theta}{dt}\right)^2\right) \tag{3-1a}$$

$$F_\theta = m\left(r\,\frac{d^2\theta}{dt^2} + 2\,\frac{dr}{dt}\,\frac{d\theta}{dt}\right) \tag{3-1b}$$

where

m = vehicle mass

F_r = force in radial direction

F_θ = force in tangential direction

Also, r and θ are polar coordinates whose sign convention is shown in Figure 3-1.

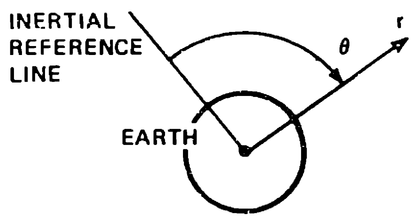


Figure III-1.

Equation (3-1), along with mass flow relations, can be resolved into a set of first-order differential equations in time, constituting a state space representation of the system convenient for future use.

Solving Eq. (3-1) for the second derivatives of r, θ one gets

$$\frac{d^2r}{dt^2} = \frac{F_r}{m} + r\left(\frac{d\theta}{dt}\right)^2 \tag{3-2a}$$

$$\frac{d^2\theta}{dt^2} = \frac{F_\theta}{mr} - \frac{2\,\frac{dr}{dt}\,\frac{d\theta}{dt}}{r} \tag{3-2b}$$

The system mass flow can be separated into three components, one for each propulsive mode. That gives

$$\dot{m} = \sum_i \dot{m}_i \tag{3-3}$$

where

$\dot{m}_i$ = mass flow of i^{th} propulsive mode

i = 1 implies the turbojet

i = 2 imples the scramjet

i = 3 implies the rocket

One also has

$$\frac{d\left(\frac{dr}{dt}\right)}{dt} = \frac{d^2r}{dt^2} \tag{3-4a}$$

$$\frac{d\left(\frac{d\theta}{dt}\right)}{dt} = \frac{d^2\theta}{dt^2} \tag{3-4b}$$

Suppose a vector x is defined as the state where

$$x = \begin{pmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \\ m_1 \\ m_2 \\ m_3 \end{pmatrix} \tag{3-5}$$

Where the dot indicates differentiation with respect to time. One has

$$\dot{x} = \begin{pmatrix} \dot{r} \\ \ddot{r} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{m}_1 \\ \dot{m}_2 \\ \dot{m}_3 \end{pmatrix} \tag{3-6}$$

Substitution of Eq. (3-2) into Eq. (3-6) yields

$$\dot{x} = \begin{pmatrix} \dot{r} \\ r\dot{\theta}^2 + \frac{F_r}{m} \\ \dot{\theta} \\ -\frac{2\dot{r}\dot{\theta}}{r} + \frac{F_\theta}{mr} \\ \dot{m}_1 \\ \dot{m}_2 \\ \dot{m}_3 \end{pmatrix} \tag{3-7}$$

Equation (3-7) is the state space dynamics equation sought. The models given in Chapter II must be used to solve for F_r, F_θ, and $\dot{m}_i$ as a function of states, parameters, and controls.

CHAPTER 4

A SUITABLE OPTIMIZATION ALGORITHM

Suppose one has a set of m ordinary differential equations, each of order n, defining the dynamic state of a physical system as a function of time. The equations can be reduced to a system of m × n first-order differential equations. Such a mathematical system can be put in state space notation by solving each resultant equation algebraically for the first derivative it contains.

Suppose the mathematical representation just described already exists, with the vector x representing the physical system states.

Define

$$\dot{x} = \frac{dx}{dt} = f(x,u,p,t_s) \tag{4-1}$$

where

x = a state vector of dimension a

u = a control vector of dimension b

p = a parameter vector of dimension c

t_s = a switch time vector of dimension d (points of discontinuities in some component of the forces contained in the dynamics)

The function f therefore represents the system dynamical behavior in a form suitable for further manipulation.

In any problem where the objective is to find an extremal value of a performance measure, it is possible to define the performance measure in terms of a function that places a mathematical cost on the system behavior. When the extremal value of the mathematical cost function is found, the extremal performance is achieved. The mathematical function can, in one form, be represented by two terms. One is a terminal time cost term, the other is a distributed cost term.

Define

$$J(\tau) = \phi(x(\tau)) + \int_{\tau}^{0} L(x(t),u(t),p(t))dt \qquad (4\text{-}2)$$

where

$J(\tau)$ = the mathematical cost imposed on the system at time τ

$\phi(x(\tau))$ = a terminal cost term evaluated at τ, dependent only on state

$L(x(t),u(t),p(t))$ = the distribution function for the distributed cost term, dependent on the states, controls, and parameters

τ = the terminal time. Since τ will have a negative value later, it is given as the integral lower bound.

By the above formulation, ϕ and L have no explicit time dependence.

From the present formulation, one can create a Hamiltonian function appropriate to finding an extremal, of the the cost.

Define

$$H = L + \lambda^T f \qquad (4\text{-}3)$$

where

H = a Hamiltonian function based on the cost

λ = a vector known as the costate variable whose differential equation will be defined later.

The intent is to find a minimum value of J. It seems likely that J will be a smooth function of the control variables and design parameters, justifying a search for optimal solutions at stationery points only. Thus, a desired set of states, controls, parameters, and switch times are found when $dJ = 0$.

To start, differentiate Eq. (4-2). One gets

$$dJ = \phi_x dx(\tau + d\tau) + \int_{\tau}^{0} (L_x \delta x + L_u \delta u + L_p \delta p)dt - L\Big|_{\tau} d\tau \qquad (4\text{-}4)$$

where

$\phi_x = \frac{\partial \phi}{\partial x}$, a row vector of dimension a

$L_x = \frac{\partial L}{\partial x}$, a row vector of dimension a

$L_u = \frac{\partial L}{\partial u}$, a row vector of dimension b

$L_p = \frac{\partial L}{\partial p}$, a row vector of dimension c

δx = a variation of x, a vector of dimension a

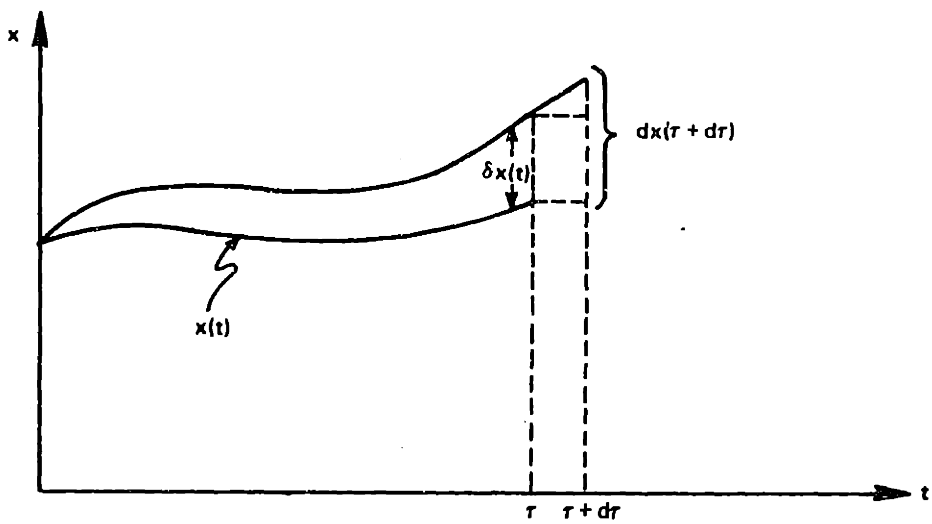
δu = a variation of u, a vector of dimension b

δp = a variation of p, a vector of dimension c

One can geometrically demonstrate that to first order

$$dx(\tau + d\tau) = \delta x(\tau) + f(\tau)d\tau \qquad (4\text{-}5)$$

(See Figure IV-1 for a scalar function representation.)



$dx(\tau + d\tau) = \delta x(\tau) + \frac{dx}{dt}(\tau)\, d\tau$ + TERMS HIGHER THAN FIRST ORDER

$dx(\tau + d\tau) = \delta x(\tau) + f(\tau)\, d\tau$ TO FIRST ORDER

Figure IV-1.

The first order approximation for dx is used since the algorithm to be developed will itself be first order. Specifically, gradient information will be used. If all information used is valid to first order then no information necessary to the algorithm is lost.

Equations (4-4) and (4-5) are combined to yield

$$dJ = \phi_x \delta x + (\phi_x f - L)|_\tau \, d\tau + \int_\tau^0 (L_x \delta x + L_u \delta u + L_p \delta p) dt \qquad (4\text{-}6)$$

It is necessary to find a relation between δx and $d\tau$ to eliminate the $d\tau$ term in Eq. (4-6).

The solution of the optimization problem requires integration of the states x starting at time = 0 until time = τ. The termination of the state integration will have to be determined by some cutoff function at $t = \tau$ since τ is a free quantity not known a priori. Suppose $\Omega(x(\tau))$ is such a cutoff conditon.

Let

$$\Omega(x(\tau)) = 0 \qquad (4\text{-}7)$$

One gets

$$\Omega(x(\tau) + dx(\tau + d\tau)) = 0 \qquad (4\text{-}8)$$

Using an expansion series one gets

$$\Omega(x(\tau) + dx(\tau + d\tau)) = \Omega(x(\tau)) + \Omega_x dx(\tau + d\tau) \qquad (4\text{-}9)$$

to first order

Use of Eqs. (4-7) and (4-8) in Eq. (4-9) yields

$$\Omega_x dx(\tau + d\tau) = 0 \qquad (4\text{-}10)$$

Use of Eq. (4-5) in Eq. (4-10) yields

$$\Omega_x (\delta x(\tau) + f(\tau) d\tau) = 0 \qquad (4\text{-}11)$$

Equation (4-11) can be algebraically solved for $d\tau$ to get

$$d\tau = -\frac{\Omega_x \delta x(\tau)}{\Omega_x f} \tag{4-12}$$

Substitution of Eq. (4-12) into Eq. (4-6) yields

$$dJ = \left(\phi_x - \frac{\left(\phi_x f - L\right)}{\Omega_x f}\Omega_x\right)\Bigg|_\tau \delta x(\tau) + \int_\tau^0 \left(L_x \delta x + L_u \delta u + L_p \delta p\right) dt \tag{4-13}$$

Two point boundary value problems are usually formulated by defining an adjoint function or sensitivity function which obeys a differential equation of adjoint form. Applying this approach in this case, one defines

$$\dot{\lambda} = \frac{d\lambda}{dt} = -F^T\lambda - L_x^T \tag{4-14}$$

where

$$F = \frac{\partial f}{\partial x} \text{ a matrix of dimension } a \times a \tag{4-15}$$

One has

$$\frac{d}{dt}(\lambda^T \delta x) = \dot{\lambda}^T \delta x + \lambda^T \delta \dot{x} \tag{4-16}$$

Use of Eq. (4-16) requires that a representation of $\delta\dot{x}$ be developed, but $\dot{x}$ can change discontinuously due to the switch points at t_s. This can be treated by considering the influence on $\dot{x}$ of each of the switch points separately.

Define

$$\delta x = \delta x_0 + \sum_i \delta x_i$$

yielding

$$\delta\dot{x} = \delta\dot{x}_0 + \sum_i \delta\dot{x}_i \tag{4-17}$$

where

x_0 = state value without switch point contributions

x_i = state contribution due to switch point i

One has

$$\delta \dot{x}_0 = \frac{\partial \dot{x}_0}{\partial x} \delta x_0 + \frac{\partial \dot{x}_0}{\partial u} \delta u + \frac{\partial \dot{x}_0}{\partial p} \delta p \qquad (4\text{-}18a)$$

$$\delta \dot{x}_i = \frac{\partial \dot{x}_i}{\partial x} \delta x_i + \frac{\partial \dot{x}_i}{\partial u} \delta u + \frac{\partial \dot{x}_i}{\partial p} \delta p \qquad (4\text{-}18b)$$

One notices, however, that x_i was chosen to represent the effect on x of the switch point i. The switch point does not influence the behavior of x in response to u or p. One, therefore, has in Eq. (4-18b)

$$\delta u = \delta p = 0 \qquad (4\text{-}19)$$

It is convenient to define two new variables

$$G = \frac{\partial \dot{x}}{\partial \mu} \qquad (4\text{-}20a)$$

$$K = \frac{\partial \dot{x}}{\partial p} \qquad (4\text{-}20b)$$

Substitution of Eqs. (4-20), (4-19), and (4-15) into Eq. (4-18) yields

$$\delta \dot{x}_0 = F\delta x_0 + G\delta u + K\delta p \qquad (4\text{-}21a)$$

$$\delta \dot{x}_i = F\delta x_i \qquad (4\text{-}21b)$$

Substitution of Eqs. (4-21) and (4-14) into Eq. (4-16) yields

$$\frac{d}{dt}\left(\lambda^T \delta x\right) = \left(-\lambda^T F - L_x\right)\delta x + \lambda^T\left(F\delta x_0 + \sum_i F\delta x_i + G\delta u + K\delta p\right) \qquad (4\text{-}22)$$

Simplification of Eq. (4-22) and use of Eq. (4-17) gives

$$\frac{d}{dt}\left(\lambda^T \delta x\right) = -L_x \delta x + \lambda^T (G\delta u + K\delta p) \tag{4-23}$$

One has

$$\delta x(0) = 0 \tag{4-24}$$

since the desired state at t = 0 is specified explicitly.

Combination of Eqs. (4-24) and (4-17) yields

$$\delta x_0(0) = \delta x_i(0) = 0 \tag{4-25}$$

Integration of Eq. (4-23) from $t = \tau$ to $t = 0$ can now be done, using Eq. (4-25) and the fact that $\delta x_i = 0$ for $\tau \leq t < t_{s_i}$, or in other words the switch points contribute nothing until they occur, permitting integration of their contribution from the times of the switch points to the end only. (Note: the inequality is based on $\tau < 0$.)

One gets

$$-\lambda^T(\tau)\delta x_0(\tau) - \sum_i \lambda^T\left(t_{s_i}\right)\delta x_i\left(t_{s_i}\right) = \int_\tau^0 \left(-L_x \delta x + \lambda^T(G\delta u + K\delta p)\right)dt \tag{4-26}$$

By geometric arguments one can show

$$\delta x_i\left(t_{s_i}\right) = -\left(f^+_{s_i} - f^-_{s_i}\right)dt_{s_i} \tag{4-27}$$

where

$$f^+_s = \dot{x}(t^+_s) \tag{4-28a}$$

$$f^-_s = \dot{x}(t^-_s) \tag{4-28b}$$

(See Figure IV-2 for a scalar function representation.)

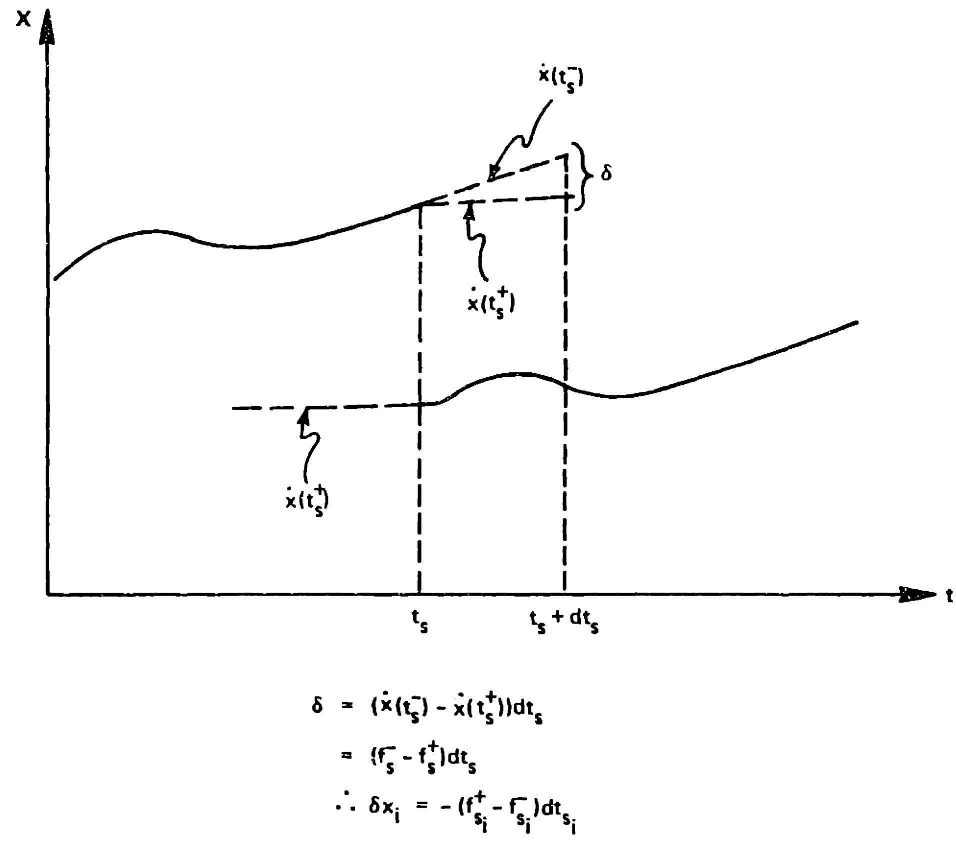


Figure IV-2.

Use of Eq. (4-28) in Eq. (4-26) yields

$$-\lambda^T(\tau)\delta x_0(\tau) = \int_\tau^0 (-L_x\delta x + \lambda^T G\delta u + \lambda^T K\delta p)dt$$

$$- \sum_i \lambda^T\left(t_{s_i}\right)\left(f_{s_i}^+ - f_{s_i}^-\right)dt_{s_i} \qquad (4\text{-}29)$$

However, since the transition points have no effect at $t = \tau$, one has

$$\delta x_0(\tau) = \delta x(\tau) \qquad (4\text{-}30)$$

Substitution of Eq. (4-30) into Eq. (4-29) yields

$$-\lambda^T(\tau)\delta x(\tau) = \int_\tau^0 (-L_x\delta x + \lambda^T G\delta u + \lambda^T K\delta p)\,dt - \sum_i \lambda^T\left(t_{s_i}\right)\left(f^+_{s_i} - f^-_{s_i}\right)dt_{s_i} \quad (4\text{-}31)$$

Comparison of Eq. (4-31) and Eq. (4-13) implies an appropriate definition of $\lambda(\tau)$.

Define

$$\lambda^T(\tau) = -\left(\phi_x - \frac{\phi_x f - L}{\Omega_x f}\,\Omega_x\right)\Bigg|_\tau \quad (4\text{-}32)$$

Equation (4-32) is substituted into Eq. (4-31) and the result is substituted into Eq. (4-13) to obtain

$$dJ = \int_\tau^0 \left(L_u\delta u + L_p\delta p + \lambda^T G\delta u + \lambda^T K\delta p\right)dt - \sum_i \lambda^T\left(t_{s_i}\right)\left(f^+_{s_i} - f^-_{s_i}\right)dt_{s_i} \quad (4\text{-}33)$$

One has

$$H_u = L_u + \lambda^T G \quad (4\text{-}34a)$$

$$H_p = L_p + \lambda^T K \quad (4\text{-}34b)$$

Substitution of Eq. (4-34) into Eq. (4-33) yields

$$dJ = \int_\tau^0 H_u\delta u\,dt + \int_\tau^0 H_p\delta p\,dt - \sum_i \lambda^T\left(t_{s_i}\right)\left(f^+_{s_i} - f^-_{s_i}\right)dt_{s_i} \quad (4\text{-}35)$$

Note that p defines geometric parameters, so it must be time invariant. One has

$$\frac{dp}{dt} = 0 \tag{4-36}$$

Use of Eq. (4-36) in Eq. (4-35) yields

$$dJ = \int_{\tau}^{0} H_u \delta u dt + \left(\int_{\tau}^{0} H_p dt \right) \delta p - \sum_i \lambda^T \left(t_{s_i} \right) \left(f_{s_i}^{+} - f_{s_i}^{-} \right) dt_{s_i} \tag{4-37}$$

Up to this point, the problem has assumed a specified state at $t = 0$, and a free state at terminal time. Typically, however, the terminal time state is constrained in some manner that may include a dependence on the parameters p. This is treated by creating a set of terminal state constraint functions, and a set of constraint influence functions, that establish a gradient contribution in the desired control parameter, and transition time variations as a function of the violations of the constraints.

Suppose one has several relations defining some equality constraints on state and parameters at terminal time in addition to Ω in Eq. (4-7). Call the constraint vector Ψ.

One has

$$\Psi = \Psi(x(\tau), p(\tau)) = \Psi(x(\tau), p) = 0 \tag{4-38}$$

where

Ψ = a vector of dimension k

Differentiation of Eq. (4-38) yields

$$d\Psi = \Psi_x dx(\tau + d\tau) + \Psi_p dp(\tau + d\tau) \tag{4-39}$$

In analogy to Eq. (4-5) one gets

$$dp(\tau + d\tau) = \delta p(\tau) + \frac{dp}{dt}(\tau) d\tau \tag{4-40}$$

Substitution of Eq. (4-36) into Eq. (4-40) and subsequent use of the result and Eq. (4-5) in Eq. (4-39) yields

$$d\Psi = \Psi_x(\delta x(\tau) + f(\tau)d\tau) + \Psi_p \delta p \tag{4-41}$$

It is necessary to create a set of differential equations that will incorporate the results of Eq. (4-41), and will propagate the influence of the equality constraints through the vehicle trajectory and geometry. To achieve this, one must define a matrix function Λ of dimension $(a + c) \times k$.

One has

$$\frac{d}{dt}\left(\Lambda^T \begin{pmatrix}\delta x \\ \delta p\end{pmatrix}\right) = \dot{\Lambda}^T \begin{pmatrix}\delta x \\ \delta p\end{pmatrix} + \Lambda^T \begin{pmatrix}\delta \dot{x} \\ \delta \dot{p}\end{pmatrix} \tag{4-42}$$

Use of Eq. (4-36) in Eq. (4-42) yields

$$\frac{d}{dt}\left(\Lambda^T \begin{pmatrix}\delta x \\ \delta p\end{pmatrix}\right) = \dot{\Lambda}^T \begin{pmatrix}\delta x \\ \delta p\end{pmatrix} + \Lambda^T \begin{pmatrix}\delta \dot{x} \\ 0\end{pmatrix} \tag{4-43}$$

As with the influence function for the cost variation, an influence function satisfying the adjoint differential equation can be used to express the effect of control variations on the constraint functions.

$$\dot{\Lambda} = -A^T \Lambda \tag{4-44}$$

where

$$A = \begin{pmatrix} f_x & f_p \\ \dot{p}_x & \dot{p}_p \end{pmatrix} \tag{4-45}$$

The matrix A definition is selected to separate state and parameter equality constraint sensitivity effects.

$$A = \begin{pmatrix} f_x & f_p \\ 0 & 0 \end{pmatrix} \tag{4-46}$$

Substitution of Eqs. (4-44) and (4-21) into Eq. (4-43) yields

$$\frac{d}{dt}\left(\Lambda^T \begin{pmatrix}\delta x \\ \delta p\end{pmatrix}\right) = -\Lambda^T A \begin{pmatrix}\delta x \\ \delta p\end{pmatrix} + \Lambda^T \begin{pmatrix}F\delta x + G\delta u + K\delta p \\ 0\end{pmatrix} \tag{4-47}$$

Substitution of Eq. (4-46) into Eq. (4-47) and simplification yields

$$\frac{d}{dt}\left(\Lambda^T\begin{pmatrix}\delta x\\ \delta p\end{pmatrix}\right) = \Lambda^T\begin{pmatrix}G\delta u\\ 0\end{pmatrix} \tag{4-48}$$

Integration of Eq. (4-48), use of Eq. (4-25) and the fact that the influence of the switch at t_{s_i} is zero for $(\tau \leq t < t_{s_i})$ if $\tau < 0$ and the integration is from $t = \tau$ to $t = 0$ yields

$$\Lambda^T(0)\begin{pmatrix}0\\ \delta p\end{pmatrix} - \Lambda^T(\tau)\begin{pmatrix}\delta x(\tau)\\ \delta p\end{pmatrix} - \sum_i \Lambda^T\left(t_{s_i}\right)\begin{pmatrix}\delta x_i\left(t_{s_i}\right)\\ 0\end{pmatrix} = \int_\tau^0 \Lambda^T\begin{pmatrix}G\delta u\\ 0\end{pmatrix}dt \tag{4-49}$$

where the t_{s_i} terms are derived in a manner similar to Eq. (4-26).

Substitution of Eq. (4-27) into Eq. (4-49) yields

$$\Lambda^T(0)\begin{pmatrix}0\\ \delta p\end{pmatrix} - \Lambda^T(\tau)\begin{pmatrix}\delta x(\tau)\\ \delta p\end{pmatrix} = \int_\tau^0 \Lambda^T\begin{pmatrix}G\delta u\\ 0\end{pmatrix}dt - \sum_i \Lambda^T\left(t_{s_i}\right)\begin{pmatrix}\left(f_{s_i}^+ - f_{s_i}^-\right)dt_{s_i}\\ 0\end{pmatrix} \tag{4-50}$$

Suppose, for convenience, Λ is partitioned into two separate matrices.

Define

$$\Lambda = \begin{pmatrix}\Lambda_1\\ \Lambda_2\end{pmatrix} \tag{4-51}$$

where

Λ_1 = a matrix of dimension $a \times k$

Λ_2 = a matrix of dimension $c \times k$

Substitution of Eq. (4-51) into Eq. (4-50) yields

$$-\Lambda_1^T(\tau)\delta x(\tau) + (\Lambda_2^T(0) - \Lambda_2^T(\tau))\delta p = \int_\tau^0 \Lambda_1^T G\delta u dt - \sum_i \Lambda_1^T\left(t_{s_i}\right)\left(f_{s_i}^+ - f_{s_i}^-\right)dt_{s_i} \tag{4-52}$$

Substitution of Eq. (4-12) into Eq. (4-41) yields

$$d\Psi = \left(\Psi_x - \frac{\Psi_x f \Omega_x}{\Omega_x f}\right)\delta x + \Psi_p \delta p \tag{4-53}$$

Comparing the forms of Eq. (4-52) with (4-53), it is clear that useful definitions of the boundary conditions are

$$\Lambda_1^T(\tau) = -\left(\Psi_x - \frac{\Psi_x f \Omega_x}{\Omega_x f}\right) \tag{4-54a}$$

$$\Lambda_2^T(\tau) = -\Psi_p \tag{4-54b}$$

Substitution of Eq. (4-54) into Eq. (4-52) and subsequently the result into Eq. (4-53) yields

$$d\Psi = \int_\tau^0 \Lambda_1^T G \delta u dt - \Lambda_2^T(0)\delta p - \sum_i \Lambda_1^T\left(t_{s_i}\right)\left(f_{s_i}^+ - f_{s_i}^-\right)dt_{s_i} \tag{4-55}$$

A certain amount of notational simplicity is useful for future manipulation. The following definitions are therefore useful.

Define

$$g = \begin{pmatrix} \int_\tau^0 H_p^T dt \\ -\lambda^T\left(t_{s_1}\right)\left(f_{s_1}^+ - f_{s_1}^-\right) \\ -\lambda^T\left(t_{s_2}\right)\left(f_{s_2}^+ - f_{s_2}^-\right) \\ \vdots \end{pmatrix} \tag{4-56a}$$

$$\delta v = \begin{pmatrix} \delta p \\ dt_{s_1} \\ dt_{s_2} \\ \vdots \end{pmatrix} \tag{4-56b}$$

$$M = \left[-\Lambda_2^T(0) \;\middle|\; -\Lambda_1^T\left(t_{s_1}\right)\left(f_{s_1}^+ - f_{s_1}^-\right) \;\middle|\; -\Lambda_1^T\left(t_{s_2}\right)\left(f_{s_2}^+ - f_{s_2}^-\right) \middle| - - - \right] \tag{4-56c}$$

Substitution of Eq. (4-56) into Eq. (4-55) and (4-37) yields

$$dJ = \int_{\tau}^{0} H_u \delta u dt + g^T \delta v \tag{4-57a}$$

$$d\Psi = \int_{\tau}^{0} \Lambda_1^T G \delta u dt + M \delta v \tag{4-57b}$$

Inspection of Eq. (4-57) reveals that simplification of the desired decreases in J and $|\Psi|$ functionally imply desired variations in δu and δv. It is necessary to control the size of the steps taken by u and v, however, to assure that the first order approximations made thus far do not prevent convergence to the extremal solution one seeks.

An appropriate way to define step size is to create a quantity which is quadratic in δu and δv to assure a step whose measure is positive.

Define

$$S = \text{step size}$$

$$S^2 = \frac{1}{2} \int_{\tau}^{0} \delta u^T U^{-1} \delta u dt + \frac{1}{2} \delta v^T V^{-1} \delta v \tag{4-58}$$

where

V = a symmetric positive definite matrix to weight relative δv variations per iteration (dimension $[c+d] \times [c+d]$)

U(t) = a symmetric time function matrix, positive definite at all times, to weight δu variations per iteration (dimension $b \times b$)

It is appropriate to seek improvement in the equality constraint violations in direct proportion to the violations.

Define

$$d\Psi = -C_\psi \Psi \tag{4-59}$$

where

C_ψ is a constant to be specified

The step size control, the constraint violation improvement, and the cost improvement relations must be unified into a single set of relations. This is accomplished by adjoining a combination of Eqs. (4-59), (4-58), and (4-57) with Lagrange multipliers.

One defines η, ν as Lagrange multipliers. Substition of Eq. (4-59) into Eq. (4-57b), and combining the result with Eqs. (4-57a) and (4-58) yields

$$d\bar{J} = \int_\tau^0 H_u \delta u dt + g^T \delta v + \eta \left(\frac{1}{2} \int_\tau^0 \delta u^T U^{-1} \delta u dt + \frac{1}{2} \delta v^T V^{-1} \delta v - S^2 \right)$$

$$+ \nu^T \int_\tau^0 \left(\Lambda_1^T G \delta u dt + M \delta v + C_\psi \Psi \right) \tag{4-60}$$

At the extremal, the variation of $d\bar{J}$ with respect to δu and δv will vanish.

One gets

$$\delta d\bar{J} = 0 = \int_\tau^0 \left(H_u + \eta \delta u^T U^{-1} + \nu^T \Lambda_1^T G \right) \delta \delta u dt$$

$$+ \left(g^T + \eta \delta v^T V^{-1} + \nu^T M \right) \delta \delta v \tag{4-61}$$

Since the variations $\delta\delta u$ and $\delta\delta v$ are independent, the $\delta\delta u$ term and the $\delta\delta v$ term must separately vanish.

One gets

$$H_u + \eta \delta u^T U^{-1} + \nu^T \Lambda_1^T G = 0 \tag{4-62a}$$

$$g^T + \eta \delta v^T V^{-1} + \nu^T M = 0 \tag{4-62b}$$

One can solve Eq. (4-62) for δu, δv explicitly, yielding

$$\delta u = -\frac{1}{\eta} U(H_u^T + G^T \Lambda_1 \nu) \tag{4-63a}$$

$$\delta v = -\frac{1}{\eta} V(g + M^T \nu) \tag{4-63b}$$

Future notational simplicity suggests the following definitions

$$C_J = \frac{1}{\eta} \tag{4-64a}$$

$$z = -\frac{1}{\eta} \nu \tag{4-64b}$$

The following results are obtained upon substitution of Eq. (4-64) into Eq. (4-63)

$$\delta u = U(-C_J H_u^T + G^T \Lambda_1 z) \tag{4-65a}$$

$$\delta v = -C_J V g + V M^T z \tag{4-65b}$$

Substitution of Eq. (4-65) into Eq. (4-57b), and use of Eq. (4-59) yields

$$-C_\psi \Psi = \int_\tau^0 \Lambda_1^T G U(-C_J H_u^T + G^T \Lambda_1 z) dt + M(-C_J V g + V M^T z) \tag{4-66}$$

Some algebraic manipulation of Eq. (4-66) yields

$$\int_\tau^0 \Lambda_1^T G U G^T \Lambda_1 dt \; z + M V M^T z = -C_\psi \Psi + C_J \int_\tau^0 \Lambda_1^T G U H_u^T dt + C_J M V g \tag{4-67}$$

Further notational simplification suggests the definitions

$$I_{\psi\psi} = \int_\tau^0 \Lambda_1^T G U G^T \Lambda_1 dt \tag{4-68a}$$

$$I_{\Psi J} = \int_{\tau}^{0} \Lambda_1^T G U H_u^T dt \tag{4-68b}$$

Substitution of Eq. (4-68) into Eq. (4-67) yields

$$(I_{\Psi\Psi} + MVM^T)z = -C_{\Psi}\Psi + C_J\ (I_{\Psi J} + MVg) \tag{4-69}$$

It is convenient to define

$$B = (I_{\Psi\Psi} + MVM^T)^{-1} \tag{4-70a}$$

$$C = (I_{\Psi J} + MVg) \tag{4-70b}$$

Substitution of Eq. (4-70) into Eq. (4-69) and solution for z yields

$$z = B(-C_{\Psi}\Psi + C_J C) \tag{4-71}$$

Substitution of Eq. (4-71) into Eq. (4-65) yields

$$\delta u = U(-C_J H_u^T + G^T \Lambda_1 B(-C_{\Psi}\Psi + C_J C)) \tag{4-72a}$$

$$\delta v = -C_J Vg + VM^T B(-C_{\Psi}\Psi + C_J C) \tag{4-72b}$$

Reordering of terms in Eq. (4-72) leads to

$$\delta u = U(-C_{\Psi} G^T \Lambda_1 B\Psi - C_J(H_u^T - G^T \Lambda_1 BC)) \tag{4-73a}$$

$$\delta v = -C_{\Psi} VM^T B\Psi - C_J(Vg - VM^T BC) \tag{4-73b}$$

An equation defining the value of C_J is necessary. Since it is also necessary to control the magnitude of the desired improvement in cost on any iteration in order to assure the validity of the first order approximations already used, it is appropriate to mathematically tie the value of C_J to the gradient equations and some suitable cost variation (a measure of step size). One therefore defines a specified cost variation dJ_s.

Using Eq. (4-73) and (4-57a) one gets

$$dJ_s = \int_\tau^0 H_u U(-C_\Psi G^T \Lambda_1 B\Psi - C_J (H_u^T - G^T \Lambda_1 BC))dt$$

$$+ g^T(-C_\Psi VM^T B\Psi - C_J(Vg - VM^T BC)) \quad (4\text{-}74)$$

Reordering terms in Eq. (4-74) yields

$$dJ_s = -C_\Psi\left(\int_\tau^0 H_u UG^T \Lambda_1 B\Psi dt + g^T VM^T B\Psi\right)$$

$$-C_J\left(\int_\tau^0 (H_u UH_u^T - H_u UG^T \Lambda_1 BC)dt + g^T(Vg - VM^T BC)\right) \quad (4\text{-}75)$$

It is convenient to define a new function

$$I_{JJ} = \int_\tau^0 H_u UH_u^T dt \quad (4\text{-}76)$$

Substitution of Eqs. (4-76) and (4-68b) into Eq. (4-75) yields

$$dJ_s = -C_\Psi\left(I_{\Psi J}^T B\Psi + g^T VM^T B\Psi\right) - C_J\left(I_{JJ} - I_{\Psi J}^T BC + g^T Vg - g^T VM^T BC\right) \quad (4\text{-}77)$$

Solving Eq. (4-77) for C_J one gets

$$C_J = -\frac{dJ_s + C_\Psi\left(I_{\Psi J}^T B\Psi + g^T VM^T B\Psi\right)}{I_{JJ} - I_{\Psi J}^T BC + g^T Vg - g^T VM^T BC} \quad (4\text{-}78)$$

With dJ_s somehow specified, use of Eqs. (4-78) and (4-73) permits one to evaluate δu and δv providing C_Ψ is computable. C_Ψ, however, is simply a value used to establish a rate at which the Ψ vector violations are reduced. Starting with a small positive value ($0 < C_\psi \leq 0.2$) when the violations of Ψ are large, and increasing C_ψ towards 1.0 as the violations diminish, permits stable convergence and suits the linear approximations used throughout.

CHAPTER V

SPECIFIC FUNCTIONAL, DERIVATIVE, AND BOUNDARY CONDITION RELATIONS

Use of the optimization algorithm outlined in Chapter IV requires definition of the cost function terms, choice of initial state boundary conditions, specification of equality constraints and the state integration cutoff conditions, computation of derivative terms, and evaluation of boundary conditions on adjoined functions.

5.1 The Cost Function

Equation (4-2) specifies a mathematical cost function to define the measure of optimal system behavior. The cost consists of two components, one evaluated at the terminal state condition, the other distributed over the entire trajectory.

Three quantities: fuel consumption, dynamic pressure, and specific force, are chosen as a minimum set of elements required to characterize optimal system performance. Fuel consumption is a quantity which ought to be minimized. Dynamic pressure and specific force (the acceleration exclusive of gravity) are quantities which ought to be constrained to fall within bounds determined by structural failure loads, and payload or crew load tolerances.

The terminal state cost function term may be viewed as the quantity to be minimized, which is total fuel mass consumed in the example under study.

This leads to the following definition

$$\phi(x(\tau)) = x_5(\tau) + x_6(\tau) + x_7(\tau) = m_1(\tau) + m_2(\tau) + m_3(\tau) \tag{5-1}$$

Thus, the terminal cost element is the sum of fuel used for all propulsive modes combined.

The distributed cost function term may be viewed as a collection of penalty terms to increase resultant system cost for the violation of desired bounds on inequality constrained quantities. Dynamic pressure and specific force can be penalized in this manner when they exceed design limits.

This gives

$$L(x(t), u(t), p) = L_1(x(t)) + L_2(x(t), u(t), p) \tag{5-2}$$

where

L_1 = penalty term on dynamic pressure

L_2 = penalty term on specific force

A proper choice of L_1 and L_2 requires that no penalty be assessed when the dynamic pressure and specific force remain within desired bounds, that penalties be assessed at increasing rates as the bound violations increase, and that no discontinuities in the penalty term should exist at the bounds. A reasonable choice of penalty terms is

$$L_1 = Q(q - q_d)^2 u_0(q - q_d) \tag{5-3a}$$

$$L_2 = C_A(F_{sp} - a_d)^2 u_0(F_{sp} - a_d) \tag{5-3b}$$

where

Q = a constant to be chosen that weights dynamic pressure bound violation penalties

C_A = a constant to be chosen that weights specific force bound violation penalties

q = dynamic pressure

q_d = desired dynamic pressure bound

F_{sp} = specific force

a_d = desired acceleration bound (excluding gravity)

$u_0(q - q_d)$ = a unit step function with the step at $q = q_d$

$u_0(F_{sp} - a_d)$ = a unit step function with the step at $F_{sp} = a_d$

Note that

$$q = \rho\left(\frac{\dot{r}^2 + r^2\left(\dot{\theta} - \omega_e\right)^2}{2}\right) = \rho\left(\frac{x_2^2 + x_1^2\left(x_4 - \omega_e\right)^2}{2}\right) \quad (5\text{-}4a)$$

$$F_{sp} = \frac{\sqrt{\left(F_r - F_g\right)^2 + F_\theta^2}}{m} \quad (5\text{-}4b)$$

where

$\rho = \rho(r) = \rho(x_1)$ = atmospheric density model, assumed a function of altitude only

F_r = net radial force on the vehicle

F_g = net gravitational force on the vehicle

F_θ = net tangential force on the vehicle

m = net vehicle mass

Substitution of Eqs. (5-3) and (5-2) into Eq. (4-2) yields

$$J(\tau) = m_1(\tau) + m_2(\tau) + m_3(\tau) + \int_\tau^0 [Q(q - q_d)^2 u_0(q - q_d)$$

$$+ C_A(F_{sp} - a_d)^2 u_0(F_{sp} - a_d)]dt \quad (5\text{-}5)$$

5.2 The Initial State Boundary Condition

The initial state is specified as the desired orbital state of the launch vehicle. (With the terminal state locating the launch conditions). The primary function of any vehicle in the space shuttle class is to deliver payloads to low earth orbit, so a representative low earth orbit is chosen as a basis of evaluation of the air-breathing vehicle performance. The choice of a seventy-five mile circular orbit seems simplest and appropriate since it represents the fringe of the sensible atmosphere.

The use of a circular reference orbit makes the desired radial velocity vanish, and the desired angular velocity independent of position. The angular position can be chosen on the basis of a convenient reference value. One gets

$$x(0) = \begin{pmatrix} r(0) \\ \dot{r}(0) \\ \theta(0) \\ \dot{\theta}(0) \\ m_1(0) \\ m_2(0) \\ m_3(0) \end{pmatrix} = \begin{pmatrix} R_e + h_0 \\ 0 \\ 0 \\ \omega_0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad (5-6)$$

where

R_e = earth radius

h_0 = orbital altitude

ω_0 = orbital angular rate

The propellant should be totally consumed at orbit insertion leading to the zero value for propellant mass at t = 0. Since angular position is an arbitrary reference, it is chosen to be zero.

Elementary orbital mechanics yield[32]

$$\omega_0 = \sqrt{\frac{GM_e}{r^3}} \qquad (5-7)$$

making $\dot{\theta}(0)$ an explicit function of specified orbit altitude.

5.3 The Equality Constraints and the State Integration Cutoff Condition

The equality constraints were formulated as a method of handling the desired terminal state conditions on vehicle parameters, desired position and velocity, and propellant quantity.

Of the four position and velocity states only three require constraint since the angular position can be tied to an arbitrary initial state angular position reference. This leaves a requirement for constraint of radial position as well as radial and tangential velocity.

The optimization algorithm allows specification of parameter-dependent constraints. Since it is desired to match propellant tank capacity with propellant requirements, a set of propellant mass constraints are derived, one for each propellant type.

Equation (4-7) specifies a function required for state integration cutoff. (The function locates the terminal time.) The cutoff function must come from the list of possible equality constraint functions, and should have an unambiguous zero crossing to assure proper choice of cutoff time.

The most promising candidate for the cutoff function is angular velocity since the entire desired velocity component for a horizontal takeoff would be tangential, and is likely to be a well defined minimum value. This defines

$$\Omega(x(\tau)) = x_4(\tau) - \dot{\theta}_d = \dot{\theta} - \dot{\theta}_d \qquad (5\text{-}8)$$

where

$\dot{\theta}_d$ = desired angular velocity at takeoff

Equation (5-8) eliminates one of the equality constraints leaving

$$\Psi = 0 = \begin{pmatrix} r - R_e \\ \dot{r} \\ m_1 - M_1 \\ m_2 - M_2 \\ m_3 - M_3 \end{pmatrix} = \begin{pmatrix} x_1 - R_e \\ x_2 \\ x_5 - M_1 \\ x_6 - M_2 \\ x_7 - M_3 \end{pmatrix} \qquad (5\text{-}9)$$

where

M_1 = turbojet propellant tank mass capacity

M_2 = scramjet propellant tank mass capacity

M_3 = rocket propellant tank mass capacity

The constraints imply takeoff from the earth's surface, with horizontal velocity only at takeoff, and with no excess tank capacity.

5.4 The Derivative Terms

Two partial derivative matrices require evaluation at each integration step in the implementation of the optimization algorithm. These are F and K defined in Eqs. (4-15) and (4-20b).

Differentiation of Eq. (3-7) yields

$$
F = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\left(\dot{\theta}^2 + \frac{1}{m}\frac{\partial F_r}{\partial r}\right) & \frac{1}{m}\frac{\partial F_r}{\partial \dot{r}} & 0 & \left(2r\dot{\theta} + \frac{1}{m}\frac{\partial F_r}{\partial \dot{\theta}}\right) & \left(-\frac{F_r}{m^2} + \frac{1}{m}\frac{\partial F_r}{\partial m_1}\right) & \left(-\frac{F_r}{m^2} + \frac{1}{m}\frac{\partial F_r}{\partial m_2}\right) & \left(-\frac{F_r}{m^2} + \frac{1}{m}\frac{\partial F_r}{\partial m_3}\right) \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\left(\frac{2\dot{r}\dot{\theta}}{r^2} - \frac{F_\theta}{mr^2} + \frac{1}{mr}\frac{\partial F_\theta}{\partial r}\right) & \left(-\frac{2\dot{\theta}}{r} + \frac{1}{mr}\frac{\partial F_\theta}{\partial \dot{r}}\right) & 0 & \left(-\frac{2\dot{r}}{r} + \frac{1}{mr}\frac{\partial F_\theta}{\partial \dot{\theta}}\right) & \left(-\frac{F_\theta}{m^2 r} + \frac{1}{mr}\frac{\partial F_\theta}{\partial m_1}\right) & \left(-\frac{F_\theta}{m^2 r} + \frac{1}{mr}\frac{\partial F_\theta}{\partial m_2}\right) & \left(-\frac{F_\theta}{m^2 r} + \frac{1}{mr}\frac{\partial F_\theta}{\partial m_3}\right) \\
\frac{\partial \dot{m}_1}{\partial r} & \frac{\partial \dot{m}_1}{\partial \dot{r}} & 0 & \frac{\partial \dot{m}_1}{\partial \dot{\theta}} & \frac{\partial \dot{m}_1}{\partial m_1} & \frac{\partial \dot{m}_1}{\partial m_2} & \frac{\partial \dot{m}_1}{\partial m_3} \\
\frac{\partial \dot{m}_2}{\partial r} & \frac{\partial \dot{m}_2}{\partial \dot{r}} & 0 & \frac{\partial \dot{m}_2}{\partial \dot{\theta}} & \frac{\partial \dot{m}_2}{\partial m_1} & \frac{\partial \dot{m}_2}{\partial m_2} & \frac{\partial \dot{m}_2}{\partial m_3} \\
\frac{\partial \dot{m}_3}{\partial r} & \frac{\partial \dot{m}_3}{\partial \dot{r}} & 0 & \frac{\partial \dot{m}_3}{\partial \dot{\theta}} & \frac{\partial \dot{m}_3}{\partial m_1} & \frac{\partial \dot{m}_3}{\partial m_2} & \frac{\partial \dot{m}_3}{\partial m_3}
\end{pmatrix} \quad (5\text{-}10a)
$$

and

$$
K = \begin{pmatrix}
0 \\
\frac{1}{m}\frac{\partial F_r}{\partial p} - \frac{F_r}{m^2}\frac{\partial m}{\partial p} \\
0 \\
\frac{1}{mr}\frac{\partial F_\theta}{\partial p} - \frac{F_\theta}{m^2 r}\frac{\partial m}{\partial p} \\
\frac{\partial \dot{m}_1}{\partial p} \\
\frac{\partial \dot{m}_2}{\partial p} \\
\frac{\partial \dot{m}_3}{\partial p}
\end{pmatrix} \quad (5\text{-}10b)
$$

The partial derivatives contained in Eq. (5-10) are not readily evaluated directly, so expressions for each must be derived in terms of numerically computable quantities.

Prior to generation of the derivative expressions, it is necessary to evaluate the forces involved in the dynamics Eq. (3-7).

Another requirement is to remove the constraint implied by the quantity Φ in Eq. (2-20) where

$$0 < \Phi \leq 1 \qquad (5\text{-}11)$$

since the optimization algorithm requires a control quantity without constraints. A simple mathematical transformation yields

$$\Phi = \frac{1}{1 + u_f^2} \qquad (5\text{-}12)$$

where

u_f = an unbounded fuel flow function

Use of Eq. (5-12) and geometric considerations yield

$$F_r = (T + P) \sin \delta + N \cos \delta - \frac{GM_e m}{r^2} \qquad (5\text{-}13a)$$

$$F_\theta = (T + P) \cos \delta - N \sin \delta \qquad (5\text{-}13b)$$

$$\dot{m}_1 = -\rho_1 u_1 A_T r_{f_T} / (1 + u_f^2) \qquad (5\text{-}13c)$$

$$\dot{m}_2 = -\rho_1 u_1 A_S r_{f_S} m_{CR} / (1 + u_f^2) \qquad (5\text{-}13d)$$

$$\dot{m}_3 = -T_R / [I_{sp_R} g_0 (1 + u_f^2)] \qquad (5\text{-}13e)$$

$$\delta = \tan^{-1} \left(\frac{\dot{r}}{r(\dot{\theta} - \omega_e)} \right) + \alpha \qquad (5\text{-}13f)$$

where

$T = T_T + T_S + T_R$ = the total vehicle thrust, constrained to be in the body x-axis direction

$P = L \sin\alpha - D \cos\alpha$ = the aerodynamic force parallel to the body x-axis and of the same sign convention as the thrust

$N = L \cos\alpha + D \sin\alpha$ = the aerodynamic force normal to the body x-axis

also

δ = the body x-axis angle relative to the local horizontal

ρ_1 = density of air at propulsion system inlet

u_1 = velocity of air at propulsion system inlet

I_{sp_R} = rocket specific impulse

Differentiation of Eq. (5-13) yields the desired partial derivative relations in terms of computable quantities

$$\frac{\partial F_r}{\partial r} = \left(\frac{\partial T}{\partial r} + \frac{\partial P}{\partial r} - N \frac{\partial \delta}{\partial r}\right) \sin\delta + \left(\frac{\partial N}{\partial r} + (T + P) \frac{\partial \delta}{\partial r}\right) \cos\delta + \frac{2GM_e m}{r^3} \tag{5-14a}$$

$$\frac{\partial F_r}{\partial \dot{r}} = \left(\frac{\partial T}{\partial \dot{r}} + \frac{\partial P}{\partial \dot{r}} - N \frac{\partial \delta}{\partial \dot{r}}\right) \sin\delta + \left(\frac{\partial N}{\partial \dot{r}} + (T + P) \frac{\partial \delta}{\partial \dot{r}}\right) \cos\delta \tag{5-14b}$$

$$\frac{\partial F_r}{\partial \dot{\theta}} = \left(\frac{\partial T}{\partial \dot{\theta}} + \frac{\partial P}{\partial \dot{\theta}} - N \frac{\partial \delta}{\partial \dot{\theta}}\right) \sin\delta + \left(\frac{\partial N}{\partial \dot{\theta}} + (T + P) \frac{\partial \delta}{\partial \dot{\theta}}\right) \cos\delta \tag{5-14c}$$

$$\frac{\partial F_r}{\partial m_i} = - \frac{GM_e}{r^2} \tag{5-14d}$$

$$\frac{\partial F_r}{\partial p} = \left(\frac{\partial T}{\partial p} + \frac{\partial P}{\partial p}\right) \sin\delta + \frac{\partial N}{\partial p} \cos\delta - \frac{GM_e}{r^2} \frac{\partial m}{\partial p} \tag{5-14e}$$

$$\frac{\partial F_\theta}{\partial r} = \left(\frac{\partial T}{\partial r} + \frac{\partial P}{\partial r} - N \frac{\partial \delta}{\partial r}\right) \cos\delta - \left(\frac{\partial N}{\partial r} + (T + P) \frac{\partial \delta}{\partial r}\right) \sin\delta \tag{5-15a}$$

$$\frac{\partial F_\theta}{\partial \dot{r}} = \left(\frac{\partial T}{\partial \dot{r}} + \frac{\partial P}{\partial \dot{r}} - N \frac{\partial \delta}{\partial \dot{r}}\right) \cos \delta - \left(\frac{\partial N}{\partial \dot{r}} + (T + P) \frac{\partial \delta}{\partial \dot{r}}\right) \sin \delta \tag{5-15b}$$

$$\frac{\partial F_\theta}{\partial \dot{\theta}} = \left(\frac{\partial T}{\partial \dot{\theta}} + \frac{\partial P}{\partial \dot{\theta}} - N \frac{\partial \delta}{\partial \dot{\theta}}\right) \cos \delta - \left(\frac{\partial N}{\partial \dot{\theta}} + (T + P) \frac{\partial \delta}{\partial \dot{\theta}}\right) \sin \delta \tag{5-15c}$$

$$\frac{\partial F_\theta}{\partial m_i} = 0 \tag{5-15d}$$

$$\frac{\partial F_\theta}{\partial p} = \left(\frac{\partial T}{\partial p} + \frac{\partial P}{\partial p}\right) \cos \delta - \frac{\partial N}{\partial p} \sin \delta \tag{5-15e}$$

$$\frac{\partial \dot{m}_1}{\partial r} = -A_T r_{f_T} \frac{\partial (\rho_1 u_1)}{\partial r} / (1 + u_f^2) \tag{5-16a}$$

$$\frac{\partial \dot{m}_1}{\partial \dot{r}} = -A_T r_{f_T} \frac{\partial (\rho_1 u_1)}{\partial \dot{r}} / (1 + u_f^2) \tag{5-16b}$$

$$\frac{\partial \dot{m}_1}{\partial \dot{\theta}} = -A_T r_{f_T} \frac{\partial (\rho_1 u_1)}{\partial \dot{\theta}} / (1 + u_f^2) \tag{5-16c}$$

$$\frac{\partial \dot{m}_1}{\partial m_i} = 0 \tag{5-16d}$$

$$\frac{\partial \dot{m}_1}{\partial p} = -r_{f_T} \frac{\partial (\rho_1 u_1 A_T)}{\partial p} / (1 + u_f^2) \tag{5-16e}$$

$$\frac{\partial \dot{m}_2}{\partial r} = -A_S r_{f_S} \frac{\partial (\rho_1 u_1 m_{CR})}{\partial r} / (1 + u_f^2) \tag{5-17a}$$

$$\frac{\partial \dot{m}_2}{\partial \dot{r}} = -A_S r_{f_S} \frac{\partial (\rho_1 u_1 m_{CR})}{\partial \dot{r}} / (1 + u_f^2) \tag{5-17b}$$

$$\frac{\partial \dot{m}_2}{\partial \dot{\theta}} = -A_S r_{f_S} \frac{\partial (\rho_1 u_1 m_{CR})}{\partial \dot{\theta}} / (1 + u_f^2) \tag{5-17c}$$

$$\frac{\partial \dot{m}_2}{\partial m_i} = 0 \tag{5-17d}$$

$$\frac{\partial \dot{m}_2}{\partial p} = -r_{f_S} \frac{\partial (\rho_1 u_1 m_{CR} A_S)}{\partial p} / (1 + u_f^2) \tag{5-17e}$$

$$\frac{\partial \dot{m}_3}{\partial r} = 0 \tag{5-18a}$$

$$\frac{\partial \dot{m}_3}{\partial \dot{r}} = 0 \tag{5-18b}$$

$$\frac{\partial \dot{m}_3}{\partial \dot{\theta}} = 0 \tag{5-18c}$$

$$\frac{\partial \dot{m}_3}{\partial m_i} = 0 \tag{5-18d}$$

$$\frac{\partial \dot{m}_3}{\partial p} = - \frac{\partial T_R}{\partial p} / (I_{sp_R} g_0 [1 + u_f^2]) \tag{5-18e}$$

Also, differentiation of Eq. (5-13f) yields

$$\frac{\partial \delta}{\partial r} = - \frac{\dot{r}(\dot{\theta} - \omega_e)}{r^2 (\dot{\theta} - \omega_e)^2 + \dot{r}^2} \tag{5-19a}$$

$$\frac{\partial \delta}{\partial \dot{r}} = \frac{r(\dot{\theta} - \omega_e)}{r^2 (\dot{\theta} - \omega_e)^2 + \dot{r}^2} \tag{5-19b}$$

$$\frac{\partial \delta}{\partial \dot{\theta}} = - \frac{r\dot{r}}{r^2 (\dot{\theta} - \omega_e)^2 + \dot{r}^2} \tag{5-19c}$$

5.5 The Boundary Conditions of the Adjoined Functions

In Chapter IV, Eqs. (4-32) and (4-53) specified boundary condition relations on the required adjoint functions. The terms ϕ_x, Ω_x, Ψ_x, and Ψ_p are required along with f and L at terminal time.

Differentiation of Eq. (5-1) with respect to states yields

$$\phi_x = (0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1) \tag{5-20}$$

Differentiaiton of Eq. (5-8) with respect to states yields

$$\Omega_x = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0) \tag{5-21}$$

Differentiation of Eq. (5-9) with respect to states yields

$$\Psi_x = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5-22}$$

Differentiation of Eq. (5-9) with respect to parameters yields

$$\Psi_p = \begin{pmatrix} 0 \\ 0 \\ -\dfrac{\partial M_1}{\partial p} \\ -\dfrac{\partial M_2}{\partial p} \\ -\dfrac{\partial M_3}{\partial p} \end{pmatrix} \tag{5-23}$$

where the rows of Eq. (5-23) are row vectors of the same dimension as the parameter vector p.

Substitution of Eqs. (5-21), (5-20), (5-3), (5-2), and (3-7) into Eq. (4-32) yields

$$\lambda^T(\tau) = (0 \quad 0 \quad 0 \quad \lambda_4 \quad -1 \quad -1 \quad -1) \tag{5-24a}$$

where

$$\lambda_4 = \left[\frac{r}{\frac{F_\theta}{m} - 2\dot{r}\dot{\theta}}(\dot{m}_1 + \dot{m}_2 + \dot{m}_3 - Q(q - q_d)^2 u_0(q - q_d) - C_A(F_{sp} - a_d)^2 u_0(F_{sp} - a_d))\right]\Bigg|_{t=\tau} \tag{5-24b}$$

Substitution of Eqs. (5-23), (5-22), (5-21), and (3-7) into Eq. (4-53) yields

$$\Lambda_1^T(\tau) = \left.\begin{pmatrix} -1 & 0 & 0 & (\Lambda_0 \dot{r}) & 0 & 0 & 0 \\ 0 & -1 & 0 & \left(\Lambda_0 (r\dot{\theta}^2 + \frac{F_r}{m})\right) & 0 & 0 & 0 \\ 0 & 0 & 0 & (\Lambda_0 \dot{m}_1) & -1 & 0 & 0 \\ 0 & 0 & 0 & (\Lambda_0 \dot{m}_2) & 0 & -1 & 0 \\ 0 & 0 & 0 & (\Lambda_0 \dot{m}_3) & 0 & 0 & -1 \end{pmatrix}\right|_{t=\tau} \tag{5-25a}$$

where

$$\Lambda_0 = \left[\frac{r}{\frac{F_\theta}{m} - 2\dot{r}\dot{\theta}}\right]\Bigg|_{t=\tau} \tag{5-25b}$$

$$\Lambda_2^T(\tau) = \begin{pmatrix} 0 \\ 0 \\ \frac{\partial M_1}{\partial p} \\ \frac{\partial M_2}{\partial p} \\ \frac{\partial M_3}{\partial p} \end{pmatrix} \tag{5-25c}$$

CHAPTER VI

NUMERICAL DIFFICULTIES AND THEIR RESOLUTION

6.1 An Overview

As anyone experienced with computer implementation of complex mathematical algorithms would expect, a number of difficult numerical problems arose in applying the technique described in Chapter IV to the air-breathing launch vehicle optimization. These numerical problems originate from idiosyncrasies of the algorithm, nonlinearities of the system dynamics and constraints, and machine data manipulation effects. To minimize frustration and wasted effort for anyone who chooses to apply the techniques contained in this dissertation, these problems are discussed, and their resolutions are characterized below.

6.2 Loss of Significant Information

The gradient type optimization techniques characteristically have sums and differences of numbers with similar magnitude. The differences lead to a decrease in the number of significant digits in the information due to the finite truncation of all information manipulated by a computer. The interpolation of tabulated data to obtain gradient information compounds the effect. The iterative techniques used to evaluate some quantities contained in the dynamics equations can further accumulate the errors.

Double precision computation of all key variables, and double precision storage of all important data seems essential to prevent complete loss of necessary information.

6.3 Equality Constraint Effect on Stability of Convergence

The optimization algorithm described in Chapter IV includes the capability to contribute to the control vector variation a quantity associated with terminal state and parameter equality constraint violations. The initial size of the violations can influence the stability of the algorithmic convergence to an extremal point. To understand the effect, it is best to consider an unconstrained example.

Suppose one chooses to find an extremal point in a problem where the cost is a function of a single scalar quantity (see Figure VI-1). There may be local minima or maxima, but no absolute extremal value. In a simple physical example, this may imply a local minimum exists for fuel consumption as a function of positive fuel flow, but the absolute minimum might require an unrealizable negative fuel flow setting.

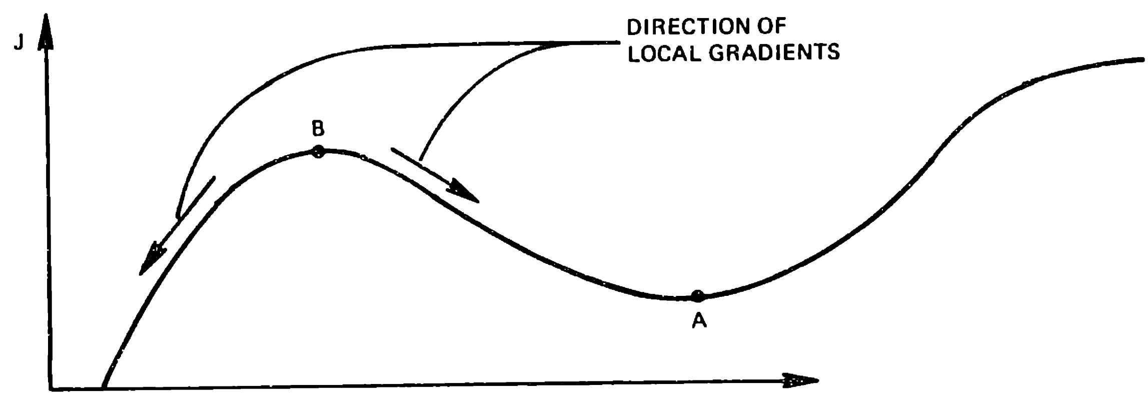


Figure VI-I.

Refering to Figure VI-1, one can see that point A represents a local minimum, point B a local maximum. In a minimization problem, convergence to point A will only result if one starts to the right of point B. To the left of point B the cost gradient points in a direction without solution.

Similar curve "bumpiness" and "over the hill" effects can be expected for equality constraint convergence—particularly when there are several interacting constraints. Too large an initial constraint violation is almost sure to push one over at least one multidimensional "hill". The result is that an initial set of controls, parameters, and transition times must be chosen to lead to an approximation of a physically realizable configuration, although suboptimal.

To judge how large a constraint violation can be tolerated which does not prevent constraint covergence, several rules of thumb, combined with a little trial and error can help.

First, the violations must be small enough such that nonlinearities in the dynamics do not direct the constraint related contributions in an entirely wrong direction.

Second, for state related constraints, energy considerations may be used to judge the relative magnitudes of upper bounds of constraint violations.

Third, the initial violations must have a scale far smaller than the natural scales of the problem.

In the air-breathing vehicle example, the rules lead to the following conclusions.

The atmosphere properties vary nonlinearly with altitude when viewed over intervals of more than a few thousand feet. The thrust of an air-breathing vehicle will superimpose that effect with its own Mach number related nonlinearities. The first rule suggests an upper bound in an altitude constraint violation of a few thousand feet.

If the equality constraints are applied at the ground end of flight, altitude provides a measure of vehicle potential energy increase, while velocity provides a measure of kinetic energy increase from ground state. The two are additive to obtain system energy changes. The potential energy is approximately linear in altitude over a few thousand feet. The kinetic energy is quadratic in velocity. Using the second rule, assuming the tolerable constraint violations can be comparable in energy measure, the maximum velocity constraint violations will be several tens of feet per second.

For parameter related constraints the third rule must be applied. Specifically, to match propellant consumed to propellant tank capacity, the initial constraint violations ought not be more than several percent of the expected fuel requirement.

Once the above listed conditions are met, constraint induced stability problems are not likely. Failure to satisfy the above conditions generally results in convergence in most constraints with divergence in at least one. (Often, the altitude constraint diverges in the air-breathing launch vehicle case.)

Finding an initial guess of a vehicle and trajectory that satisfies the equality constraint violation bounds established above is not a simple task. It can be done, however, through the iterative use of a forward time vehicle flight simulation combined with a forward time optimization routine which treats only equality constraint violations of states at the desired orbit.

The forward time simulation consists of a vehicle dynamics model, and a guessed set of parameters and controls. The air-breathing part of the flight is "flown" to fuel depletion, automatically satisfying propellant consumption/tank volume matching constraints. The rocket trajectory can then be experimentally "flown", with multiple control histories, to burnout to establish the feasible range of orbits for the configuration used in the guess. If the energy of the achievable orbits is higher than is desired, the entire configuration should be scaled down accordingly. If the energy of the achievable orbits is lower than is desired, scale the configuration up. When the desired energy level seems possible with a given configuration, experimentally find a control history that results in an orbital velocity very close to the desired value, with large altitude errors permissible. (At orbital velocities, small velocity changes have the same system energy effect as large altitude changes.)

The configuration obtained by the process just detailed is used as an initial guess for a forward time optimization. The optimization routine is the forward time equivalent of the method detailed in Chapter IV except that no performance measure is included (ϕ=L=0). Only equality constraints on the terminal orbital states are applied as a basis for choosing control variations.

The result of applying the forward time constraint violation reduction algorithm for a few iterations is a suitable configuration for the backwards time optimization technique. Even after roundoff errors introduced in inverting the time varying control histories, an initial guess with suitably small equality constraint violations is likely to result.

6.4 Choosing Metrics

Step size scaling within the various elements of the control vector from one iteration of the optimization algorithm to the next is accomplished through the use of the matrices U and V, referred to as metrics.

The metric V weights the variations of the parameters p and the transition times t_s, which are all time invariant. Consequently, the elements of V are time invariant.

The metric U weights the variations of the time varying control histories, and consequently should be time varying.

The rate of convergence to the extremal points is highly sensitive to the choice of U and V. In fact, the effect of the metrics on the convergence may be seen as a reflection of the nonuniformity of the performance function constant cost contours' curvature with respect to each of the control elements in the locality of the current control values. This leads to the conclusion that the best choice of U and V requires a good estimate of the nature of the second derivatives of the performance function geometry.

Since it is often difficult to discern much detail of the likely second order behavior prior to applying an optimization technique, some more "rules of thumb" must be used to get an initial set of metrics. Some experimentation with the values arrived at by these rules is suggested.

A desirable goal is to assure that the step taken in the direction of each control element leads to unit changes in performance. For the parameters, a good starting point is to assume that the curvature effects are approximately proportional to the initial value of the parameter.

The V matrix includes cross coupling terms in the off diagonal, and single element terms on the diagonal. Little is likely to be known, a priori, about interaction between parameters or transition times. Consequently, one should choose a diagonal matrix whose parameter associated elements are proportional to the square of the parameter values. (The square requirement results from the metric appearing in a quadratic term of control variations to control step size.) For the transition times, the characteristic dimension to judge metric size should be a substantial fraction of the total flight time.

The U matrix includes cross coupling terms which ought to be set to zero for reasons similar to those used in assigning off diagonal elements of the V matrix (i.e., ignorance of anything better to do). The diagonal elements are more difficult to select than for the V matrix, however. A magnitude is needed, and a shape is needed as a function of time.

The time integrated magnitude of the diagonal elements of U may be chosen by evaluating the time integrated magnitude of the corresponding controls over the entire flight, using these values as the characteristic dimension in a fashion similar to that used when selecting the V matrix diagonal values.

The shape as a function of time of the U matrix ought to reflect time dependent behavior of the performance function constant cost contour curvature which is very difficult to fathom without analytical formulation of all the dynamics equations. Experimental variation of the shape must be attempted to judge the effect on algorithmic convergence rates.

To avoid too much complexity without good physical justification, it is recommended that the shape function be no more complex than a linear function of time.

6.5 Choosing a Penalty Function Weighting Factor

The inequality constraints are accommodated through the use of quadratic penalties outside the constraint bounds. The weight assigned to the penalty functions influences the relative rate of reduction of the penalty terms and the fuel performance terms. The effect is not as strong as the metric values on resulting step size, but it does have an effect on how closely the constraint bounds are satisfied. It is probably not desirable to apply the full force of the penalities initially. The penalty coefficient's initial value should be chosen so that the cost contribution is comparable to, or a low multiple of, the expected non-penalty performance improvement, with optimization allowed to proceed until most of the initial penalty contribution is small. The penalty gain should be boosted and the process repeated until the desired convergence is achieved toward the inequality constraint bounds. The precise weighting factor should be chosen realizing that the bounds in the optimal solution will be slightly violated, since pushing them to zero is offset by cost increases in other terms. If the bound requirement is very rigid, a final gain ten to one hundred times higher is desirable than for loose bound requirements.

6.6 Choosing a Specified Cost Improvement

The quantity dJ_s, the specified cost improvement, determines the overall step size of the optimization algorithm perturbations in all dimensions at once. It clearly can force the step to range from nearly linear to highly nonlinear changes in the performance function of a system with nonlinear dynamics. A non-linear step may prevent stable algorithm convergence.

Choosing a proper value for a specified cost improvement requires that one decrease its magnitude as the extremal point is approached to prevent the larger and larger control element variations that would be required to satisfy the specified quantity. To fail to change the specified improvement results in increasingly nonlinear, and possibly destabilizing, performance measure changes. If one has a rough estimate of the cost improvement still possible, the specified cost improvement should never exceed 15-20% of that value, and should probably only be a few percent of the expected improvement when starting the optimization process. Stability of equality constraint violations should be monitored during convergence of cost to assure that the specified cost improvement is in fact small enough.

CHAPTER VII

COMPUTER RESULTS

7.1 An Overview

A software implementation of the algorithm specified in Chaper IV with the dynamics and boundary conditions specified in Chapters II, III, and V was developed and used to evaluate the characteristics of the optimal air-breathing vehicle geometry, propulsion history, and trajectory as a function of different bounds on the inequality constraints. (The software listing is provided in the appendix.)

Cases were run fixing the specific force bound while varying the dynamic pressure bound. Cases were also run fixing the dynamic pressure bound while varying the specific force bound. The details of the cases run, the results of the optimizations, and some physical interpretations are presented below.

7.2 Details of Computer Cases

All cases run imposed the same orbit and ground boundary conditions. The orbit chosen was a 75 mile circular orbit. This represents the delivery of a payload to an orbit at the outer fringe of the sensible atmosphere where a small propulsive device similar to the space shuttle orbital maneuvering system would be capable of transferring the vehicle to the desired low earth orbit parameters. The ground conditions chosen represent horizontal sea level takeoff at an air speed of 250 miles per hour. The takeoff speed was chosen as a compromise between low takeoff speed to keep runway requirements down, and a lift requirement from the vehicle first stage to achieve flight with minimum wing size.

All cases run assumed equatorial launch to equatorial orbit, allowing use of two-dimensional orbital mechanics. A consequence of these assumptions is also minimal energy required to achieve orbit.

(The launch was assumed to be due East.) The results therefore shed the most favorable light on the potential of the air-breathing vehicle.

The specific inequality constraints used in each case are provided in Table VII-1. The dynamic pressure bounds were chosen to cover the range of values considered reasonable for air-breathing vehicles in a wide range of studies. The specific force bounds were chosen to keep g loads endurable on any passengers who may fly the vehicle (the space shuttle limits are close to 600 psf and 3 g's). The initial guess used for all runs was the same, and consistently quite different from the converged solutions. Convergence was assumed to occur when the gradient magnitude on any iteration was less than two orders of magnitude smaller than that of the initial iteration.

Table VII-1. Inequality constraints on computer runs.

Case	Dynamic Pressure Bound lbf/ft^2	Specific Force Bound g's
I	1000	3.0
II	800	3.0
III	600	3.0
IV	800	2.5

7.3 Optimization Data

The data from the optimization runs is presented tabularly and in plots. Tables VII-2 through VII-5 present the parameters, transition point times, relevant states at the transition points, and fuel consumption data for the cases run. Figures VII-1 through VII-20 present the interesting time dependent variables including controls or control-dependent functions, inequality constrained variables, two state related functions, and Mach number.

Table VII-2 presents the parameters in units of radians, feet, slugs, and pounds force. The meaning of each parameter is labeled.

Table VII-3 presents transition point times and other relevant flight times in seconds from ground takeoff. The significance of each time is labeled in Table VII-3.

Table VII-4 presents some relevant state variable related data at the transition points for each case run in feet, radians, and seconds.

Table VII-2. Parameter values.

Parameter	Physical Variable	Case			
		I	II	III	IV
p_1	Nose Angle	0.01925	0.01909	0.01879	0.01912
p_2	Fuselage Width	34.3	35.2	35.7	34.9
p_3	Scramjet Inlet Height	14.56	15.80	16.66	15.53
p_4	Turbojet Inlet Height	9.66	10.25	10.74	9.92
p_5	Wing Span	323	293	271	308
p_6	Delta Wing Angle	1.340	1.272	1.344	1.313
p_7	TJ/SCRJ Tank Volume Ratio	0.417	0.417	0.418	0.417
p_8	Fuselage Length	151.6	150.9	149.5	151.0
p_9	Rocket Fuel Capacity	3.16×10^4	3.16×10^4	3.15×10^4	3.15×10^4
p_{10}	Maximum Rocket Thrust	1.724×10^6	1.738×10^6	1.820×10^6	1.726×10^6

Table VII-3. Transition and flight times.

Time	Physical Meaning	Case			
		I	II	III	IV
t_{s_1}	Scramjet Ignition	42.4	55.3	65.3	47.8
t_{s_2}	Turbojet Shutdown	161.7	169.9	183.3	167.5
t_{s_3}	Staging	179.0	190.9	213.2	179.6
τ	Total Flight Time	512	521	532	513
$\tau-t_{s_3}$	Rocket Flight Time	333	331	319	333

Table VII-4. Transition point state functions.

	Transition Point	Altitude	Radial Velocity	Angular Velocity	Tangential Air Speed
Case	t_{s_i}	$r \times 10^4$	$\dot{r}$	$\dot{\theta} \times 10^{-4}$	$(r+Re)(\dot{\theta}-\omega_e) \times 10^3$
I	1	2.77	781	1.291	1.175
II	1	3.49	634	1.356	1.313
III	1	3.94	573	1.347	1.293
IV	1	3.24	702	1.301	1.196
I	2	7.82	136	2.31	3.32
II	2	7.55	103	2.32	3.33
III	2	7.48	43.1	2.30	3.30
IV	2	7.01	2.92	2.35	3.39
I	3	7.99	186	2.30	3.29
II	3	7.72	231	2.31	3.32
III	3	7.72	306	2.30	3.30
IV	3	7.18	344	2.33	3.36

Table VII-5. Ground value of propellant masses.

	Turbojet Fuel	Scramjet Fuel	Rocket Fuel
Case	$m_1 \times 10^3$	$m_2 \times 10^2$	$m_3 \times 10^4$
I	3.16	4.42	3.16
II	3.20	4.45	3.16
III	3.14	4.36	3.15
IV	3.17	4.43	3.15

Table VII-5 presents total fuel consumption for each propulsive mode in slugs. The first mode is the turbojet using a hydrocarbon fuel. The second mode is the scramjet using liquid hydrogen. The third mode is the rocket using liquid hydrogen and liquid oxygen.

Figures VII-1 through VII-20 are plots of the interesting time dependent variables, and are divided into two groups. The first group, Figures VII-1 through VII-10, are plots of the cases with the specific force bound held at 3.0 g's and the dynamic presure bound varied. The second group, Figures VII-11 through VII-20, are plots of the cases with the dynamic pressure bound held at 800 lbf/ft^2 and the specific force bound varied.

Within each group of ten plots, the first four show control functions. The first two plots show angle of attack on different time scales (early flight/entire flight). The third and fourth plots show throttle setting on the same two time scales. By expanding the time scale in early flight, more detail is visible in the plots.

The fifth through seventh plots in each group show constrained variables. The first of these is dynamic pressure, plotted only in the range where the atmosphere is dense enough to cause the constraint to affect the vehicle. The other two plots show specific force on the same two time scales as the control variables.

The eighth through tenth plots in each group present variables plotted through first stage flight which show interesting structure in that flight phase only. The radial velocity and tangential air speed are plotted to show energy, velocity, and constraint tradeoff effects. The Mach number is presented to correlate other first stage effects with Mach number, and to show staging Mach number similarities.

7.4 Physical Effects in Optimal Results

A wide variety of physical effects influencing the optimization results are apparent upon inspection of the results of the different cases run. These include effects resulting from the dynamic differences in the two stages, effects resulting from aerodynamic properties of the first stage, and effects resulting from the influence of the inequality constraints.

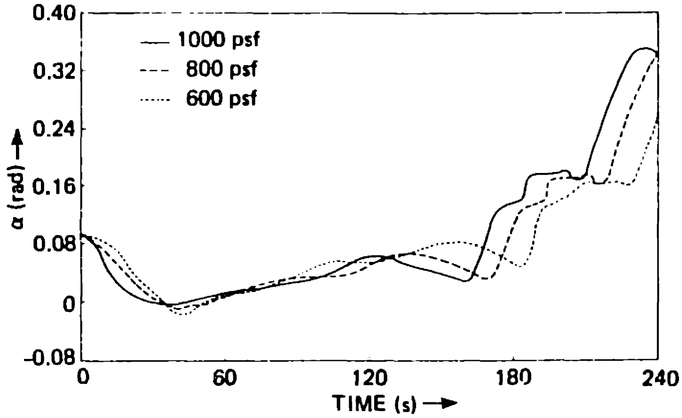


Figure VII-1. Angle of attack α vs. t. First 240 seconds. Specific force bound = 3 g's. Dynamic pressure bound varied.

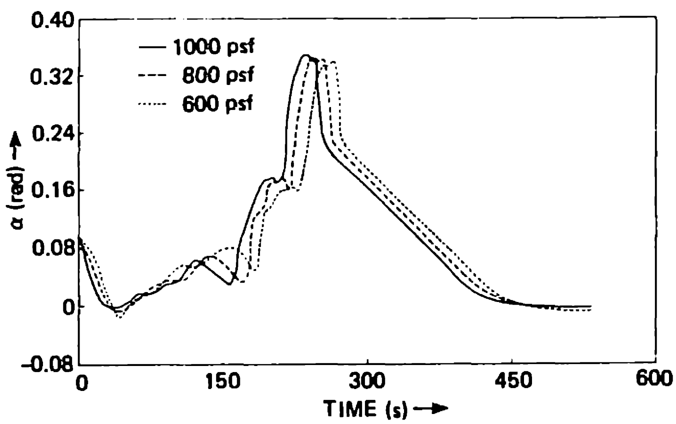


Figure VII-2. Angle of attack α vs. t. Full flight time. Specific force bound = 3 g's. Dynamic pressure bound varied.

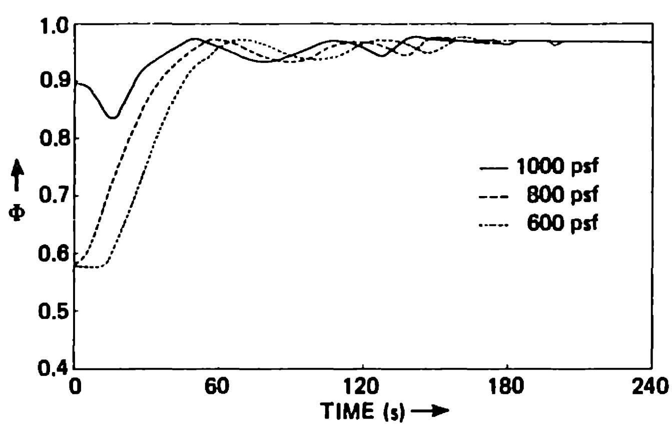


Figure VII-3. Throttle setting Φ vs. t. First 240 seconds. Specific force bound = 3 g's. Dynamic pressure bound varied.

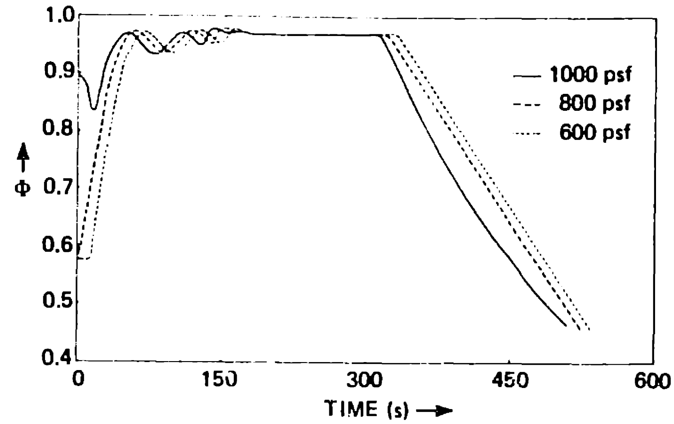


Figure VII-4. Throttle setting Φ vs. t. Full flight time. Specific force bound = 3 g's. Dynamic pressure bound varied.

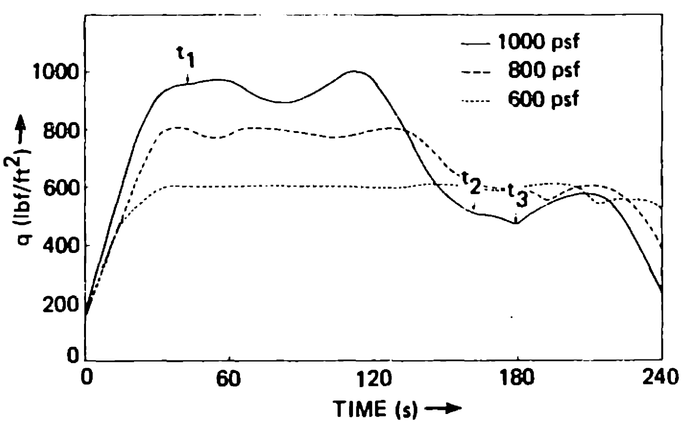


Figure VII-5. Dynamic pressure q vs. t. Through dense atmosphere. Specific force bound = 3 g's. Dynamic pressure bound varied.

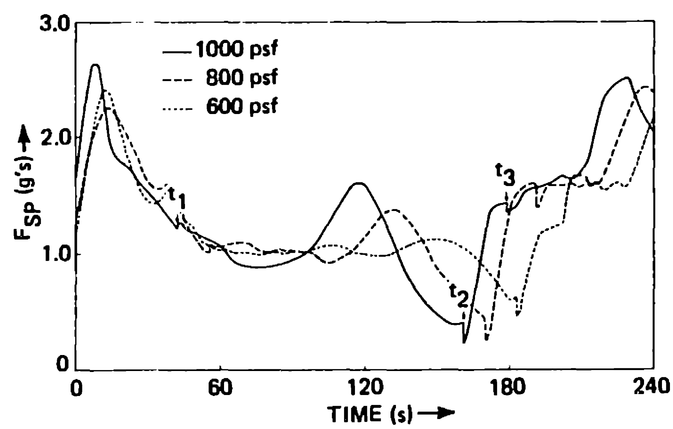


Figure VII-6. Specific force F_{SP} vs. t. First 240 seconds. Specific force bound = 3 g's. Dynamic pressure bound varied.

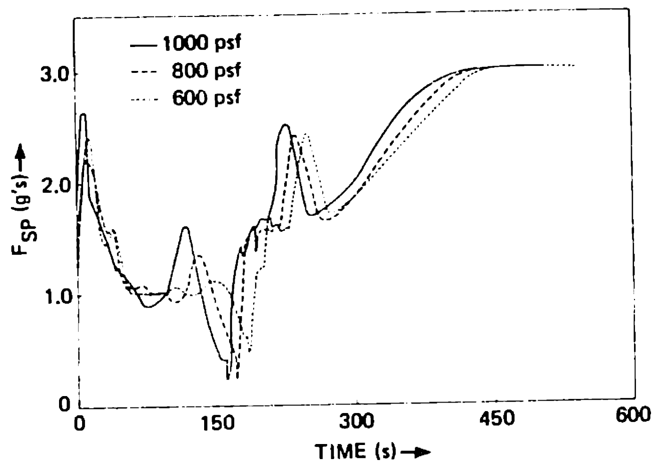


Figure VII-7. Specific force F_{SP} vs. t. Full flight time. Specific force bound = 3 g's. Dynamic pressure bound varied.

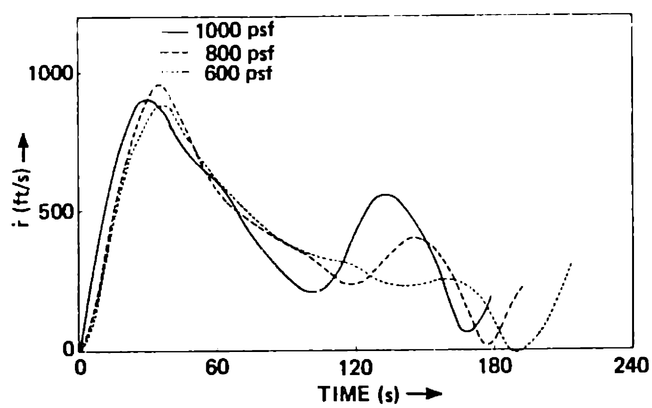


Figure VII-8. Radial velocity $\dot{r}$ vs. t. Through first stage flight. Specific force bound = 3 g's. Dynamic pressure bound varied.

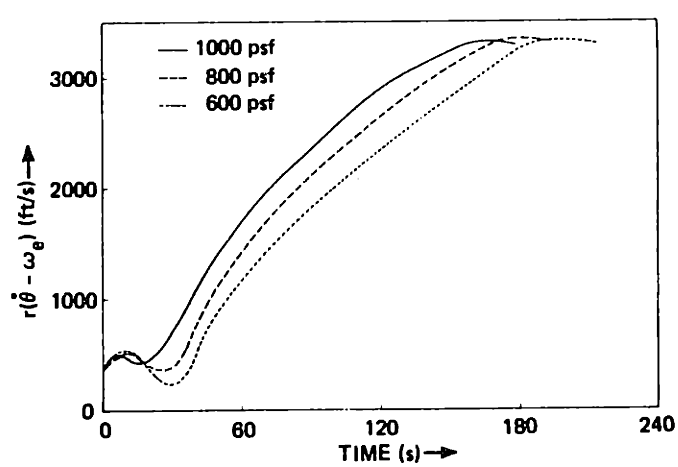


Figure VII-9. Tangential air speed $r(\dot{\theta}-\omega_e)$ vs. t. Through first stage flight. Specific force bound = 3 g's. Dynamic pressure bound varied.

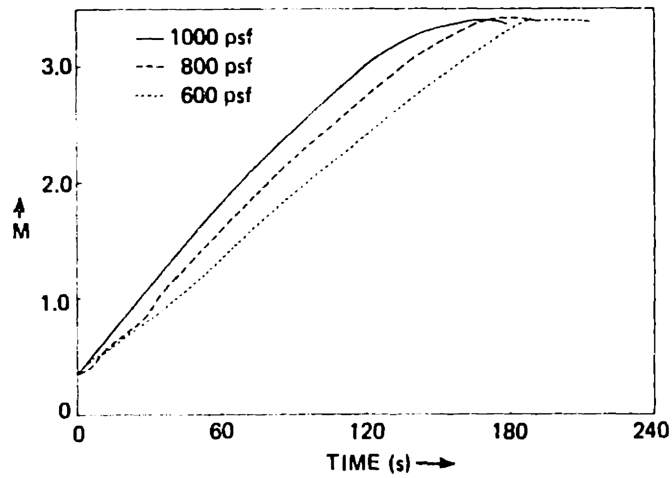


Figure VII-10. Mach number M vs. t. Through first stage flight. Specific force bound = 3 g's. Dynamic pressure bound varied.

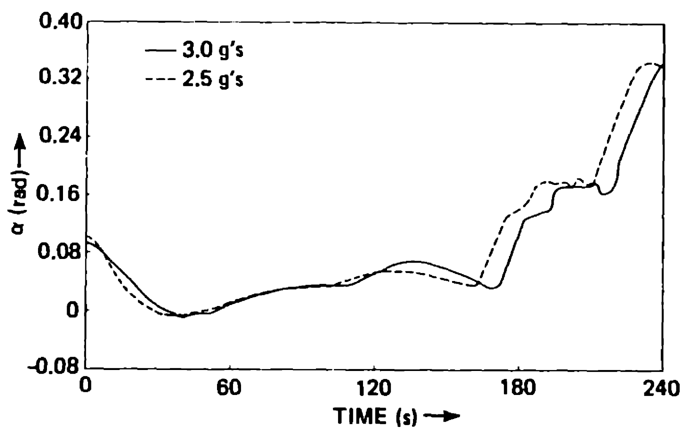


Figure VII-11. Angle of attack α vs. t. First 240 seconds. Dynamic pressure bound = 800 psf. Specific force bound varied.

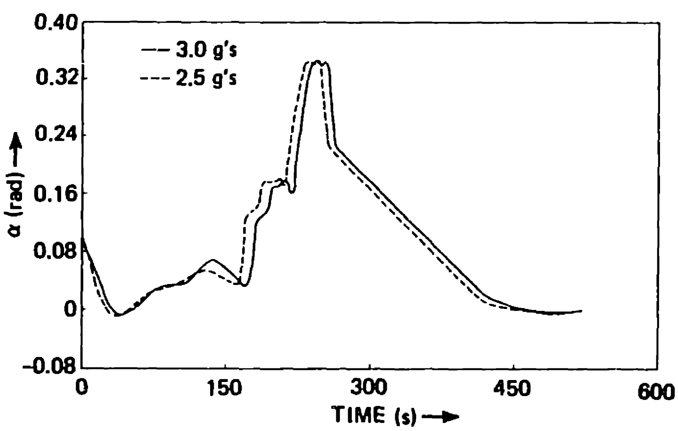


Figure VII-12. Angle of attack α vs. t. Full flight time. Dynamic pressure bound = 800 psf. Specific force bound varied.

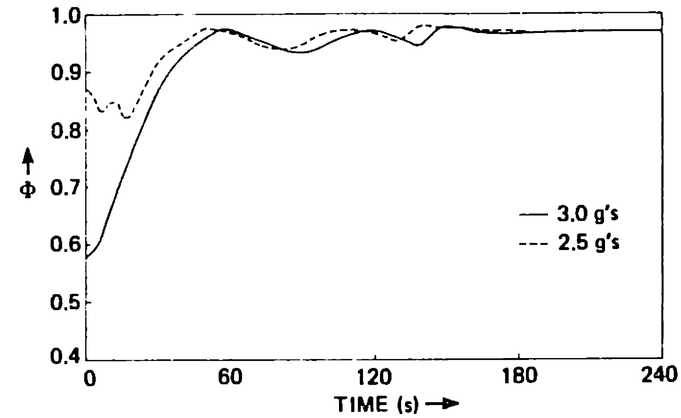


Figure VII-13. Throttle setting Φ vs. t. First 240 seconds. Dynamic pressure bound = 800 psf. Specific force bound varied.

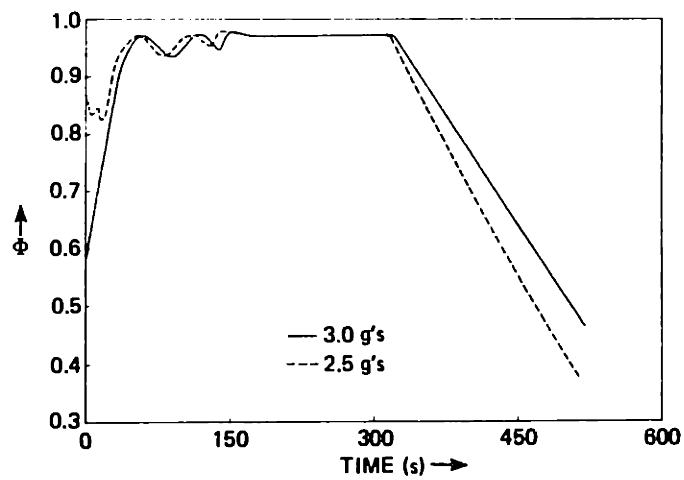


Figure VII-14. Throttle setting Φ vs. t. Full flight time. Dynamic pressure bound = 800 psf. Specific force bound varied.

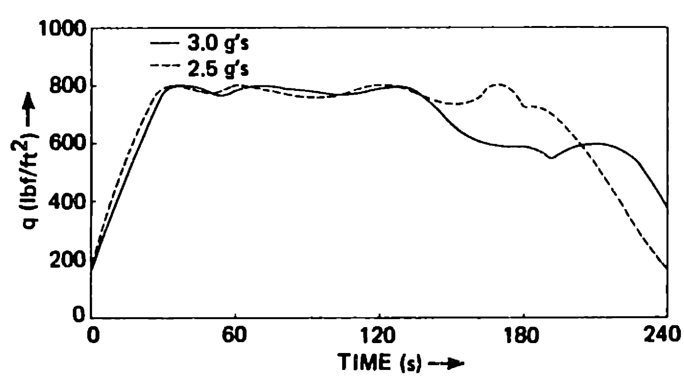


Figure VII-15. Dynamic pressure q vs. t. Through dense atmosphere. Dynamic pressure bound = 800 psf. Specific force bound varied.

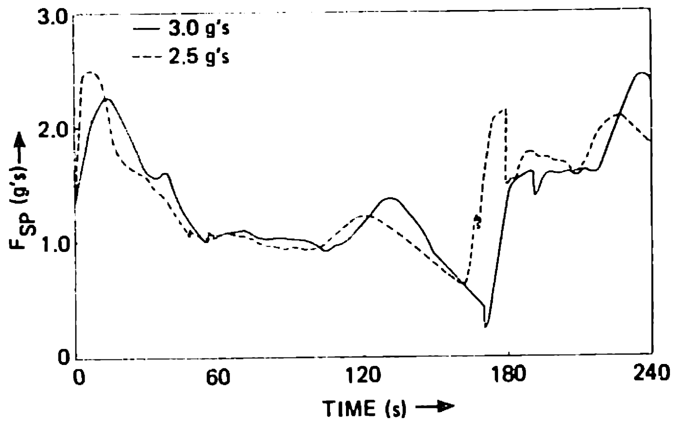


Figure VII-16. Specific force F_{sp} vs. t. First 240 seconds. Dynamic pressure bound = 800 psf. Specific force bound varied.

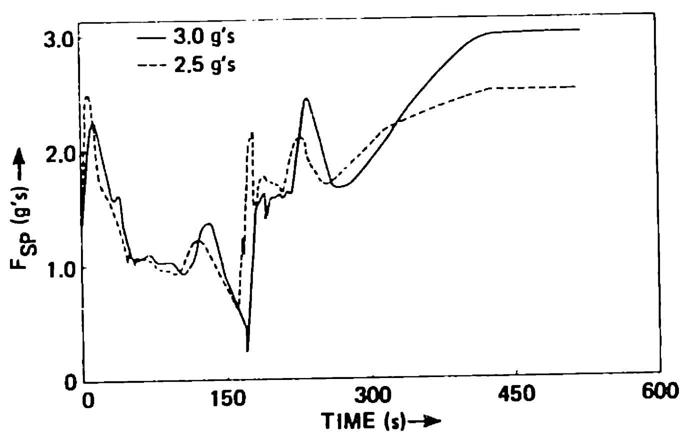


Figure VII-17. Specific force F_{sp} vs. t. Full flight time. Dynamic pressure bound = 800 psf. Specific force bound varied.

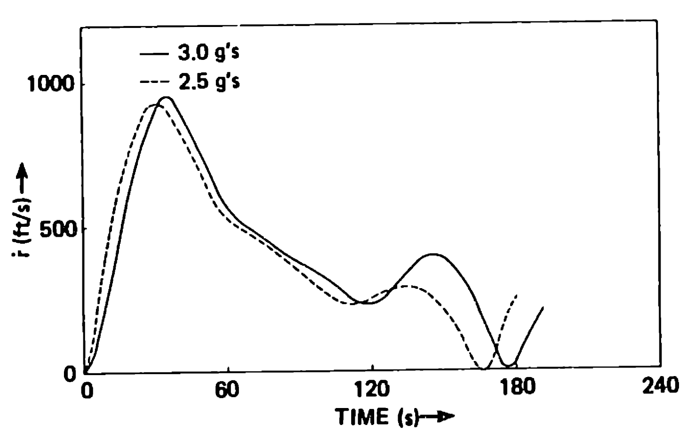


Figure VII-18. Radial velocity $\dot{r}$ vs. t. Through first stage flight. Dynamic pressure bound = 800 psf. Specific force bound varied.

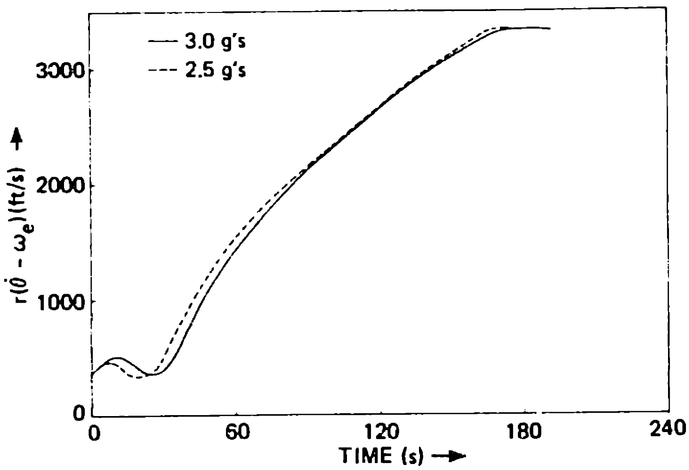


Figure VII-19. Tangential air speed. $r(\dot{\theta}-\omega_e)$ vs. t. Through first stage flight. Dynamic pressure bound = 800 psf. Specific force bound varied.

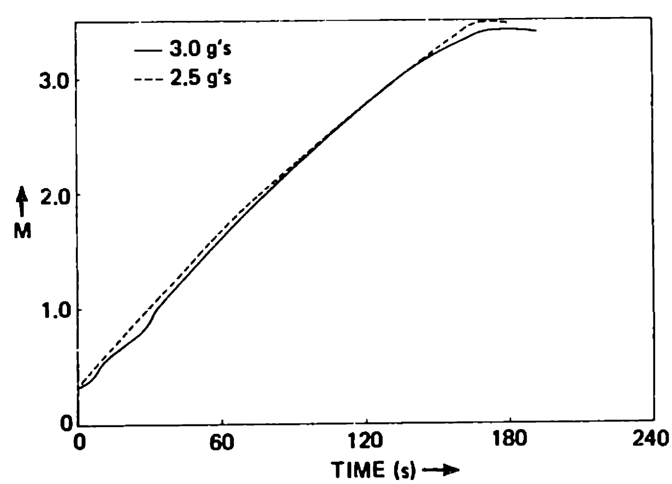


Figure VII-20. Mach number M vs. t. Through first stage flight. Dynamic pressure bound = 800 psf. Specific force bound varied.

Some trends are common to all cases run, indicating their association with the equality constraints and fuel performance cost function contributions. Other trends are apparent only upon variation of one of the inequality constraints, indicating a tradeoff between vehicle solution character and inequality constraint bounds.

7.4.1 Universal Solution Properties

A variety of state, trajectory, and propellant properties are apparent in all cases run, indicating their association with the specifics of vehicle dynamics as well as state and propellant constraints.

The most striking similarity in all cases run is seen in Table VII-5. The fuel consumption is only weakly associated with the inequality constraint bounds used. In fact, given the slow terminal convergence of gradient algorithms, the difference in the propellant consumption numbers can be attributed as much to small differences in the degree of optimality achieved at the cutoff of the optimization algorithm as it can be associated with genuine physical effects. In spite of simultaneous application of both inequality bounds, and substantial variation in each bound in the different cases run on the computer, it seems that within the range of inequality constraint bounds studied, fuel consumption effects of constraint-bound variations can be offset by vehicle configuration and trajectory changes.

Referring to Table VII-5 it is apparent that the high specific impulse of the air-breathing system when compared to the rocket leads to the vast bulk of the propellant consumption occurring during rocket flight in spite of the substantial velocity contributions of the first stage. (Data in Table VII-4 can be used to show air speed ranges between 3300 and 3400 feet per second at staging.) This implies that the air-breathing propulsion is very desirable, since a large gain in system energy is achieved with a relatively small fuel expenditure when compared to expected "first stage" rocket fuel usage rates on the space shuttle. It also suggests that significant air-breathing system improvements will only marginally improve overall system fuel usage.

Further reference to Table VII-4 shows that the general altitude range at the transition points is similar in all cases, with the altitude changing little between turbojet shutdown and staging. Table VII-3 shows a consistent time lapse of less than 20 seconds between these last two transition points.

Inspection of Figures VII-1 and VII-11 show consistent early angle-of-attack patterns during air-breathing flight. All cases start with relatively high angle of attack to achieve necessary lift for takeoff. The angle of attack subsequently undergoes a continuous decline in the first 35-45 seconds to a slightly negative value. The nadir of the angle-of-attack plots compared with the Mach number plots in Figures VII-10 and VII-20 consistently occur in the transonic region (between M = 1.0 and M = 1.2) where the drag coefficient peaks. This is the consequence of the utilization of stored potential energy (altitude) combined with thrust to rapidly go through the most dissipative aerodynamic flight region. An angle-of-attack increase follows to achieve climb, it peaks, then declines again as air-breathing thrust capability diminishes with altitude. The angle of attack bottoms out again at about the time when the turbojet shuts down. (t_{s_2} in Table VII-3 provides the shutdown time.) A pullup maneuver is then executed under scramjet power alone. The pullup orients the vehicle for effective use of the rocket power at staging. Since the rocket uses fuel much more rapidly than the air-breathing engines, improper rocket orientation can lead to inefficient use of rocket fuel to achieve a substantial change in flight path angle. The likely reason for scramjet power only in a primarily aerodynamic maneuver is the continued high specific impulse available to that propulsive mode under conditions where the turbojet specific impulse has dropped considerably. The dominance of the aerodynamic effect on the pullup is apparent upon inspection of the staging Mach number in Figures VII-10 and VII-20. Staging consistently occurs between M = 3.3 and M = 3.5. At these low Mach numbers the scramjet has low mass capture and therefore low power.

The consistency of the staging Mach number is itself an extremely important effect. Its value suggests that the scramjet provides little net benefit while contributing a high dry mass penalty to the first stage. Recalling that the scramjet mass model is based on the high mass of a system capable of tolerating the heat soak of high Mach numbers, this conclusion may in part be a consequence of design conservatism.

Figures VII-2 and VII-12 plot the angle-of-attack histories for the entire flight, consistently showing a large peak in the early phase of rocket flight followed by a dropoff to near zero. This effect is a consequence of the need to achieve a substantial radial velocity component in order to obtain the desired altitude at rocket burnout. The dropoff in angle of attack near the flight end is a consequence of an increasing use of gravity turning prior to orbit insertion.

Figures VII-3, VII-4, VII-13, and VII-14 all plot fuel throttle settings. They show a characteristic throttled-down setting at takeoff to prevent excessive vehicle loading that could result from the high mass capture rate and high lift that the dense atmosphere and high angle of attack induce. The throttle setting increases subsequently to achieve high thrust through the dissipative transonic region, followed by a slight throttling down in the less dissipative supersonic environment, and then a move toward maximum thrust is made (hence maximum acceleration) until specific force effects dominate in the late phases of rocket flight. The failure to achieve full throttle settings ($\Phi = 0.97$ rather than $\Phi = 1.00$) is a consequence of the slow terminal convergence of the gradient algorithm to the optimum and the optimization algorithm cutoff point.

In Figures VII-8 and VII-18 radial velocity through first stage flight is plotted showing a persistent double peak effect and pre-staging climb out. The second peak is affected by constraints which are discussed later. The first peak is a consequence of the transonic negative angle of attack followed by a need to stay in a sufficiently dense atmosphere to hold turbojet thrust levels high, permitting continued tangential velocity acceleration. The second peak seems to be a consequence of Mach number associated gains in rate of mass capture, permitting higher altitude flight with continued acceleration. The pullup, as discussed earlier, permits good vehicle orientation for rocket ignition.

Figures VII-9 and VII-19, the tangential air speed through first stage flight, show consistent acceleration except early and late in flight. The acceleration demonstrates that the main orbital energy benfit from the first stage comes in the tangential velocity component. The late velocity dropoff is a consequence of a combination of atmospheric rarefaction, reducing thrust, and finally an aerodynamic energy transfer to radial velocity to achieve vehicle pullup. The early flight tangential velocity dip is associated with a preferential early radial velocity gain to permit a subsequent altitude gain while accelerating. This effect permits vehicle load relief discussed later in more detail.

7.4.2 Dynamic Pressure Constraint Effects

A variety of trends in the vehicle geometric parameters, transition times, and time dependent behavior are apparent upon variation of the dynamic pressure inequality constraint bound. Three cases were run with the specific force bound held at 3.0 g's and the dynamic pressure bound set at 1000, 800, and 600 pounds force per square foot. Tables VII-2 through VII-5 and Figures VII-1 through VII-10 present the results of optimization with these bounds. (Table VII-1 specifies which bounds apply to each case run.)

Table VII-2 shows definite dynamic pressure related trends in the parameters p_1 through p_5, p_8, and p_{10}. Parameters p_2 through p_4 all define inlet areas for the air-breathing propulsion systems, and all grow with a decline in dynamic pressure bound. Comparison of altitudes at the first transition point in Table VII-4 shows a steady altitude increase for this point with the decrease in dynamic pressure bound. These effects taken together permit the vehicle to hold down the dynamic pressure due to the lower air density at higher altitude while keeping reasonable levels of air-breathing thrust as a result of larger propulsion system intake. Table VII-5 shows fuel consumption to be relatively constant during air-breathing flight for all cases run. An increase in p_2 combined with the constraint that demands the propellant volume to match the tank volume requires a decline in other fuselage dimensions. Thus, parameters p_1 and p_8 decrease. A secondary effect of increasing air-breathing propulsion system inlet size is to increase available thrust at takeoff, permitting somewhat reduced wing lift capability for the vehicle. A decline in p_5, the wing span, results with reduced dynamic pressure bound reflecting the lower lift requirements when combined with the delta wing sweep angle, p_6, and the $dc_L/d\alpha$ data in Table II-1. The increase in p_{10}, maximum rocket thrust, with declining dynamic pressure bound has an effect apparent in the bottom entry of Table VII-3, a decline in rocket flight time. With the staging states very similar, as seen in Table VII-4, the cause of the rocket thrust effect is obscure.

The time dependent variable plots show a great deal of dynamic pressure bound related structure. Figure VII-5 shows the dynamic pressure plotted through dense atmospheric flight. The most obvious effect is the close tracking of the dynamic pressure bounds, in all cases, until the decline in atmospheric density reduces the value below the bound for the remainder of flight. With less stringent constraints the bound is less precisely tracked. This is a result of the vehicle's ability to more closely approximate the desired unconstrained behavior. As the bound is decreased more time is spent near the limit, which is a consequence of greater constraints on acceleration. The increasing acceleration limits are seen in Tables VII-3 and VII-4 which show increasing amounts of time spent in air-breathing flight in spite of very similar staging states. The dynamic pressure plots also all show a temporary rise in dynamic pressure after staging as a result of velocity increase effects initially exceeding the atmospheric density decrease with altitude when the vehicle begins the high rate of rocket acceleration.

The angle-of-attack histories seen in Figures VII-1 and VII-2 all show similar structure. It is interesting, however, to note the differences in the pairs of air-breathing flight angle of attack valleys, the magnitudes of which are secondary effects of reduced wing lift capability. In the transonic negative angle of attack region a more negative angle of attack is permitted for tighter dynamic pressure bounds. The result is similar net negative lift, and similar first peak radial velocity behavior, seen in Figure VII-8, eventually leading to similar terminal first stage states. The second angle of attack valley is just prior to pre-rocket ignition pullup. Reduced wing lift capability results in a somewhat greater angle of attack to help keep up the net lift for the flight phase immediately preceeding the aerodynamic pullup maneuver. In spite of the higher angle of attack, net lift does decline with wing span in the flight region before the second radial velocity peak. The reduced lift also pulls down the magnitude of the second peak, and causes lower radial velocities just before the aerodynamic pullup.

Figure VII-6, specific force during early flight, shows one significant dynamic pressure bound dependent structure. The second peak declines with decreases in the bound. This effect is strictly aerodynamic, being a consequence of reduced lift from the reduced wing size, with inadequate increases in the angle of attack to compensate for the smaller wing.

The first stage tangential air speed, plotted in Figure VII-9, also shows one significant dynamic pressure bound related effect in early flight. The velocity component declines once in early flight in all cases run, with the decline becoming more pronounced as the dynamic pressure bound is reduced. The vehicle with a reduced bound needs to achieve higher altitude early in flight to keep air density down, controlling the dynamic pressure magnitude. The radial velocity component is given preference to achieve the altitude increase. Therefore, as the bound is reduced, greater emphasis is put on the radial component, leading to less energy being fed into the tangential component. The result, when aerodynamic dissipation is included, is a greater dip in magnitude of the tangential air speed.

7.4.3 Specific Force Constraint Effects

There are some physical phenomena apparent in the optimization results that are clearly a consequence of the specific force bounds. There is, however, more limited data from cases run in which the specific force bound was varied, making it necessary to be more cautious on conclusions drawn about the subtler effects.

Table VII-2 shows a trend in p_2 through p_4, the air-breathing propulsion system inlet sizing parameters. A decline in the specific force bound from Case II to Case IV results in a small but consistent decrease in all three parameters. Figure VII-16, the plot of specific force versus time in early flight shows that with the more constrained case the bound is reached in early air-breathing flight. The influence of the constraint would be expected to hold down the propulsion system size since acceleration is limited by the bound. Since this specific force peak precedes initiation of scramjet use the effect is principally on p_2 and p_4, the turbojet inlet dimensions. The effect on p_3 may be a consequence of reduced system dry mass with the decline in the turbojet size. The parameters p_1 and p_8 would be expected to show a small increase with a decline in p_2 due to the nearly constant fuel consumption figures of Table VII-5, combined with the propellant volume/tank volume matching constraint, which is in fact seen to be true, though it is a very small effect.

There is a significant increase in p_5, the wing span, with the decline in specific force bound. This effect, combined with the increase in p_6 and the effect on $dc_L/d\alpha$ given in Table II-1, actually has little effect on wing lift capability. Therefore, no trends can be extracted from the phenomena.

Figure VII-14 shows the most obvious effects of the specific force bound on time varying flight controls, the continuous decline in throttle setting in the later phase of rocket flight. The reason for the effect is clearly seen in Figure VII-17, the need to stay within the specific force bound as the rocket mass declines. Figure VII-14 shows that the lower specific force bound results in a steeper and greater decline in throttle setting, a consequence of similar rocket stage mass properties and maximum rocket thrust magnitudes, with substantially reduced acceleration bounds.

Figure VII-15 shows a more extended time spent near the dynamic pressure bound for the lower specific force bound case. This effect is a consequence of the lower staging altitude combined with high rocket acceleration in the denser atmosphere, pushing the dynamic pressure back up temporarily. The lower staging altitude also causes the higher specific force during the prestaging aerodynamic pullup maneuver, as seen in Figure VII-16. The higher atmospheric density generates more lift, leading to the higher specific force.

7.4.4 Conflict Between Inequality Constraints

There are circumstances under which application of stringent constraints in both specific force and dynamic pressure can force substantial changes in solution character making it difficult to satisfy both bounds while still matching the requirements of the terminal equality constraints. A case was run where this effect was evident.

A run with a dynamic pressure bound of 2.0 g's was attempted. Table VII-6 and Figures VII-21 through VII-23 present some relevant results.

Table VII-6. Data from 2.0 g's, 800 psf bound case. (Not a fully converged solution.)

Data Type	Value	Transition Point	Time (s)	Altitude (10^3 ft)
Fuselage Width	35.1 ft	Scramjet Ignition	27.1	22.9
Scramjet Height	16.21 ft.	Turbojet Shutdown	171.9	73.2
Turbojet Height	9.76 ft	Staging	173.7	73.4
Max. Rocket Thrust	1.679×10^6 lbf			
Staging Mach No.	3.56			
Rocket Flight Time	348 s			

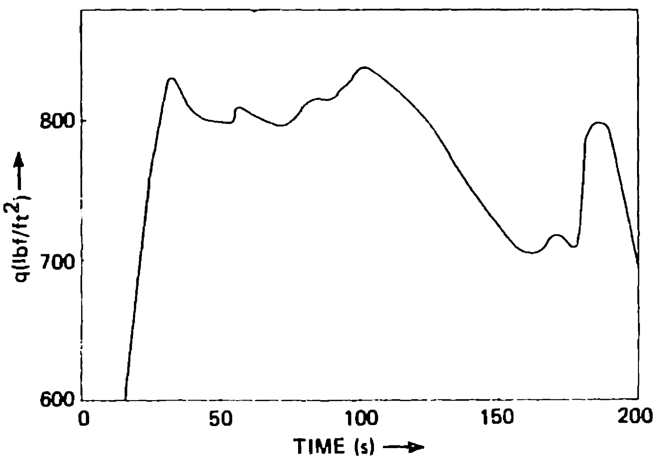


Figure VII-21. Dynamic pressure q vs. t. Bound violation region. Specific force bound = 2 g's. Dynamic pressure bound = 800 psf.

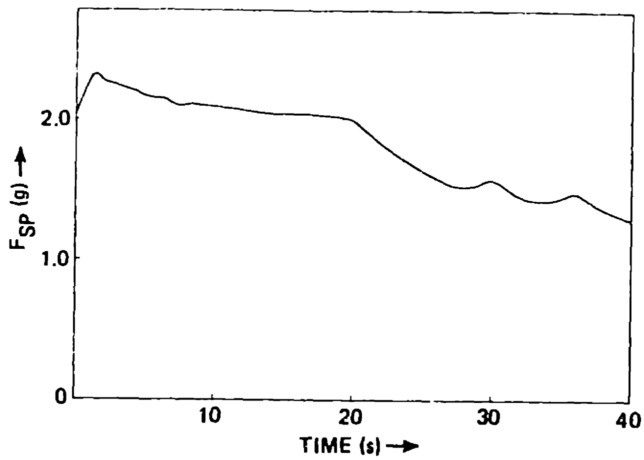


Figure VII-22. Specific force F_{sp} vs. t. Early flight. Specific force bound = 2 g's. Dynamic pressure bound = 800 psf.

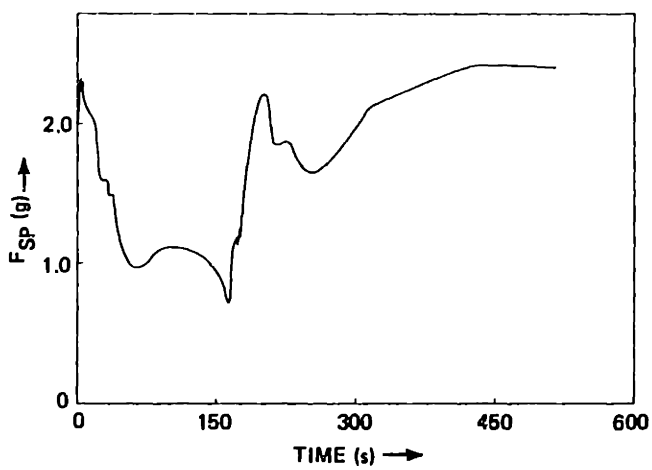


Figure VII-23. Specific force F_{sp} vs. t. Full flight time. Specific force bound = 2 g's. Dynamic pressure bound = 800 psf.

As one would expect with reduced specific force bounds, both the rocket and turbojet thrust capabilities decline, though slightly. The maximum rocket thrust is less than both cases II and IV, and the turbojet thrust capability, proportional to the product of the fuselage width and turbojet height, is less than both cases II and IV. However, inspection of Figure VII-21, the dynamic pressure in the high q range, and Figures VII-22 and VII-23, the specific force plots, show substantial violation of both bounds in spite of configuration and trajectory adjustments.

Comparison of Figures VII-21 and VII-22 show that the early flight specific force bound violations overlap the first peak in the dynamic pressure bound violations. Also, the dynamic pressure peak in rocket flight, at the right of Figure VII-21, hits the dynamic pressure bound at the same time as the second specific force bound violation peak occurs in Figure VII-23. This demonstrates that a conflict exists between the two bounds. The violation of the specific force bound in late rocket flight, seen at the right in Figure VII-23, is probably indirectly the result of the earlier time-bound difficulties combined with the continued requirement to satisfy the terminal equality constraints.

As a consequence of the overlapping violations, attempted satisfaction of one bound can adversely affect the magnitude of the violation of the other. The relative violation in the two different constraints depends on how heavily each violation is penalized.

The root of the problem is apparent upon inspection of the altitude of the first transition point. Comparison with cases II and IV show the altitude at scramjet ignition to decline with specific force bound. A lower altitude profile for the entire air-breathing flight, until near the staging point, results. With a similar staging altitude and Mach number to the other cases, higher Mach number flight at the lower altitudes results. The higher atmospheric density at the lower altitudes affects both dynamic pressure and specific force. Large changes in the early flight profile are likely to be necessary to get a suitable solution. The second order effects are likely to undergo major changes as well. This affects the choice of the metrics U and V, and makes the task of achieving convergence difficult.

7.5 Some Comments About the Scramjet

The low staging Mach number in all cases run on the computer provides little support for use of a scramjet in a vehicle with the flight profile and configuration studied. However, this should not be interpreted as a universal condemnation of the supersonic combustion propulsion system.

The turbojet system model includes a rapid dropoff of specific impulse near the staging Mach number. When this effect is coupled with the dynamic pressure constraint, thrust diminishes rapidly. The scramjet mass capture ratio remains well below unity, limiting its thrust contribution, and forcing vehicle staging to sustain acceleration under rocket power.

Several changes in the vehicle design considerations could change the scramjet contributions.

Use of a rocket in the first stage, or a single-stage-to-orbit vehicle, which could overlap with air-breathing propulsion, could drive the vehicle to a Mach number range where the scramjet thrust could sustain vehicle acceleration to a considerably higher Mach number.

Development of a higher temperature tolerant turbojet system, with the resulting higher impulse, could accelerate the vehicle to the Mach number range where scramjet thrust propogates the air-breathing flight to higher velocities.

Reduction in scramjet system mass by development of light weight high temperature tolerant materials (i.e., ceramics) could sufficiently reduce first stage dry mass to permit the diminished low Mach number scramjet thrust to continue to accelerate the vehicle to the higher scramjet thrust Mach number range, again permitting the continuation of air-breathing flight until the scramjet impulse declines, and atmospheric density is insufficient to permit much thrust.

7.6 Summary

A variety of physical effects are evident as a result of the optimization cases run. The effects are seen to depend both marginally and heavily on the inequality constraint bounds.

Fuel consumption is very weakly associated with the magnitude of the inequality constraint bounds within the ranges studied.

Staging consistently occurs near Mach 3.4 for all cases studied implying little if any gain from scramjet propulsion, though this could be partly the consequence of the conservatively high scramjet mass model.

Air-breathing propulsion fuel consumption is small for the substantial velocity gain that results.

Throttling capability is essential to stay within specific force bounds at both takeoff and in late rocket flight. The slope in the throttle history plots during rocket flight is a function of the bound.

The altitude and time of the first transition point, and the times of the second and third transition points are strongly tied to dynamic pressure bounds. The air-breathing propulsion inlets are also a function of the dynamic pressure constraints. Lower air density at higher altitude, and larger inlets to retain sufficient mass capture combine to meet increasingly stringent dynamic pressure bounds.

The relative rate of energy gain in the two velocity components is a function of constraints, and affects the structure of the plots in first stage flight.

Finally, the angle of attack plots show effects from physical influences of the environment. Required lift for takeoff results in a high angle of attack in early flight, with dissipation properties in transonic flight forcing it negative. An aerodynamic turn preceeding rocket ignition causes an increase, and radial velocity requirements after staging result in a sharp peak. Gravity turning then permits the angle of attack to approach zero at powered flight completion. The size of the effects in air-breathing flight are bound-dependent.

CHAPTER VIII

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

8.1 An Overview

A gradient type algorithm for two-point boundary value optimization problems, permitting evaluation of optimal parameter values, optimal dynamic discontinuity locations, and optimal history of time varying controls, has been developed and applied to a two-staged air-breathing launch vehicle concept. Mathematical peculiarities of the algorithm, and physical characteristics of the air-breathing launch vehicle have been evaluated. Many interesting conclusions can be drawn from the work already done. Some significant research areas have been identified which, if pursued, could significantly expand the capabilities of this approach to flight vehicle design analysis.

8.2 Conclusions

It has been demonstrated that a gradient-type algorithm incorporating several classes of controls, including time varying control histories, time invariant design parameters, and locations of dynamic discontinuities, and permitting application of equality and inequality constraints can be derived and successfully applied to a design problem involving complex system dynamics.

The stability and rate of algorithm convergence are highly sensitive to several factors. An initial guess at the solution is necessary and will violate the equality constraints to some degree. A gradient correction term for the violations is included, but excessive initial constraint violations will destabilize the algorithm. The algorithm incorporates weighting matrices for each control vector element, referred to as metrics, whose element magnitudes strongly affect convergence rates. Guesses at system second derivative properties provide a basis for good selection of the metric elements. The algorithm step size is tied to a desired improvement in performance per iteration, and inappropriate assignment of the specified improvement can also adversely influence convergence rates.

The algorithm can in principle handle any system with any number of elements in the control vector or constraint set, though practical considerations limit this. The computation time grows rapidly with system dimension, and resolution of the numerical problems listed above becomes much more difficult with growth in system complexity.

The algorithm has been successfully implemented on a digital computer and applied to a hypothetical two staged launch vehicle with an air-breathing first stage and rocket second stage.

The most significant conclusion about the proposed launch vehicle concept is that overall fuel consumption to deliver a space shuttle sized payload to low earth orbit is relatively insensitive to substantial variations in the inequality constraint bounds on dynamic pressure and specific force when vehicle geometry modifications, propulsion mode change time modifications, and trajectory changes are all allowed to accommodate the constraint-bound variations.

In all cases run, the staging Mach number, the transition from air-breathing flight to rocket flight, occurred between M = 3.3 and M = 3.5. The relatively low staging Mach number in spite of the availability of a supersonic combustion ramjet, implies that little is gained from an air-breathing system suited to high Mach number flight. Turbojet power alone in the first stage, followed by rocket flight, is likely to provide similar performance with a less complex system. These effects may, however, be propulsion system mass model dependent.

Highly variable throttle settings in the propulsion systems are necessary to accommodate the specific force bounds imposed on the vehicle. The effect is particularly obvious in the later stages of rocket flight when the bound is followed in spite of declining system mass.

Reducing the dynamic pressure bound expands the air-breathing propulsion system inlets and raises the early low Mach number flight altitude, though staging states are very similar.

There is no reason to suppose that application of other inequality constraints will produce results any less interesting. The general approach seems to work well with definite trends apparent after a limited number of optimization cases are run to completion.

8.3 Suggested Future Research

The material presented in this dissertation may be perceived as a demonstration of a design technique on the simplest possible mathematical model of an air-breathing launch vehicle. This was done to keep computer software complexity within reasonable limits. It is clear that a number of research extensions would be worthwhile in characterizing the detailed behavior and physical properties of the desired launch vehicle. Some of these areas are suggested below, and active investigation of each is encouraged.

8.3.1 Rollout Fuel Consumption

A factor contributing to fuel consumption of the air-breathing launch vehicle not considered in the dissertation is fuel consumed in accelerating to takeoff speed. Due to the high thrust nature of the vehicle, and short flight time of the air-breathing stage, this acceleration may have a non-negligible effect on performance. It would be worthwhile to incorporate it into the dynamics model. Also, an optimal takeoff speed would result, eliminating the effect of fixing it as was done in the dissertation research.

8.3.2 Booster Flyback

A fully reuseable launch vehicle will require first stage flyback with corresponding fuel consumption and aerodynamic effects. The algorithm has not been designed to treat two separate vehicle trajectories simultaneously as would be required to track the booster stage after separation. Extension of the technique to handle the flyback booster problem would be enormously useful.

8.3.3 Scramjet Removal

The low staging Mach number and high scramjet structural weight suggest marginal benefit of a propulsion system that will require large economic investment to develop. Investigation of the comparative performance of a vehicle with turbojet and rocket propulsion only would be useful.

8.3.4 Better Wing Models

Development of a software package allowing greater wing shape flexibility and characterizing fuselage wing interaction explicitly would lead to more realistic configuration conclusions.

8.3.5 Aerodynamic Interaction Between Stages

Aerodynamic interaction between stages in the noted configuration was ignored. The drag contributions due to interference effects are probably not negligible. Though modelling these effects is complicated it would be a worthwhile endeavor.

8.3.6 Variable Rocket Fuel Mix

Work has been done elsewhere[33,34] evaluating methods of mixing fuel types or changing fuel/oxidizer ratios in rockets to improve launch performance. Applying these techniques to the rocket flight portion of the air-breathing launch vehicle may result in some performance improvement. To permit comparison of the air-breathing vehicle to the pure rocket configurations incorporating these technologies requires variable rocket propulsion models to be included in the analysis.

8.3.7 Interactive Design

Many design considerations are not easily expressed mathematically, but can be perceived by an experienced engineer upon inspection. Permitting interactive optimization via CRT monitored results and plotting, and allowing interactive constraint adjustment may lead to rapid prototype design development. While improved computer processing rates may be necessary to make this technique effective, development of interactive software packages and CRT driver routines would be worthwhile.

8.3.8 Adaptive Metrics

Convergence of the optimization routine is not generally uniform in all dimensions, leading to the need to adjust the metrics to accommodate changing second derivative properties in the different control elements as the extremal point is approached. Devising a method to automatically adjust the metrics on a per iteration basis would be useful and would save substantial computer processing time.

8.3.9 Higher Fidelity Vehicle Dynamics Model

A variety of improvements to the dynamics model of the vehicle already studied can be applied to permit study of more detailed system behavior.

System rotational dynamics could be included, requiring aerodynamic pressure distributions as well as vehicle inertial properties. This would clearly affect the rate of change allowed in angle of attack.

Three dimensional orbital mechanics could be included, allowing study of the effects of inclined orbits and crossrange requirements on system behavior and performance.

Internal modeling of air-breathing propulsion dynamics could be developed to permit separate optimization of turbine and compressor sizing as well as inlet diffuser and outlet expansion nozzle shape. More accurate system dry mass properties would result.

The consequence of all the improvements mentioned above is a considerably higher control vector dimension, more state variables, and more constraints resulting in greatly expanded computer processing time requirements.

8.3.10 Single Stage to Orbit with Air-Breathing Propulsion

Some work has been done on single-stage-to-orbit launch vehicles including air-breathing propulsion in the lower Mach number flight regions.[35] Optimization efforts could lead to interesting conclusions, particularly with the possibility of overlapping air-breathing and rocket power.

8.3.11 Less Conservative Scramjet Mass Models

The low staging Mach numbers obtained in the cases shown in Chapter VII suggest that the scramjet mass models may have been based on too high a heat flux tolerance. Comparison of results with cases using a more optimistic dry mass model would be useful to judge the true scramjet launch vehicle potential.

APPENDIX

OPTIMIZATION SOFTWARE PROGRAM

The following material is a compilation listing of the software implementation of the algorithm in Chapter IV with the dynamics and boundary conditions given in Chapters II, III, and V. All system-dependent references and debugging routines have been extracted.

The code is HAL, a language developed for use in the space shuttle program, and should be easily read. A few notes on how to read some keys in the code, how to recognize certain mathematical operands, and how to use the cross reference are useful.

Within the code listing, on the left, one will note statement numbers followed by the letters C, E, M, or S. The letter C simply denotes a nonexecutable comment. The other three letters specify the type of code line to the right. E represents the exponent line, if any, to operate on the M line representing the middle or main code line. S represents subscripts, and can occur more than once in a given statement if subscripts are nested.

To the right, after each statement, is a note of the procedure in which the statement is contained.

At the end of each procedure a listing of variables used within, but defined outside, is given. The mention of a compool is a reference to the HAL equivalent of a common block, found at the beginning of the listing in RUN_POOL.

Most mathematical operations are self-evident, but the notation for different multiplication routines may not be obvious.

For a scalar multiple of any function, a space is used. For vectors, the inner product uses a period (the "dot" product), the outer product uses a space, and the cross product uses a single asterisk. Matrix/vector multiples use a space. Division uses a slash.

The cross reference contains much useful information. For each variable, a specification to type of usage in each statement is given to the left of the statement number. The key is at the top of the cross reference. Multiple types of usages in a single statement add together in the key. At the end of the cross reference is a listing of preprogrammed function calls. This includes normal math functions. Some nonstandard functions are also included. These are Modulo Counters ("MOD"), scalar-to-integer roundup and round down functions ("CEILING" and "FLOOR"), scalar to integer truncation ("TRUNCATE") and matrix transposition ("TRANSPOSE").

```
 STMT                                   SOURCE                                              CURRENT SCOPE
    D| INCLUDE @@RUNPOOL                                                                   |
       *****START OF INCLUDED MEMBER, RVL 00, CATENATION NUMBER 1*****
  1+M| RUN_POOL:                                                                           | RUN_POOL
  1+M| EXTERNAL COMPOOL;                                                                   | RUN_POOL
  2+M|    DECLARE NUM_STATES INTEGER CONSTANT(7);                                          | RUN_POOL
  3+M|    DECLARE EARTH_RADIUS SCALAR DOUBLE CONSTANT(2.09E07);                            | RUN_POOL
  4+M|    DECLARE UNIVERSAL_G_CONSTANT SCALAR DOUBLE CONSTANT(3.44E-08);                   | RUN_POOL
  5+M|    DECLARE EARTH_MASS SCALAR DOUBLE CONSTANT(4.1E23);                               | RUN_POOL
  6+M|    DECLARE NOZZLE_ANGLE SCALAR DOUBLE CONSTANT(.209);                               | RUN_POOL
  7+M|    DECLARE H2_DENSITY SCALAR DOUBLE CONSTANT(.136);                                 | RUN_POOL
  8+M|    DECLARE HYDROCARBON_DENSITY SCALAR DOUBLE CONSTANT(1.363);                       | RUN_POOL
  9+M|    DECLARE TJ_MAX_FUEL_AIR_RATIO SCALAR DOUBLE CONSTANT(.0633);                     | RUN_POOL
 10+M|    DECLARE SCRJ_MAX_FUEL_AIR_RATIO SCALAR DOUBLE CONSTANT(.027778);                 | RUN_POOL
 11+M|    DECLARE EARTH_OMEGA SCALAR DOUBLE CONSTANT(7.29E-05);                            | RUN_POOL
 12+M|    DECLARE G0 SCALAR DOUBLE CONSTANT(32.28863808);                                  | RUN_POOL
 13+M|    DECLARE ROCKET_ISP SCALAR DOUBLE CONSTANT(450.);                                 | RUN_POOL
 14+M|    DECLARE MIN_SCRJ_MASS_PER_FT SCALAR DOUBLE CONSTANT(15.);                        | RUN_POOL
 15+M|    DECLARE GAMMA0 SCALAR DOUBLE CONSTANT(1.4);                                      | RUN_POOL
 16+M|    DECLARE GAM3 SCALAR DOUBLE CONSTANT(GAMMA0 - 1.);                                | RUN_POOL
 17+M|    DECLARE GAM1 SCALAR DOUBLE CONSTANT(GAM3 / 2.);                                  | RUN_POOL
 18+M|    DECLARE GAM2 SCALAR DOUBLE CONSTANT(GAMMA0 + 1.);                                | RUN_POOL
 19+M|    DECLARE R_0 SCALAR DOUBLE CONSTANT(1727.76);                                     | RUN_POOL
 20+M|    DECLARE DEGREES_PER_RADIAN SCALAR DOUBLE CONSTANT(57.295779513082);              | RUN_POOL
 21+M|    DECLARE L_D_SCALE_FACTOR SCALAR DOUBLE CONSTANT(3.);                             | RUN_POOL
 22+M|    DECLARE MAX_ZLD_INDEX INTEGER CONSTANT(16);                                      | RUN_POOL
 23+M|    DECLARE M_ZLD ARRAY(MAX_ZLD_INDEX) SCALAR DOUBLE CONSTANT(.25, .6, .8, .9, .95, 1.05, 1.1, 1.2, | RUN_POOL
 23+M|    1.3, 1.5, 2., 3., 4., 5., 8., 10.);                                              | RUN_POOL
 24+M|    DECLARE ZLD ARRAY(MAX_ZLD_INDEX) SCALAR DOUBLE CONSTANT(.0203, .0208, .0228, .0268, .0351, .0492 | RUN_POOL
```

```
STMT                                   SOURCE                                                                      CURRENT SCOPE
 24+M| , .0507, .0518, .0521, .0516, .047, .0378, .0327, .0295, .0263, .026);                                     | RUN_POOL
 25+M| DECLARE MCR_MAX_INDEX INTEGER CONSTANT(5);                                                                 | RUN_POOL
 26+M| DECLARE MCR_M ARRAY(MCR_MAX_INDEX) SCALAR DOUBLE CONSTANT(2.5, 3.5, 5., 5.5, 8.);                          | RUN_POOL
 27+M| DECLARE MCR_C ARRAY(MCR_MAX_INDEX) SCALAR DOUBLE CONSTANT(.35, .6, .8, .85, .95);                          | RUN_POOL
 28+M| DECLARE MAX_FS_AR_INDEX INTEGER CONSTANT(5);                                                               | RUN_POOL
 29+M| DECLARE MAX_FS_M_INDEX INTEGER CONSTANT(10);                                                               | RUN_POOL
 30+M| DECLARE FS_CD_MAT MATRIX(MAX_FS_AR_INDEX, MAX_FS_M_INDEX) DOUBLE CONSTANT(.77, .79, .80, .86,              | RUN_POOL
 30+M| .89, .92, 1.0, 1.1, 1.45, 2.0, .47, .48, .42, .39, .41, .43, .58, .69, 1.1, 1.99, .30, .30, .27,          | RUN_POOL
 30+M| .24, .26, .27, .34, .48, 1.0, 1.99, .25, .25, .23, .21, .22, .24, .31, .46, 1.0, 1.99, .22, .22,          | RUN_POOL
 30+M| .21, .195, .21, .22, .28, .45, 1.0, 1.99);                                                                 | RUN_POOL
 31+M| DECLARE FS_CL_MAT MATRIX(MAX_FS_AR_INDEX, MAX_FS_M_INDEX) DOUBLE CONSTANT(.012, .013, .014,                | RUN_POOL
 31+M| .0145, .0155, .015, .0145, .014, .012, .0075, .020, .023, .025, .028, .032, .030, .028, .024,             | RUN_POOL
 31+M| .015, .0075, .040, .044, .048, .053, .068, .062, .053, .038, .0175, .0075, .053, .059, .063,               | RUN_POOL
 31+M| .072, .087, .074, .062, .042, .018, .0075, .065, .070, .073, .081, .098, .080, .070, .044, .019,          | RUN_POOL
 31+M| .0075);                                                                                                    | RUN_POOL
 32+M| DECLARE FS_M_VAL ARRAY(MAX_FS_M_INDEX) SCALAR DOUBLE CONSTANT(.25, .6, .7, .9, 1.1, 1.2, 1.5, 2.          | RUN_POOL
 32+M| , 4., 8.);                                                                                                 | RUN_POOL
 33+M| DECLARE FS_A_VAL ARRAY(MAX_FS_AR_INDEX) SCALAR DOUBLE CONSTANT(.5, 1., 2., 3., 4.);                        | RUN_POOL
 34+M| DECLARE DELIVERED_MASS SCALAR DOUBLE CONSTANT(6500.);                                                      | RUN_POOL
 35+M| DECLARE DELIVERED_PLANFORM_AREA SCALAR DOUBLE CONSTANT(4000.);                                             | RUN_POOL
 36+M| DECLARE MAX_SS_ANGLE_INDEX INTEGER CONSTANT(5);                                                            | RUN_POOL
 37+M| DECLARE MAX_SS_M_INDEX INTEGER CONSTANT(16);                                                               | RUN_POOL
 38+M| DECLARE SS_CL_MAT MATRIX(MAX_SS_ANGLE_INDEX, MAX_SS_M_INDEX) DOUBLE CONSTANT(-.4974, -.5239, -            | RUN_POOL
 38+M| .5734, -.6084, -.6373, -.6308, -.6283, -.6034, -.5590, -.4784, -.3735, -.3011, -.2390, -.2073, -          | RUN_POOL
 38+M| .1860, -.1300, -.0393, -.0469, -.0501, -.0500, -.0501, -.0298, -.0257, -.0197, -.0118, -.0168, -          | RUN_POOL
 38+M| .0348, -.0402, -.0595, -.0578, -.0529, -.0520, .4281, .4383, .4419, .4521, .5123, .5458, .5417,           | RUN_POOL
 38+M| .5236, .4993, .4336, .3281, .2114, .1674, .1399, .1069, .1010, 1.0001, .9432, .8956, .8770,                | RUN_POOL
 38+M| .9360, 1.0163, .9958, .9585, .9269, .8433, .6792, .5118, .4505, .4180, .3733, .3621, 1.1420,               | RUN_POOL
```

```
STMT                                      SOURCE                                                              CURRENT SCOPE
 38+M| 1.0618, .9138, .9109, .9560, 1.1198, 1.1137, 1.0934, 1.0748, .9967, .8401, .6500, .6007, .5621,   | RUN_POOL
 38+M| .5174, .5093);                                                                                    | RUN_POOL
 39+M| DECLARE SS_CD_MAT MATRIX(MAX_SS_ANGLE_INDEX, MAX_SS_M_INDEX) DOUBLE CONSTANT(.1047, .1194, .1533   | RUN_POOL
 39+M| , .1877, .2220, .2674, .2703, .2683, .2621, .2477, .2177, .1839, .1628, .1519, .1414, .1400,       | RUN_POOL
 39+M| .0610, .0625, .0683, .0805, .1052, .1477, .1522, .1557, .1563, .1547, .1410, .1135, .0922, .0894   | RUN_POOL
 39+M| , .0788, .0784, .0891, .0969, .1271, .1538, .1884, .2421, .2448, .2426, .2364, .2149, .1817,       | RUN_POOL
 39+M| .1347, .1125, .0990, .0849, .0837, .2823, .3578, .3870, .4093, .4518, .5227, .5200, .5051, .4867   | RUN_POOL
 39+M| , .4381, .3576, .2751, .2423, .2259, .2023, .1981, .4753, .5074, .4965, .5274, .5706, .6812,       | RUN_POOL
 39+M| .6630, .6673, .6501, .5928, .4976, .3917, .3587, .3373, .3102, .3063);                            | RUN_POOL
 40+M| DECLARE SS_M_VAL ARRAY(MAX_SS_M_INDEX) SCALAR DOUBLE CONSTANT(.25, .6, .8, .9, .95, 1.05, 1.1,     | RUN_POOL
 40+M| 1.2, 1.3, 1.5, 2., 3., 4., 5., 8., 10.);                                                          | RUN_POOL
 41+M| DECLARE SS_ANGLE_OF_ATTACK_VAL ARRAY(MAX_SS_ANGLE_INDEX) SCALAR DOUBLE CONSTANT(-.17453, 0.,       | RUN_POOL
 41+M| .17453, .34907, .43633);                                                                          | RUN_POOL
 42+M| DECLARE MAX_ALT INTEGER CONSTANT(50);                                                             | RUN_POOL
 43+M| DECLARE ATM_TEMP ARRAY(MAX_ALT + 1) SCALAR DOUBLE CONSTANT(288.150, 262.166, 236.215, 216.650,    | RUN_POOL
 43+M| 216.650, 216.650, 220.560, 224.527, 228.490, 239.282, 250.350, 261.403, 270.650, 270.650,         | RUN_POOL
 43+M| 263.628, 255.772, 243.202, 227.529, 211.876, 196.24, 180.65, 180.65, 180.65, 182.62, 198.45,      | RUN_POOL
 43+M| 210.02, 229.18, 247.85, 275.85, 313.01, 349.49, 423.90, 497.36, 570.09, 642.32, 714.22, 785.87,   | RUN_POOL
 43+M| 857.24, 918.94, 970.88, 1022.23, 1054.67, 1087.01, 1115.73, 1136.02, 1156.12, 1176.03, 1195.73,  | RUN_POOL
 43+M| 1211.66, 1223.88, 1235.95);                                                                       | RUN_POOL
 44+M| DECLARE ATM_DENS ARRAY(MAX_ALT + 1) SCALAR DOUBLE CONSTANT(1.0000, .66885, .42921, .25464,        | RUN_POOL
 44+M| .13589, .072579, .038317, .020470, .011065, .0059248, .0032618, .0018440, .0010749, .00065389,    | RUN_POOL
 44+M| .00040622, .00024973, .00015377, 9.3051E-05, 5.4361E-05, 3.050E-05, 1.632E-05, 7.807E-06,        | RUN_POOL
 44+M| 3.739E-06, 1.584E-06, 8.229E-07, 4.060E-07, 2.034E-07, 1.080E-07, 5.839E-08, 3.294E-08,           | RUN_POOL
 44+M| 1.968E-08, 1.171E-08, 7.533E-09, 5.165E-09, 3.714E-09, 2.770E-09, 2.129E-09, 1.676E-09,           | RUN_POOL
 44+M| 1.359E-09, 1.128E-09, 9.459E-10, 8.139E-10, 7.040E-10, 6.149E-10, 5.415E-10, 4.782E-10,           | RUN_POOL
 44+M| 4.236E-10, 3.762E-10, 3.359E-10, 3.014E-10, 2.708E-10);                                           | RUN_POOL
 45+M| DECLARE FEET_PER_METER SCALAR DOUBLE CONSTANT(3.280839895);                                        | RUN_POOL
```

```
STMT                                   SOURCE                                                    CURRENT SCOPE
 46+M|  DECLARE ALT_METER_INTERVAL SCALAR DOUBLE CONSTANT(4000.);                                 | RUN_POOL
 47+M|  DECLARE GROUND_RO SCALAR DOUBLE CONSTANT(2.38E-03);                                       | RUN_POOL
 48+M|  DECLARE NUM_CONSTANT_PARAMETERS INTEGER CONSTANT(10);                                     | RUN_POOL
 49+M|  DECLARE NUM_CONTROLS INTEGER CONSTANT(2);                                                 | RUN_POOL
 50+M|  DECLARE NUM_TRANS_PTS INTEGER CONSTANT(3);                                                | RUN_POOL
 51+M|  DECLARE NUM_CONSTRAINTS INTEGER CONSTANT(5);                                              | RUN_POOL
 52+M|  DECLARE FS_FIXED_PARAMETERS INTEGER CONSTANT(8);                                          | RUN_POOL
 53+M|  DECLARE ALT_FINAL SCALAR DOUBLE CONSTANT(3.96E05);                                        | RUN_POOL
 54+M|  DECLARE THETA_FINAL SCALAR DOUBLE CONSTANT(0.);                                           | RUN_POOL
 55+M|  DECLARE U_R_FINAL SCALAR DOUBLE CONSTANT(0.);                                             | RUN_POOL
 56+M|  DECLARE U_THETA_FINAL SCALAR DOUBLE CONSTANT(25734.879);                                  | RUN_POOL
 57+M|  DECLARE M1_FINAL SCALAR DOUBLE CONSTANT(0.);                                              | RUN_POOL
 58+M|  DECLARE M2_FINAL SCALAR DOUBLE CONSTANT(0.);                                              | RUN_POOL
 59+M|  DECLARE M3_FINAL SCALAR DOUBLE CONSTANT(0.);                                              | RUN_POOL
 60+M|  DECLARE NORM_TIME_STEP SCALAR DOUBLE CONSTANT(.5);                                        | RUN_POOL
 61+M|  DECLARE V MATRIX(NUM_CONSTANT_PARAMETERS + NUM_TRANS_PTS, NUM_CONSTANT_PARAMETERS +       | RUN_POOL
 61+M|  NUM_TRANS_PTS) DOUBLE INITIAL(1.1E-03, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  | RUN_POOL
 61+M|  2.3E03, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 2.E07, 0., 0., 0., 0., 0., 0., 0.,  | RUN_POOL
 61+M|  0., 0., 0., 0., 0., 0., 1.E05, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.6E07, 0.,  | RUN_POOL
 61+M|  0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 6., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  | RUN_POOL
 61+M|  0., 0., 2.1E-02, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 2.3E04, 0., 0., 0., 0., 0., | RUN_POOL
 61+M|  0., 0., 0., 0., 0., 0., 0., 0., 4.E08, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 4.E12 | RUN_POOL
 61+M|  , 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 5.E10, 0., 0., 0., 0., 0., 0., 0., 0., 0., | RUN_POOL
 61+M|  0., 0., 0., 0., 5.E11, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 5.E11);           | RUN_POOL
 62+M|  DECLARE PSI_WEIGHT MATRIX(NUM_CONSTRAINTS, NUM_CONSTRAINTS) DOUBLE CONSTANT(4., 0., 0., 0., 0., | RUN_POOL
 62+M|  0., 400., 0., 0., 0., 0., 0., 100., 0., 0., 0., 0., 0., 10000., 0., 0., 0., 0., 0., 100.);   | RUN_POOL
 63+M|  DECLARE STEP_DIM INTEGER CONSTANT(2000);                                                  | RUN_POOL
 64+M|  DECLARE MAX_BETA_CYCLES INTEGER CONSTANT(30);                                             | RUN_POOL
```

```
STMT                                   SOURCE                                                              CURRENT SCOPE
 65+M|  DECLARE PSI_START SCALAR DOUBLE CONSTANT(.1);                                                      | RUN_POOL
 66+M|  DECLARE MAX_CUTOFF_ITERATIONS INTEGER CONSTANT(30);                                                | RUN_POOL
 67+M|  DECLARE NEG_TIME_STEP ARRAY(NUM_TRANS_PTS) SCALAR DOUBLE INITIAL(.375, .375, .001);                 | RUN_POOL
 68+M|  DECLARE POS_TIME_STEP ARRAY(NUM_TRANS_PTS) SCALAR DOUBLE INITIAL(.125, .125, .499);                 | RUN_POOL
 69+M|  DECLARE W ARRAY((STEP_DIM + 2) / 2) MATRIX(NUM_CONTROLS, NUM_CONTROLS) DOUBLE;                      | RUN_POOL
 70+M|  DECLARE OMEGA_I_TIME ARRAY(NUM_TRANS_PTS) INTEGER;                                                 | RUN_POOL
 71+M|  DECLARE ITERATION INTEGER;                                                                         | RUN_POOL
 72+M|  DECLARE FIRST_PSI_MAG SCALAR DOUBLE;                                                               | RUN_POOL
 73+M|  DECLARE L_FILE INTEGER;                                                                            | RUN_POOL
 74+M|  DECLARE FIRST_ITERATION_FLAG BIT(1);                                                               | RUN_POOL
 75+M|  DECLARE P VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE;                                                  | RUN_POOL
 76+M|  DECLARE U_ACTIVE ARRAY(STEP_DIM + 1) VECTOR(NUM_CONTROLS) DOUBLE;                                   | RUN_POOL
 77+M|  DECLARE U_TIME_KEEP ARRAY((STEP_DIM + 2) / 2) SCALAR DOUBLE;                                       | RUN_POOL
 78+M|  DECLARE J_SCALE_FACTOR SCALAR DOUBLE INITIAL(3.);                                                  | RUN_POOL
 79+M|  DECLARE PSI_SCALE_FACTOR SCALAR DOUBLE INITIAL(2.);                                                | RUN_POOL
 80+M|  DECLARE DELTA_COST_CHECK SCALAR DOUBLE INITIAL(-50.);                                              | RUN_POOL
 81+M|  DECLARE DELTA_COST_SHRINK SCALAR DOUBLE INITIAL(.8);                                               | RUN_POOL
 82+M|  DECLARE THETA_DOT_INITIAL SCALAR DOUBLE INITIAL(9.05E-05);                                         | RUN_POOL
 83+M|  DECLARE CAP_Q SCALAR DOUBLE INITIAL(1.);                                                           | RUN_POOL
 84+M|  DECLARE Q_D SCALAR DOUBLE INITIAL(800.);                                                           | RUN_POOL
 85+M|  DECLARE CAP_CA SCALAR DOUBLE INITIAL(10.);                                                         | RUN_POOL
 86+M|  DECLARE G_D SCALAR DOUBLE INITIAL(2.0);                                                            | RUN_POOL
 87+M|  DECLARE STEP_SCALE_PSI SCALAR DOUBLE INITIAL(1.);                                                  | RUN_POOL
 88+M|  DECLARE STEP_SCALE_J SCALAR DOUBLE INITIAL(1.);                                                    | RUN_POOL
 89+M|  DECLARE PSI_COST_MIX SCALAR DOUBLE INITIAL(.04);                                                   | RUN_POOL
 90+M|  DECLARE MIN_PSI_THRESHOLD SCALAR DOUBLE INITIAL(.04);                                              | RUN_POOL
 91+M|  DECLARE MAX_STEP_I INTEGER INITIAL(4);                                                             | RUN_POOL
 92+M|  DECLARE MIN_COST_RQMT SCALAR DOUBLE INITIAL(.1);                                                   | RUN_POOL
```

```
STMT                              SOURCE                                              CURRENT SCOPE
  93+M|    DECLARE MIN_PSI_RQMT SCALAR DOUBLE INITIAL(2.E03);                         | RUN_POOL
  94+M|    DECLARE DJS_INIT SCALAR DOUBLE INITIAL(-4000.);                            | RUN_POOL
  95+M|    DECLARE MAX_ITERATIONS INTEGER INITIAL(1341);                              | RUN_POOL
  96+M|    DECLARE PSI_CHECK SCALAR DOUBLE INITIAL(4.E-06);                           | RUN_POOL
  97+M|    DECLARE R_CUTOFF_CHECK SCALAR DOUBLE INITIAL(2500.);                       | RUN_POOL
  98+M|    DECLARE X_STORE ARRAY(STEP_DIM + 1) VECTOR(NUM_STATES) DOUBLE;             | RUN_POOL
  99+M|    DECLARE U_OLD_TIME ARRAY(STEP_DIM + 1) SCALAR DOUBLE;                      | RUN_POOL
 100+M|    DECLARE FINAL_STEP INTEGER;                                                | RUN_POOL
 101+M| CLOSE;                                                                        | RUN_POOL
    +D|  VERSION  1                                                                   |
      *****END OF INCLUDED MEMBER, RVL 00, CATENATION NUMBER 1*****
```

```
STMT                               SOURCE                                      CURRENT SCOPE
102 M| AIR_BREATHER_OPTIMIZATION:                                             | AIR_BREATHER_OPTIMIZAT
102 M| PROCEDURE;                                                             | AIR_BREATHER_OPTIMIZAT
103 M|    DECLARE DYNAMIC_P0 SCALAR DOUBLE STATIC;                            | AIR_BREATHER_OPTIMIZAT
104 M|    DECLARE STAGE_SEP BIT(1) STATIC;                                    | AIR_BREATHER_OPTIMIZAT
105 M|    DECLARE H2_TANK_VOL SCALAR DOUBLE STATIC;                           | AIR_BREATHER_OPTIMIZAT
106 M|    DECLARE HC_TANK_VOL SCALAR DOUBLE STATIC;                           | AIR_BREATHER_OPTIMIZAT
107 M|    DECLARE TJ_FUEL_FLOW SCALAR DOUBLE STATIC;                          | AIR_BREATHER_OPTIMIZAT
108 M|    DECLARE SS_FUEL_FLOW SCALAR DOUBLE STATIC;                          | AIR_BREATHER_OPTIMIZAT
109 M|    DECLARE SCRJ_FUEL_FLOW SCALAR DOUBLE STATIC;                        | AIR_BREATHER_OPTIMIZAT
110 M|    DECLARE NET_R_FORCE SCALAR DOUBLE STATIC;                           | AIR_BREATHER_OPTIMIZAT
111 M|    DECLARE NET_THETA_FORCE SCALAR DOUBLE STATIC;                       | AIR_BREATHER_OPTIMIZAT
112 M|    DECLARE N_AERO SCALAR DOUBLE STATIC;                                | AIR_BREATHER_OPTIMIZAT
113 M|    DECLARE P_AERO SCALAR DOUBLE STATIC;                                | AIR_BREATHER_OPTIMIZAT
114 M|    DECLARE SCRAMJET_THRUST SCALAR DOUBLE STATIC;                       | AIR_BREATHER_OPTIMIZAT
115 M|    DECLARE TURBOJET_THRUST SCALAR DOUBLE STATIC;                       | AIR_BREATHER_OPTIMIZAT
116 M|    DECLARE ROCKET_THRUST SCALAR DOUBLE STATIC;                         | AIR_BREATHER_OPTIMIZAT
117 M|    DECLARE FIRST_STAGE_DRY_MASS SCALAR DOUBLE STATIC;                  | AIR_BREATHER_OPTIMIZAT
118 M|    DECLARE TURBOJET_POWER BIT(1) STATIC;                               | AIR_BREATHER_OPTIMIZAT
119 M|    DECLARE SCRAMJET_POWER BIT(1) STATIC;                               | AIR_BREATHER_OPTIMIZAT
120 M|    DECLARE SS_DRY_MASS SCALAR DOUBLE STATIC;                           | AIR_BREATHER_OPTIMIZAT
121 M|    DECLARE I_TIME INTEGER STATIC;                                      | AIR_BREATHER_OPTIMIZAT
122 M|    DECLARE RESHAPE_FLAG BIT(1) STATIC;                                 | AIR_BREATHER_OPTIMIZAT
123 M|    DECLARE RO_0 SCALAR DOUBLE STATIC;                                  | AIR_BREATHER_OPTIMIZAT
124 M|    DECLARE RO_2 SCALAR DOUBLE STATIC;                                  | AIR_BREATHER_OPTIMIZAT
125 M|    DECLARE U2 SCALAR DOUBLE STATIC;                                    | AIR_BREATHER_OPTIMIZAT
126 M|    DECLARE DRO_DR SCALAR DOUBLE STATIC;                                | AIR_BREATHER_OPTIMIZAT
127 M|    DECLARE CAP_PHI SCALAR DOUBLE STATIC;                               | AIR_BREATHER_OPTIMIZAT
128 M|    DECLARE SCRJ_MASS_CAPTURE_RATIO SCALAR DOUBLE STATIC;               | AIR_BREATHER_OPTIMIZAT
```

```
STMT                          SOURCE                                          CURRENT SCOPE
 129 M|  DECLARE SCRAMJET_MASS SCALAR DOUBLE STATIC;                         | AIR_BREATHER_OPTIMIZAT
 130 M|  DECLARE M0 SCALAR DOUBLE STATIC;                                    | AIR_BREATHER_OPTIMIZAT
 131 M|  DECLARE VEHICLE_ANGLE SCALAR DOUBLE STATIC;                         | AIR_BREATHER_OPTIMIZAT
 132 M|  DECLARE G_LOAD SCALAR DOUBLE STATIC;                                | AIR_BREATHER_OPTIMIZAT
 133 M|  DECLARE STATE_INTEGRATION_FLAG BIT(1) STATIC;                       | AIR_BREATHER_OPTIMIZAT
 134 M|  DECLARE L_TIME INTEGER STATIC;                                      | AIR_BREATHER_OPTIMIZAT
 135 M|  DECLARE M0_2 SCALAR DOUBLE STATIC;                                  | AIR_BREATHER_OPTIMIZAT
 136 M|  DECLARE T0 SCALAR DOUBLE STATIC;                                    | AIR_BREATHER_OPTIMIZAT
 137 M|  DECLARE M2 SCALAR DOUBLE STATIC;                                    | AIR_BREATHER_OPTIMIZAT
 138 M|  DECLARE BETA_NOSE_SHOCK SCALAR DOUBLE STATIC;                       | AIR_BREATHER_OPTIMIZAT
 139 M|  DECLARE THETA_NOSE SCALAR DOUBLE STATIC;                            | AIR_BREATHER_OPTIMIZAT
 140 M|  DECLARE TURBOJET_ISP SCALAR DOUBLE STATIC;                          | AIR_BREATHER_OPTIMIZAT
 141 M|  DECLARE SCRAMJET_ISP SCALAR DOUBLE STATIC;                          | AIR_BREATHER_OPTIMIZAT
 142 M|  DECLARE WING_AREA SCALAR DOUBLE STATIC;                             | AIR_BREATHER_OPTIMIZAT
 143 M|  DECLARE SS_PLANFORM_AREA SCALAR DOUBLE STATIC;                      | AIR_BREATHER_OPTIMIZAT
 144 M|  DECLARE LIFT SCALAR DOUBLE STATIC;                                  | AIR_BREATHER_OPTIMIZAT
 145 M|  DECLARE DRAG SCALAR DOUBLE STATIC;                                  | AIR_BREATHER_OPTIMIZAT
 146 M|  DECLARE MD_MASS SCALAR DOUBLE STATIC;                               | AIR_BREATHER_OPTIMIZAT
 147 M|  DECLARE NET_X_FORCE SCALAR DOUBLE STATIC;                           | AIR_BREATHER_OPTIMIZAT
 148 M|  DECLARE T2 SCALAR DOUBLE STATIC;                                    | AIR_BREATHER_OPTIMIZAT
 149 M|  DECLARE OBLIQUE_SHOCK_FLAG BIT(1) STATIC;                           | AIR_BREATHER_OPTIMIZAT
 150 M|  DECLARE NORMAL_SHOCK_FLAG BIT(1) STATIC;                            | AIR_BREATHER_OPTIMIZAT
 151 M|  DECLARE EXPANSION_FLAG BIT(1) STATIC;                               | AIR_BREATHER_OPTIMIZAT
 152 M|  DECLARE SUBSONIC_FLAG BIT(1) STATIC;                                | AIR_BREATHER_OPTIMIZAT
 153 M|  DECLARE DT0_DR SCALAR DOUBLE STATIC;                                | AIR_BREATHER_OPTIMIZAT
 154 M|  DECLARE DMCR_DM2 SCALAR DOUBLE STATIC;                              | AIR_BREATHER_OPTIMIZAT
 155 M|  DECLARE DCL1_DM0 SCALAR DOUBLE STATIC;                              | AIR_BREATHER_OPTIMIZAT
 156 M|  DECLARE DCL2_DM0 SCALAR DOUBLE STATIC;                              | AIR_BREATHER_OPTIMIZAT
```

```
STMT                            SOURCE                                      CURRENT SCOPE
 157 M|  DECLARE DCD1_DM0 SCALAR DOUBLE STATIC;                      | AIR_BREATHER_OPTIMIZAT
 158 M|  DECLARE DCD2_DM0 SCALAR DOUBLE STATIC;                      | AIR_BREATHER_OPTIMIZAT
 159 M|  DECLARE DCL1_DAR SCALAR DOUBLE STATIC;                      | AIR_BREATHER_OPTIMIZAT
 160 M|  DECLARE DCD1_DAR SCALAR DOUBLE STATIC;                      | AIR_BREATHER_OPTIMIZAT
 161 M|  DECLARE CL SCALAR DOUBLE STATIC;                            | AIR_BREATHER_OPTIMIZAT
 162 M|  DECLARE CD SCALAR DOUBLE STATIC;                            | AIR_BREATHER_OPTIMIZAT
 163 M|  DECLARE SS_CL SCALAR DOUBLE STATIC;                         | AIR_BREATHER_OPTIMIZAT
 164 M|  DECLARE SS_CD SCALAR DOUBLE STATIC;                         | AIR_BREATHER_OPTIMIZAT
 165 M|  DECLARE SIN_VEHICLE_ANGLE SCALAR DOUBLE STATIC;             | AIR_BREATHER_OPTIMIZAT
 166 M|  DECLARE COS_VEHICLE_ANGLE SCALAR DOUBLE STATIC;             | AIR_BREATHER_OPTIMIZAT
 167 M|  DECLARE G SCALAR DOUBLE STATIC;                             | AIR_BREATHER_OPTIMIZAT
 168 M|  DECLARE DCL2_DUA1 SCALAR DOUBLE STATIC;                     | AIR_BREATHER_OPTIMIZAT
 169 M|  DECLARE DCD2_DUA1 SCALAR DOUBLE STATIC;                     | AIR_BREATHER_OPTIMIZAT
 170 M|  DECLARE DCL_DALPHA SCALAR DOUBLE STATIC;                    | AIR_BREATHER_OPTIMIZAT
 171 M|  DECLARE DCD_DCL2 SCALAR DOUBLE STATIC;                      | AIR_BREATHER_OPTIMIZAT
```

```
STMT                                  SOURCE                                              CURRENT SCOPE
 172 M| MODEL_DRIVER:                                                                      | MODEL_DRIVER
 172 M| PROCEDURE;                                                                         | MODEL_DRIVER
     C|  THE THRUST IS ALWAYS ASSUMED IN PLANE WITH THE X BODY AXIS.                       | MODEL_DRIVER
 173 M|    DECLARE ROCKET_MAX_T SCALAR DOUBLE STATIC;                                      | MODEL_DRIVER
 174 M|    DECLARE NOSE_ANGLE SCALAR DOUBLE STATIC;                                        | MODEL_DRIVER
 175 M|    DECLARE W_SCRJ SCALAR DOUBLE STATIC;                                            | MODEL_DRIVER
 176 M|    DECLARE H_C SCALAR DOUBLE STATIC;                                               | MODEL_DRIVER
 177 M|    DECLARE H_C_TJ SCALAR DOUBLE STATIC;                                            | MODEL_DRIVER
 173 M|    DECLARE WING_SPAN SCALAR DOUBLE STATIC;                                         | MODEL_DRIVER
 179 M|    DECLARE DELTA_ANGLE SCALAR DOUBLE STATIC;                                       | MODEL_DRIVER
 180 M|    DECLARE HC_TANK_VOL_FRACTION SCALAR DOUBLE STATIC;                              | MODEL_DRIVER
 181 M|    DECLARE FIRST_STAGE_LENGTH SCALAR DOUBLE STATIC;                                | MODEL_DRIVER
 182 M|    DECLARE SS_MAX_FUEL_LOAD SCALAR DOUBLE STATIC;                                  | MODEL_DRIVER
 183 M|    DECLARE MAX_ROCKET_THRUST SCALAR DOUBLE STATIC;                                 | MODEL_DRIVER
 184 M|    DECLARE P0 SCALAR DOUBLE STATIC;                                                | MODEL_DRIVER
 185 M|    DECLARE U0 SCALAR DOUBLE STATIC;                                                | MODEL_DRIVER
 186 M|    DECLARE M1 SCALAR DOUBLE STATIC;                                                | MODEL_DRIVER
 187 M|    DECLARE P1 SCALAR DOUBLE STATIC;                                                | MODEL_DRIVER
 188 M|    DECLARE RO_1 SCALAR DOUBLE STATIC;                                              | MODEL_DRIVER
 189 M|    DECLARE T1 SCALAR DOUBLE STATIC;                                                | MODEL_DRIVER
 190 M|    DECLARE U1 SCALAR DOUBLE STATIC;                                                | MODEL_DRIVER
 191 M|    DECLARE F1 SCALAR DOUBLE AUTOMATIC;                                             | MODEL_DRIVER
 192 M|    DECLARE F2 SCALAR DOUBLE AUTOMATIC;                                             | MODEL_DRIVER
 193 M|    DECLARE F3 SCALAR DOUBLE AUTOMATIC;                                             | MODEL_DRIVER
 194 M|    DECLARE F4 SCALAR DOUBLE AUTOMATIC;                                             | MODEL_DRIVER
 195 M|    DECLARE F5 SCALAR DOUBLE AUTOMATIC;                                             | MODEL_DRIVER
 196 M|    DECLARE I_BETA INTEGER AUTOMATIC;                                               | MODEL_DRIVER
 197 M|    DECLARE R_NEW_BOUND SCALAR DOUBLE AUTOMATIC;                                    | MODEL_DRIVER
```

```
STMT                              SOURCE                                              CURRENT SCOPE
 198 M|  DECLARE BETA_UPPER_BOUND SCALAR DOUBLE AUTOMATIC;                           | MODEL_DRIVER
 199 M|  DECLARE BETA_LOWER_BOUND SCALAR DOUBLE AUTOMATIC;                           | MODEL_DRIVER
 200 M|  DECLARE L_BETA SCALAR DOUBLE AUTOMATIC;                                     | MODEL_DRIVER
 201 M|  DECLARE BETA_NEW_BOUND SCALAR DOUBLE AUTOMATIC;                             | MODEL_DRIVER
 202 M|  DECLARE M1_2 SCALAR DOUBLE AUTOMATIC;                                       | MODEL_DRIVER
 203 M|  DECLARE NU0 SCALAR DOUBLE AUTOMATIC;                                        | MODEL_DRIVER
 204 M|  DECLARE NU1 SCALAR DOUBLE AUTOMATIC;                                        | MODEL_DRIVER
 205 M|  DECLARE LOW_M_2 SCALAR DOUBLE AUTOMATIC;                                    | MODEL_DRIVER
 206 M|  DECLARE HIGH_M_2 SCALAR DOUBLE AUTOMATIC;                                   | MODEL_DRIVER
 207 M|  DECLARE M_2_FLAG BIT(1) AUTOMATIC;                                          | MODEL_DRIVER
 208 M|  DECLARE HIGH_NU SCALAR DOUBLE AUTOMATIC;                                    | MODEL_DRIVER
 209 M|  DECLARE MID_M_2 SCALAR DOUBLE AUTOMATIC;                                    | MODEL_DRIVER
 210 M|  DECLARE MID_NU SCALAR DOUBLE AUTOMATIC;                                     | MODEL_DRIVER
 211 M|  DECLARE EXTREMAL_ARRAY ARRAY(2) SCALAR DOUBLE AUTOMATIC;                    | MODEL_DRIVER
 212 M|  DECLARE SIN_BETA_THETA_MAX SCALAR DOUBLE AUTOMATIC;                         | MODEL_DRIVER
 213 M|  DECLARE BETA_THETA_MAX SCALAR DOUBLE AUTOMATIC;                             | MODEL_DRIVER
 214 M|  DECLARE TAN_THETA_MAX SCALAR DOUBLE AUTOMATIC;                              | MODEL_DRIVER
 215 M|  DECLARE THETA_MAX SCALAR DOUBLE AUTOMATIC;                                  | MODEL_DRIVER
 216 M|  DECLARE ANGLE_OF_ATTACK SCALAR DOUBLE STATIC;                               | MODEL_DRIVER
 217 M|  DECLARE R SCALAR DOUBLE STATIC;                                             | MODEL_DRIVER
 218 M|  DECLARE U_R SCALAR DOUBLE STATIC;                                           | MODEL_DRIVER
 219 M|  DECLARE U_THETA SCALAR DOUBLE STATIC;                                       | MODEL_DRIVER
 220 M|  DECLARE U_T_AIR SCALAR DOUBLE STATIC;                                       | MODEL_DRIVER
 221 M|  DECLARE FS_LIFT SCALAR DOUBLE STATIC;                                       | MODEL_DRIVER
 222 M|  DECLARE FS_DRAG SCALAR DOUBLE STATIC;                                       | MODEL_DRIVER
 223 M|  DECLARE SS_LIFT SCALAR DOUBLE STATIC;                                       | MODEL_DRIVER
 224 M|  DECLARE SS_DRAG SCALAR DOUBLE STATIC;                                       | MODEL_DRIVER
```

```
STMT                              SOURCE                                  CURRENT SCOPE
 225 M| VEHICLE:                                                          | VEHICLE
 225 M| PROCEDURE;                                                        | VEHICLE
 226 M|    DECLARE HIGH_M INTEGER AUTOMATIC;                              | VEHICLE
 227 M|    DECLARE LOW_M INTEGER AUTOMATIC;                               | VEHICLE
 228 M|    DECLARE I_SEARCH INTEGER AUTOMATIC;                            | VEHICLE
 229 M|    DECLARE HIGH_CD SCALAR DOUBLE AUTOMATIC;                       | VEHICLE
 230 M|    DECLARE LOW_CD SCALAR DOUBLE AUTOMATIC;                        | VEHICLE
 231 M|    DECLARE HIGH_CL SCALAR DOUBLE AUTOMATIC;                       | VEHICLE
 232 M|    DECLARE LOW_CL SCALAR DOUBLE AUTOMATIC;                        | VEHICLE
 233 M|    DECLARE M_FIND BIT(1) AUTOMATIC;                               | VEHICLE
```

```
STMT                                   SOURCE                                          CURRENT SCOPE
 234 M| FIRST_STAGE:                                                                  | FIRST_STAGE
 234 M| PROCEDURE;                                                                    | FIRST_STAGE
 235 M|    DECLARE CD0 SCALAR DOUBLE STATIC;                                          | FIRST_STAGE
 236 M|    DECLARE ASPECT_RATIO SCALAR DOUBLE STATIC;                                 | FIRST_STAGE
 237 M|    DECLARE TURBOJET_MASS SCALAR DOUBLE AUTOMATIC;                             | FIRST_STAGE
 238 M|    DECLARE FUSELAGE_SURFACE_AREA SCALAR DOUBLE AUTOMATIC;                     | FIRST_STAGE
 239 M|    DECLARE BODY_WING_MASS SCALAR DOUBLE AUTOMATIC;                            | FIRST_STAGE
 240 M|    DECLARE HC_TANK_MASS SCALAR DOUBLE AUTOMATIC;                              | FIRST_STAGE
 241 M|    DECLARE H2_TANK_MASS SCALAR DOUBLE AUTOMATIC;                              | FIRST_STAGE
 242 M|    DECLARE FIND_FLAG BIT(1) AUTOMATIC;                                        | FIRST_STAGE
 243 M|    DECLARE I_ZLD INTEGER AUTOMATIC;                                           | FIRST_STAGE
 244 M|    DECLARE LOW_CD0 SCALAR DOUBLE AUTOMATIC;                                   | FIRST_STAGE
 245 M|    DECLARE LOW_M_S SCALAR DOUBLE AUTOMATIC;                                   | FIRST_STAGE
 246 M|    DECLARE HIGH_CD0 SCALAR DOUBLE AUTOMATIC;                                  | FIRST_STAGE
 247 M|    DECLARE HIGH_M_S SCALAR DOUBLE AUTOMATIC;                                  | FIRST_STAGE
 248 M|    DECLARE I_MCR INTEGER AUTOMATIC;                                           | FIRST_STAGE
 249 M|    DECLARE MCR_FLAG BIT(1) AUTOMATIC;                                         | FIRST_STAGE
 250 M|    DECLARE A_FIND BIT(1) AUTOMATIC;                                           | FIRST_STAGE
 251 M|    DECLARE LOW_A INTEGER AUTOMATIC;                                           | FIRST_STAGE
 252 M|    DECLARE HIGH_A INTEGER AUTOMATIC;                                          | FIRST_STAGE
 253 M|    DECLARE DRY_TANK_VOLUME SCALAR DOUBLE AUTOMATIC;                           | FIRST_STAGE
 254 M|    DECLARE FUSELAGE_VOLUME SCALAR DOUBLE AUTOMATIC;                           | FIRST_STAGE
 255 M|    DECLARE DDCD_DCL2_DAR SCALAR DOUBLE AUTOMATIC;                             | FIRST_STAGE
 256 M|    DECLARE DDCD_DCL2_DM0 SCALAR DOUBLE AUTOMATIC;                             | FIRST_STAGE
 257 M|    DECLARE DCD0_DM0 SCALAR DOUBLE AUTOMATIC;                                  | FIRST_STAGE
     E|                                                                               |
 258 M|    IF RESHAPE_FLAG = ON THEN                                                  | FIRST_STAGE
 259 M|       DO;                                                                     | FIRST_STAGE
```

```
STMT                                   SOURCE                                              CURRENT SCOPE

     C| THIS BRANCH COMPUTES GEOMETRY DEPENDENT FIRST STAGE PROPERTIES                      | FIRST_STAGE
     C| ALL UNITS ARE MASS=SLUGS,LENGTH=FEET,TIME=SECONDS                                   | FIRST_STAGE

 260 M| 1         SCRAMJET_MASS = (15.2 H_C) - (4.6 / H_C);                                 | FIRST_STAGE

 261 M| 1         IF SCRAMJET_MASS < MIN_SCRJ_MASS_PER_FT THEN                              | FIRST_STAGE

 262 M| 1            SCRAMJET_MASS = MIN_SCRJ_MASS_PER_FT;                                  | FIRST_STAGE

 263 M| 1         SCRAMJET_MASS = SCRAMJET_MASS W_SCRJ;                                     | FIRST_STAGE

     C| FOR THE BASIS OF THE SCRAMJET MASS MODEL                                            | FIRST_STAGE
     C| SEE FIG. 16 AND TABLE I OF 'SCRAMJET PERFORMANCE CHARACTERISTICS'                   | FIRST_STAGE
     C| THE LOWER BOUND MASS IS BASED ON NON-PROPULSIVE SYSTEMS REQUIRED                    | FIRST_STAGE
     C| TO OPERATE THE SCRAMJET                                                             | FIRST_STAGE

 264 M| 1         TURBOJET_MASS = 5. W_SCRJ H_C_TJ;                                         | FIRST_STAGE

     C| THE MODEL FOR THE TURBOJET MASS IS                                                  | FIRST_STAGE
     C| BASED ON GE CF6-6 DATA (JP-121 NOTES)                                               | FIRST_STAGE
     C| SCALED BY 2/3 FOR ADVANCED ENGINES                                                  | FIRST_STAGE

 265 M| 1         WING_AREA = (WING_SPAN WING_SPAN) / (4. TAN(DELTA_ANGLE));                | FIRST_STAGE

     C| TO KEEP THE NUMBER OF PARAMETERS TO A MINIMUM, THE ENGINE WIDTH IS                  | FIRST_STAGE
     C| ASSUMED EQUAL TO THE FUSELAGE WIDTH                                                 | FIRST_STAGE

 266 M| 1         FUSELAGE_VOLUME = (FIRST_STAGE_LENGTH FIRST_STAGE_LENGTH W_SCRJ SIN(NOSE_ANGLE) SIN(       | FIRST_STAGE

 266 M| 1         NOZZLE_ANGLE)) / (SIN(NOSE_ANGLE + NOZZLE_ANGLE) 2.);                     | FIRST_STAGE

 267 M| 1         FUSELAGE_SURFACE_AREA = (FIRST_STAGE_LENGTH (1. + ((SIN(NOSE_ANGLE) + SIN(NOZZLE_ANGLE)) / | FIRST_STAGE

 267 M| 1         SIN(NOSE_ANGLE + NOZZLE_ANGLE))) W_SCRJ) + ((2. FUSELAGE_VOLUME) / W_SCRJ);               | FIRST_STAGE

 268 M| 1         BODY_WING_MASS = .25 (FUSELAGE_SURFACE_AREA + WING_AREA);                 | FIRST_STAGE

     C| THE BASIS OF THE BODY AND WING WEIGHT MODEL IS                                      | FIRST_STAGE
     C| EQ. 2.60 MARTIN MS THESIS                                                           | FIRST_STAGE

 269 M| 1         DRY_TANK_VOLUME = .8 FUSELAGE_VOLUME;                                     | FIRST_STAGE

 270 M| 1         HC_TANK_VOL = DRY_TANK_VOLUME HC_TANK_VOL_FRACTION;                       | FIRST_STAGE

 271 M| 1         HC_TANK_MASS = .01 HYDROCARBON_DENSITY HC_TANK_VOL;                       | FIRST_STAGE

     C| THE HYDROCARBON TANK MASS IS                                                        | FIRST_STAGE
     C| BASED ON SCALED DOWN H2 TANK VALUE SINCE HC IS NOT CRYOGENIC                        | FIRST_STAGE

 272 M| 1         H2_TANK_VOL = DRY_TANK_VOLUME (1. - HC_TANK_VOL_FRACTION);                | FIRST_STAGE

 273 M| 1         H2_TANK_MASS = .25 H2_DENSITY H2_TANK_VOL;                                | FIRST_STAGE

     C| THE HYDROGEN TANK MASS IS                                                           | FIRST_STAGE
     C| BASED ON H2/O2 PHI=1.6 TANKS=.05 FUEL WEIGHT SCALED TO H2 ONLY                      | FIRST_STAGE
```

```
STMT                                          SOURCE                                                          CURRENT SCOPE

274 M| 1         FIRST_STAGE_DRY_MASS = SCRAMJET_MASS + TURBOJET_MASS + BODY_WING_MASS + HC_TANK_MASS +     | FIRST_STAGE

274 M| 1         H2_TANK_MASS;                                                                             | FIRST_STAGE

275 M|         END;                                                                                        | FIRST_STAGE

276 M|      ELSE                                                                                           | FIRST_STAGE

276 M|         DO;                                                                                         | FIRST_STAGE

    C|  THIS BRANCH COMPUTES MACH NUMBER, ALTITUDE, AND ANGLE OF ATTACK                                    | FIRST_STAGE
    C|  DEPENDENT FIRST STAGE PROPERTIES                                                                   | FIRST_STAGE

    E|                                  .                                                                  |
277 M| 1         IF ((M2 > 1.) AI:D (SCRAMJET_FC:ER = ON)) THEN                                            | FIRST_STAGE

278 M| 1            DO;                                                                                    | FIRST_STAGE

279 M| 2               IF M2 > MCR_M                 THEN                                                  | FIRST_STAGE
    S|                            MCR_MAX_INDEX                                                            |

280 M| 2                  SCRJ_MASS_CAPTURE_RATIO = MCR_C             ;                                    | FIRST_STAGE
    S|                                                 MCR_MAX_INDEX                                       |

281 M| 2               ELSE                                                                                | FIRST_STAGE

281 M| 2                  DO;                                                                              | FIRST_STAGE

    E|                       .                                                                             |
282 M| 3                     MCR_FLAG = ON;                                                                | FIRST_STAGE

    E|                                                 .                                                   |
283 M| 3                     DO FOR I_MCR = 2 TO MCR_MAX_INDEX WHILE MCR_FLAG = ON;                        | FIRST_STAGE

284 M| 4                        IF ((M2 < MCR_M      ) OR (M2 = MCR_M      )) THEN                         | FIRST_STAGE
    S|                                        I_MCR                  I_MCR                                 |

285 M| 4                           DO;                                                                     | FIRST_STAGE

    E|                                .                                                                    |
286 M| 5                              MCR_FLAG = OFF;                                                      | FIRST_STAGE

287 M| 5                              HIGH_M = I_MCR;                                                      | FIRST_STAGE

288 M| 4                           END;                                                                    | FIRST_STAGE

289 M| 3                     END;                                                                          | FIRST_STAGE

290 M| 3                     LOW_M = HIGH_M - 1;                                                           | FIRST_STAGE

291 M| 3                     SCRJ_MASS_CAPTURE_RATIO = MCR_C      + (((M2 - MCR_M      ) (MCR_C       -    | FIRST_STAGE
    S|                                                    LOW_M                  LOW_M        HIGH_M       |

291 M| 3                     MCR_C      )) / (MCR_M       - MCR_M      ));                                 | FIRST_STAGE
    S|                          LOW_M            HIGH_M       LOW_M                                        |
```

```
STMT                                        SOURCE                                                        CURRENT SCOPE

292 M| 3                DMCR_DM2 = (MCR_C        - MCR_C      ) / (MCR_M        - MCR_M      );          | FIRST_STAGE
    S|                                 HIGH_M         LOW_M             HIGH_M          LOW_M            |

293 M| 2              END;                                                                               | FIRST_STAGE

294 M| 2            IF SCRJ_MASS_CAPTURE_RATIO < 0. THEN                                                 | FIRST_STAGE

295 M| 2              DO;                                                                                | FIRST_STAGE

296 M| 3                SCRJ_MASS_CAPTURE_RATIO = 0.;                                                    | FIRST_STAGE

297 M| 3                DMCR_DM2 = 0.;                                                                   | FIRST_STAGE

298 M| 2              END;                                                                               | FIRST_STAGE

299 M| 1          END;                                                                                   | FIRST_STAGE

300 M| 1        ELSE                                                                                     | FIRST_STAGE

300 M| 1          SCRJ_MASS_CAPTURE_RATIO = 0.;                                                          | FIRST_STAGE

301 M| 1        TURBOJET_ISP = 3800. - (300. M2) - (100. M2 M2);                                         | FIRST_STAGE

    E|                                 1.6                 .52                                       |
302 M| 1        SCRAMJET_ISP = 15000. (M2    ) EXP(-1.73 (M2    ));                                    | FIRST_STAGE

    C|  THE INSTALLED SPECIFIC IMPULSE EQUATIONS GIVEN ABOVE ARE BASED ON                                | FIRST_STAGE
    C|  A CURVE FIT.                                                                                     | FIRST_STAGE
    C|  SEE JONES AND HUEER 'AIRFRAME INTEGRATED PROPULSION SYSTEM FOR                                   | FIRST_STAGE
    C|  HYFERSONIC CRUISE VEHICLES'                                                                      | FIRST_STAGE

    E|                   .                                                                               |
303 M| 1        IF TURBOJET_POWER = OFF THEN                                                             | FIRST_STAGE

304 M| 1          TURBOJET_THRUST = 0.;                                                                  | FIRST_STAGE

305 M| 1        ELSE                                                                                     | FIRST_STAGE

305 M| 1          TURBOJET_THRUST = RO_2 U2 W_SCRJ H_C_TJ TJ_MAX_FUEL_AIR_RATIO G0 TURBOJET_ISP CAP_PHI;  | FIRST_STAGE

    E|                   .                                                                               |
306 M| 1        IF SCRAMJET_POWER = OFF THEN                                                             | FIRST_STAGE

307 M| 1          SCRAMJET_THRUST = 0.;                                                                  | FIRST_STAGE

308 M| 1        ELSE                                                                                     | FIRST_STAGE

308 M| 1          SCRAMJET_THRUST = RO_2 U2 W_SCRJ H_C SCRJ_MASS_CAPTURE_RATIO SCRJ_MAX_FUEL_AIR_RATIO G0 | FIRST_STAGE

308 M| 1          SCRAMJET_ISP CAP_PHI;                                                                  | FIRST_STAGE

    C|  THRUST IS COMPUTED FROM AN ASSUMED STOICHIOMETRIC FUEL/AIR MIXTURE                               | FIRST_STAGE
    C|  SCALED TO ACTUAL FUEL FLOW RATE                                                                  | FIRST_STAGE

309 M| 1        ASPECT_RATIO = (WING_SPAN WING_SPAN) / WING_AREA;                                        | FIRST_STAGE
```

```
STMT                                        SOURCE                                              CURRENT SCOPE

 310 M| 1      IF ((ASPECT_RATIO = FS_A_VAL                ) OR (ASPECT_RATIO > FS_A_VAL                 ))  | FIRST_STAGE
     S|                             MAX_FS_AR_INDEX                              MAX_FS_AR_INDEX              |
 310 M| 1      THEN                                                                                          | FIRST_STAGE
 311 M| 1         HIGH_A = MAX_FS_AR_INDEX;                                                                  | FIRST_STAGE
 312 M| 1      ELSE                                                                                          | FIRST_STAGE
 312 M| 1         DO;                                                                                        | FIRST_STAGE
     E|               .                                                                                       |
 313 M| 2            A_FIND = ON;                                                                            | FIRST_STAGE
     E|                                                            .                                          |
 314 M| 2            DO FOR I_SEARCH = 2 TO MAX_FS_AR_INDEX WHILE A_FIND = ON;                               | FIRST_STAGE
 315 M| 3               IF FS_A_VAL          > ASPECT_RATIO THEN                                             | FIRST_STAGE
     S|                           I_SEARCH                                                                   |
 316 M| 3                  DO;                                                                               | FIRST_STAGE
 317 M| 4                     HIGH_A = I_SEARCH;                                                             | FIRST_STAGE
     E|                          .                                                                           |
 318 M| 4                     A_FIND = OFF;                                                                  | FIRST_STAGE
 319 M| 3                  END;                                                                              | FIRST_STAGE
 320 M| 2            END;                                                                                    | FIRST_STAGE
 321 M| 1         END;                                                                                       | FIRST_STAGE
 322 M| 1      LOW_A = HIGH_A - 1;                                                                           | FIRST_STAGE
 323 M| 1      IF ((M0 = FS_M_VAL                ) OR (M0 > FS_M_VAL                )) THEN                  | FIRST_STAGE
     S|                        MAX_FS_M_INDEX                      MAX_FS_M_INDEX                            |
 324 M| 1         HIGH_M = MAX_FS_M_INDEX;                                                                   | FIRST_STAGE
 325 M| 1      ELSE                                                                                          | FIRST_STAGE
 325 M| 1         DO;                                                                                        | FIRST_STAGE
     E|               .                                                                                       |
 326 M| 2            M_FIND = ON;                                                                            | FIRST_STAGE
     E|                                                            .                                          |
 327 M| 2            DO FOR I_SEARCH = 2 TO MAX_FS_M_INDEX WHILE M_FIND = ON;                                | FIRST_STAGE
 328 M| 3               IF FS_M_VAL          > M0 THEN                                                       | FIRST_STAGE
     S|                           I_SEARCH                                                                   |
 329 M| 3                  DO;                                                                               | FIRST_STAGE
```

```
STMT                                    SOURCE                                                  CURRENT SCOPE
 330 M| 4               HIGH_M = I_SEARCH;                                                       | FIRST_STAGE
     E|                 .                                                                        |
 331 M| 4               M_FIND = OFF;                                                            | FIRST_STAGE
 332 M| 3            END;                                                                        | FIRST_STAGE
 333 M| 2         END;                                                                           | FIRST_STAGE
 334 M| 1       END;                                                                             | FIRST_STAGE
 335 M| 1      LOW_M = HIGH_M - 1;                                                               | FIRST_STAGE
 336 M| 1      F1 = (M0 - FS_M_VAL      ) / (FS_M_VAL       - FS_M_VAL     );                    | FIRST_STAGE
     S|                   LOW_M                HIGH_M           LOW_M                            |
 337 M| 1      LOW_CL = FS_CL_MAT            + (F1 (FS_CL_MAT             - FS_CL_MAT            ));  | FIRST_STAGE
     S|                  LOW_A,LOW_M                  LOW_A,HIGH_M              LOW_A,LOW_M           |
 338 M| 1      LOW_CD = FS_CD_MAT            + (F1 (FS_CD_MAT             - FS_CD_MAT            ));  | FIRST_STAGE
     S|                  LOW_A,LOW_M                  LOW_A,HIGH_M              LOW_A,LOW_M           |
 339 M| 1      HIGH_CL = FS_CL_MAT            + (F1 (FS_CL_MAT              - FS_CL_MAT             )); | FIRST_STAGE
     S|                   HIGH_A,LOW_M                  HIGH_A,HIGH_M              HIGH_A,LOW_M        |
 340 M| 1      HIGH_CD = FS_CD_MAT            + (F1 (FS_CD_MAT              - FS_CD_MAT             )); | FIRST_STAGE
     S|                   HIGH_A,LOW_M                  HIGH_A,HIGH_M              HIGH_A,LOW_M        |
 341 M| 1      F1 = (ASPECT_RATIO - FS_A_VAL     ) / (FS_A_VAL       - FS_A_VAL     );             | FIRST_STAGE
     S|                               LOW_A              HIGH_A           LOW_A                  |
 342 M| 1      IF ASPECT_RATIO > FS_A_VAL                THEN                                    | FIRST_STAGE
     S|                            MAX_FS_AR_INDEX                                               |
 343 M| 1         DCD_DCL2 = HIGH_CD;                                                            | FIRST_STAGE
 344 M| 1      ELSE                                                                              | FIRST_STAGE
 344 M| 1         DCD_DCL2 = LOW_CD + (F1 (HIGH_CD - LOW_CD));                                   | FIRST_STAGE
 345 M| 1      DCL_DALPHA = LOW_CL + (F1 (HIGH_CL - LOW_CL));                                    | FIRST_STAGE
 346 M| 1      CL = DCL_DALPHA ANGLE_OF_ATTACK DEGREES_PER_RADIAN;                               | FIRST_STAGE
 347 M| 1      DCL1_DAR = ((HIGH_CL - LOW_CL) / (FS_A_VAL       - FS_A_VAL     )) (ANGLE_OF_ATTACK | FIRST_STAGE
     S|                                           HIGH_A           LOW_A                         |
 347 M| 1      DEGREES_PER_RADIAN);                                                              | FIRST_STAGE
 348 M| 1      IF ASPECT_RATIO > FS_A_VAL                THEN                                    | FIRST_STAGE
     S|                            MAX_FS_AR_INDEX                                               |
 349 M| 1         DDCD_DCL2_DAR = 0.;                                                            | FIRST_STAGE
 350 M| 1      ELSE                                                                              | FIRST_STAGE
```

```
STMT                                        SOURCE                                                      CURRENT SCOPE

 350 M| 1         DDCD_DCL2_DAR = (HIGH_CD - LOW_CD) / (FS_A_VAL       - FS_A_VAL      );              | FIRST_STAGE
     S|                                                       HIGH_A            LOW_A                  |

 351 M| 1      DCD1_DAR = (DDCD_DCL2_DAR CL CL) + (2. DCD_DCL2 CL DCL1_DAR);                            | FIRST_STAGE

 352 M| 1      LOW_CL = FS_CL_MAT           + (F1 (FS_CL_MAT              - FS_CL_MAT            ));    | FIRST_STAGE
     S|                  LOW_A,LCW_M                  HIGH_A,LOW_M             LOW_A,LOW_M             |

 353 M| 1      LOW_CD = FS_CD_MAT           + (F1 (FS_CD_MAT              - FS_CD_MAT            ));    | FIRST_STAGE
     S|                  LOW_A,LOW_M                  HIGH_A,LOW_M             LOW_A,LOW_M             |

 354 M| 1      HIGH_CL = FS_CL_MAT            + (F1 (FS_CL_MAT                - FS_CL_MAT             ));  | FIRST_STAGE
     S|                   LOW_A,HIGH_M                  HIGH_A,HIGH_M              LOW_A,HIGH_M        |

 355 M| 1      HIGH_CD = FS_CD_MAT            + (F1 (FS_CD_MAT                - FS_CD_MAT             ));  | FIRST_STAGE
     S|                   LOW_A,HIGH_M                  HIGH_A,HIGH_M              LOW_A,HIGH_M        |

 356 M| 1      DCL1_DM0 = ((HIGH_CL - LOW_CL) / (FS_M_VAL       - FS_M_VAL      )) (ANGLE_OF_ATTACK     | FIRST_STAGE
     S|                                               HIGH_M           LOW_M                           |

 356 M| 1      DEGREES_PER_RADIAN);                                                                     | FIRST_STAGE

 357 M| 1      DDCD_DCL2_DM0 = (HIGH_CD - LOW_CD) / (FS_M_VAL       - FS_M_VAL      );                  | FIRST_STAGE
     S|                                                   HIGH_M           LOW_M                       |

 358 M| 1      DCD0_DM0 = 0.;                                                                           | FIRST_STAGE

 359 M| 1      IF ((M0 < M_ZLD ) OR (M0 = M_ZLD )) THEN                                                 | FIRST_STAGE
     S|                     1                 1                                                        |

 360 M| 1         CD0 = ZLD ;                                                                           | FIRST_STAGE
     S|                  1                                                                             |

 361 M| 1      ELSE                                                                                     | FIRST_STAGE

 361 M| 1         DO;                                                                                   | FIRST_STAGE

 362 M| 2            IF M0 ` M_ZLD              THEN                                                    | FIRST_STAGE
     S|                       MAX_ZLD_INDEX                                                            |

 363 M| 2               CD0 = ZLD              ;                                                        | FIRST_STAGE
     S|                      MAX_ZLD_INDEX                                                             |

 364 M| 2            ELSE                                                                               | FIRST_STAGE

 364 M| 2               DO;                                                                             | FIRST_STAGE

     E|                    .                                                                           |
 365 M| 3                  FIND_FLAG = OFF;                                                             | FIRST_STAGE

     E|                                                     .                                          |
 366 M| 3                  DO FOR I_ZLD = 2 TO MAX_ZLD_INDEX WHILE FIND_FLAG = OFF;                     | FIRST_STAGE

 367 M| 4                     IF ((M0 < M_ZLD      ) OR (M0 = M_ZLD      )) THEN                         | FIRST_STAGE
     S|                                   I_ZLD                I_ZLD                                   |
```

```
STMT                               SOURCE                                           CURRENT SCOPE
 368 M| 4              DO;                                                           | FIRST_STAGE
 369 M| 5                LOW_CDO = ZLD        ;                                      | FIRST_STAGE
     S|                           I_ZLD-1                                            |
 370 M| 5                LOW_M_S = M_ZLD        ;                                    | FIRST_STAGE
     S|                             I_ZLD-1                                          |
 371 M| 5                HIGH_CDO = ZLD     ;                                        | FIRST_STAGE
     S|                            I_ZLD                                             |
 372 M| 5                HIGH_M_S = M_ZLD     ;                                      | FIRST_STAGE
     S|                              I_ZLD                                           |
     E|                        .                                                     |
 373 M| 5                FIND_FLAG = ON;                                             | FIRST_STAGE
 374 M| 4              END;                                                          | FIRST_STAGE
 375 M| 3          END;                                                              | FIRST_STAGE
 376 M| 3          CDO = LOW_CDO + (((MO - LOW_M_S) (HIGH_CDO - LOW_CDO)) / (HIGH_M_S - LOW_M_S)) | FIRST_STAGE
 376 M| 3          ;                                                                 | FIRST_STAGE
 377 M| 3          DCDO_DMO = (HIGH_CDO - LOW_CDO) / (HIGH_M_S - LOW_M_S);           | FIRST_STAGE
 378 M| 2        END;                                                                | FIRST_STAGE
 379 M| 1      END;                                                                  | FIRST_STAGE
 380 M| 1    CD = (DCD_DCL2 CL CL) + CDO;                                            | FIRST_STAGE
 381 M| 1    DCD1_DMO = (DDCD_DCL2_DMO CL CL) + DCDO_DMO + (2. DCD_DCL2 CL DCL1_DMO); | FIRST_STAGE
 382 M| 1    F1 = SQRT(L_D_SCALE_FACTOR);                                            | FIRST_STAGE
 383 M| 1    CD = CD / F1;                                                           | FIRST_STAGE
 384 M| 1    CL = CL F1;                                                             | FIRST_STAGE
 385 M| 1    DCD1_DMO = DCD1_DMO / F1;                                               | FIRST_STAGE
 386 M| 1    DCL1_DMO = DCL1_DMO F1;                                                 | FIRST_STAGE
 387 M| 1    DCD1_DAR = DCD1_DAR / F1;                                               | FIRST_STAGE
 388 M| 1    DCL1_DAR = DCL1_DAR F1;                                                 | FIRST_STAGE
     E|                           3                                                  |
 389 M| 1    DCD_DCL2 = DCD_DCL2 / (F1 );                                            | FIRST_STAGE
 390 M| 1    DCL_DALPHA = DCL_DALPHA F1;                                             | FIRST_STAGE
     C|  F1 IS A LIFT AND DRAG SCALING FACTOR TO PERMIT USING EXISTING DATA          | FIRST_STAGE
```

```
STMT                                    SOURCE                                        CURRENT SCOPE

    C| WHILE ASSUMING IMPROVED WING DESIGN                                            | FIRST_STAGE
 391 M| 1        FS_LIFT = CL WING_AREA DYNAMIC_P0;                                   | FIRST_STAGE
 392 M| 1        FS_DRAG = CD WING_AREA DYNAMIC_P0;                                   | FIRST_STAGE
 393 M| 1        IF FS_DRAG < 0. THEN                                                 | FIRST_STAGE
 394 M| 1           DO;                                                               | FIRST_STAGE
 395 M| 2              FS_DRAG = 0.;                                                  | FIRST_STAGE
 396 M| 2              CD = 0.;                                                       | FIRST_STAGE
 397 M| 2              DCD1_DM0 = 0.;                                                 | FIRST_STAGE
 398 M| 2              DCD1_DAR = 0.;                                                 | FIRST_STAGE
 399 M| 2              DCD_DCL2 = 0.;                                                 | FIRST_STAGE
 400 M| 1           END;                                                              | FIRST_STAGE
 401 M|       END;                                                                    | FIRST_STAGE
 402 M| CLOSE FIRST_STAGE;                                                             | FIRST_STAGE

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
    MIN_SCRJ_MASS_PER_FT, NOZZLE_ANGLE, HYDROCARBON_DENSITY, H2_DENSITY, MCR_MAX_INDEX, MCR_M, MCR_C, TJ_MAX_FUEL_AIR_RATIO, G0
    SCRJ_MAX_FUEL_AIR_RATIO, MAX_FS_AR_INDEX, FS_A_VAL, MAX_FS_M_INDEX, FS_M_VAL, FS_CL_MAT, FS_CD_MAT, DEGREES_PER_RADIAN, M_ZLD
    ZLD, MAX_ZLD_INDEX, L_D_SCALE_FACTOR

OUTER VARIABLES USED
    RESHAPE_FLAG, SCRAMJET_MASS*, H_C, SCRAMJET_MASS, W_SCRJ, H_C_TJ, WING_AREA*, WING_SPAN, DELTA_ANGLE, FIRST_STAGE_LENGTH
    NOSE_ANGLE, WING_AREA, HC_TANK_VOL*, HC_TANK_VOL_FRACTION, HC_TANK_VOL, H2_TANK_VOL*, H2_TANK_VOL, FIRST_STAGE_DRY_MASS*, M2
    SCRAMJET_POWER, SCRJ_MASS_CAPTURE_RATIO*, HIGH_M*, LOW_M*, HIGH_M, LOW_M, DMCR_DMC*, SCRJ_MASS_CAPTURE_RATIO, TURBOJET_ISP*
    SCRAMJET_ISP*, TURBOJET_POWER, TURBOJET_THRUST*, RO_2, U2, TURBOJET_ISP, CAP_PHI, SCRAMJET_THRUST*, SCRAMJET_ISP, I_SEARCH*
    I_SEARCH, M0, M_FIND*, M_FIND, F1*, LOW_CL*, F1, LOW_CD*, HIGH_CL*, HIGH_CD*, DCD_DCL2*, HIGH_CD, LOW_CD, DCL_DALFHA*, LOW_CL
    HIGH_CL, CL*, DCL_DALFHA, ANGLE_OF_ATTACK, DCL1_DAR*, DCD1_DAR*, CL, DCD_DCL2, DCL1_DAR, DCL1_DM0*, CD*, DCD1_DM0*, DCL1_DM0, CD
    DCD1_DM0, DCD1_DAR, FS_LIFT*, DYNAMIC_P0, FS_DRAG*, FS_DRAG
```

```
STMT                                   SOURCE                                                    CURRENT SCOPE

 403 M| SCND_STAGE:                                                                              | SCND_STAGE

 403 M| PROCEDURE;                                                                               | SCND_STAGE

 404 M|    DECLARE ANG_FIND BIT(1) AUTOMATIC;                                                    | SCND_STAGE

 405 M|    DECLARE LOW_ANGLE INTEGER AUTOMATIC;                                                  | SCND_STAGE

 406 M|    DECLARE HIGH_ANGLE INTEGER AUTOMATIC;                                                 | SCND_STAGE

 407 M|    DECLARE MO_FLAG BIT(1) AUTOMATIC;                                                     | SCND_STAGE

     E|               .                                                                          |
 408 M|    IF RESHAPE_FLAG = ON THEN                                                             | SCND_STAGE

 409 M|       DO;                                                                                | SCND_STAGE

     C|  ALL UNITS ARE MASS=SLUGS,LENGTH=FEET,TIME=SECONDS                                       | SCND_STAGE

 410 M| 1       SS_DRY_MASS = DELIVERED_MASS + (.04 SS_MAX_FUEL_LOAD) + (MAX_ROCKET_THRUST / 3220.); | SCND_STAGE

     C|  THE ROCKET PROPELLANT TANK MASS MODEL IS                                                | SCND_STAGE
     C|  BASED ON TANK MASS=.04 FUEL MASS                                                        | SCND_STAGE

     E|                                                                         .6667            |
 411 M| 1       SS_PLANFORM_AREA = DELIVERED_PLANFORM_AREA ((1. + (.0000334 SS_MAX_FUEL_LOAD))      ); | SCND_STAGE

     C|  THE PLANFORM AREA IS DERIVED FROM A VOLUME/AREA SCALING LAW                             | SCND_STAGE
     C|  BASED ON FHI=1.6                                                                        | SCND_STAGE
     C|  AVAILABLE VOLUME FOR FUEL=.8 TANK VOLUME                                                | SCND_STAGE
     C|  VOLUME DELIVERED=59000 CU. FT.                                                          | SCND_STAGE
     C|  PLANFORM AREA PROPORTIONAL TO TANK VOLUME                                               | SCND_STAGE
     C|  LO2 DENSITY=2.228                                                                       | SCND_STAGE
     C|  LH2 DENSITY=.1357                                                                       | SCND_STAGE

 412 M|       END;                                                                               | SCND_STAGE

 413 M|    ELSE                                                                                  | SCND_STAGE

 413 M|       DO;                                                                                | SCND_STAGE

 414 M| 1       IF ((ANGLE_OF_ATTACK = SS_ANGLE_OF_ATTACK_VAL                    ) OR (ANGLE_OF_ATTACK > | SCND_STAGE
     S|                                                 MAX_SS_ANGLE_INDEX                       |

 414 M| 1       SS_ANGLE_OF_ATTACK_VAL                    )) THEN                                    | SCND_STAGE
     S|                             MAX_SS_ANGLE_INDEX                                           |

 415 M| 1          HIGH_ANGLE = MAX_SS_ANGLE_INDEX;                                              | SCND_STAGE

 416 M| 1       ELSE                                                                             | SCND_STAGE

 416 M| 1          DO;                                                                           | SCND_STAGE

     E|                 .                                                                        |
 417 M| 2             ANG_FIND = ON;                                                             | SCND_STAGE
```

```
STMT                                    SOURCE                                                CURRENT SCOPE

     E|                                                                                       |
418 M| 2            DO FOR I_SEARCH = 2 TO MAX_SS_ANGLE_INDEX WHILE ANG_FIND = ON;             | SCND_STAGE

419 M| 3               IF SS_ANGLE_OF_ATTACK_VAL          > ANGLE_OF_ATTACK THEN              | SCND_STAGE
    S|                                         I_SEARCH                                       |

420 M| 3                  DO;                                                                  | SCND_STAGE

421 M| 4                     HIGH_ANGLE = I_SEARCH;                                            | SCND_STAGE

     E|                                                                                       |
422 M| 4                     ANG_FIND = OFF;                                                   | SCND_STAGE

423 M| 3                  END;                                                                 | SCND_STAGE

424 M| 2            END;                                                                       | SCND_STAGE

425 M| 1         END;                                                                          | SCND_STAGE

426 M| 1      LOW_ANGLE = HIGH_ANGLE - 1;                                                      | SCND_STAGE

427 M| 1      IF M0 < SS_M_VAL  THEN                                                           | SCND_STAGE
    S|                      1                                                                 |

428 M| 1         DO;                                                                           | SCND_STAGE

429 M| 2            LOW_CL = SS_CL_MAT            ;                                            | SCND_STAGE
    S|                               LOW_ANGLE,1                                              |

430 M| 2            LOW_CD = SS_CD_MAT            ;                                            | SCND_STAGE
    S|                               LOW_ANGLE,1                                              |

431 M| 2            HIGH_CL = SS_CL_MAT            ;                                           | SCND_STAGE
    S|                                HIGH_ANGLE,1                                            |

432 M| 2            HIGH_CD = SS_CD_MAT            ;                                           | SCND_STAGE
    S|                                HIGH_ANGLE,1                                            |

433 M| 1         END;                                                                          | SCND_STAGE

434 M| 1      IF ((M0 = SS_M_VAL              ) OR (M0 > SS_M_VAL              )) THEN         | SCND_STAGE
    S|                      MAX_SS_M_INDEX                    MAX_SS_M_INDEX                  |

435 M| 1         DO;                                                                           | SCND_STAGE

436 M| 2            LOW_CL = SS_CL_MAT                         ;                               | SCND_STAGE
    S|                               LOW_ANGLE,MAX_SS_M_INDEX                                 |

437 M| 2            LOW_CD = SS_CD_MAT                         ;                               | SCND_STAGE
    S|                               LOW_ANGLE,MAX_SS_M_INDEX                                 |

438 M| 2            HIGH_CL = SS_CL_MAT                         ;                              | SCND_STAGE
    S|                                HIGH_ANGLE,MAX_SS_M_INDEX                               |
```

```
STMT                                      SOURCE                                              CURRENT SCOPE

 439 M| 2           HIGH_CD = SS_CD_MAT                           ;                          | SCND_STAGE
     S|                                HIGH_ANGLE,MAX_SS_M_INDEX                              |

 440 M| 1         END;                                                                       | SCND_STAGE

     E|         .                                                                            |
 441 M| 1       M0_FLAG = OFF;                                                               | SCND_STAGE

 442 M| 1       IF (((M0 = SS_M_VAL ) OR (M0 > SS_M_VAL )) AND (M0 < SS_M_VAL              )) THEN | SCND_STAGE
     S|                           1                 1                  MAX_SS_M_INDEX         |

 443 M| 1         DO;                                                                        | SCND_STAGE

     E|              .                                                                       |
 444 M| 2           M_FIND = ON;                                                             | SCND_STAGE

     E|                                                       .                              |
 445 M| 2           DO FOR I_SEARCH = 2 TO MAX_SS_M_INDEX WHILE M_FIND = ON;                 | SCND_STAGE

 446 M| 3             IF SS_M_VAL         > M0 THEN                                          | SCND_STAGE
     S|                          I_SEARCH                                                    |

 447 M| 3               DO;                                                                  | SCND_STAGE

 448 M| 4                 HIGH_M = I_SEARCH;                                                 | SCND_STAGE

     E|                    .                                                                 |
 449 M| 4                 M_FIND = OFF;                                                      | SCND_STAGE

 450 M| 3               END;                                                                 | SCND_STAGE

 451 M| 2           END;                                                                     | SCND_STAGE

 452 M| 2           LOW_M = HIGH_M - 1;                                                      | SCND_STAGE

 453 M| 2           F1 = (M0 - SS_M_VAL     ) / (SS_M_VAL       - SS_M_VAL     );             | SCND_STAGE
     S|                             LOW_M              HIGH_M           LOW_M                |

 454 M| 2           LOW_CL = SS_CL_MAT                 + (F1 (SS_CL_MAT                - SS_CL_MAT | SCND_STAGE
     S|                            LOW_ANGLE,LOW_M                LOW_ANGLE,HIGH_M           |

 454 M| 2                                  ));                                               | SCND_STAGE
     S|             LOW_ANGLE,LOW_M                                                          |

 455 M| 2           LOW_CD = SS_CD_MAT                 + (F1 (SS_CD_MAT                - SS_CD_MAT | SCND_STAGE
     S|                            LOW_ANGLE,LOW_M                LOW_ANGLE,HIGH_M           |

 455 M| 2                                  ));                                               | SCND_STAGE
     S|             LOW_ANGLE,LOW_M                                                          |

 456 M| 2           HIGH_CL = SS_CL_MAT                 + (F1 (SS_CL_MAT                - SS_CL_MAT | SCND_STAGE
     S|                             HIGH_ANGLE,LOW_M                HIGH_ANGLE,HIGH_M        |

 456 M| 2                                   ));                                              | SCND_STAGE
     S|             HIGH_ANGLE,LOW_M                                                         |
```

```
STMT                                       SOURCE                                                    CURRENT SCOPE

457 M| 2         HIGH_CD = SS_CD_MAT               + (F1 (SS_CD_MAT                  - SS_CD_MAT    | SCND_STAGE
    S|                      HIGH_ANGLE,LOW_M                  HIGH_ANGLE,HIGH_M                     |

457 M| 2                             ));                                                            | SCND_STAGE
    S|           HIGH_ANGLE,LOW_M                                                                   |

    E|              .                                                                               |
458 M| 2         M0_FLAG = ON;                                                                      | SCND_STAGE

459 M| 1      END;                                                                                  | SCND_STAGE

460 M| 1   F1 = (ANGLE_OF_ATTACK - SS_ANGLE_OF_ATTACK_VAL          ) / (SS_ANGLE_OF_ATTACK_VAL     | SCND_STAGE
    S|                                                LOW_ANGLE                                     |

460 M| 1               - SS_ANGLE_OF_ATTACK_VAL           );                                        | SCND_STAGE
    S|     HIGH_ANGLE                          LOW_ANGLE                                            |

461 M| 1   SS_CL = LOW_CL + (F1 (HIGH_CL - LOW_CL));                                                | SCND_STAGE

462 M| 1   SS_CD = LOW_CD + (F1 (HIGH_CD - LOW_CD));                                                | SCND_STAGE

463 M| 1   DCL2_DUA1 = (HIGH_CL - LOW_CL) / (SS_ANGLE_OF_ATTACK_VAL           -                    | SCND_STAGE
    S|                                                          HIGH_ANGLE                          |

463 M| 1   SS_ANGLE_OF_ATTACK_VAL          );                                                       | SCND_STAGE
    S|                         LOW_ANGLE                                                            |

464 M| 1   DCD2_DUA1 = (HIGH_CD - LOW_CD) / (SS_ANGLE_OF_ATTACK_VAL           -                    | SCND_STAGE
    S|                                                          HIGH_ANGLE                          |

464 M| 1   SS_ANGLE_OF_ATTACK_VAL          );                                                       | SCND_STAGE
    S|                         LOW_ANGLE                                                            |

    E|        .                                                                                     |
465 M| 1   IF M0_FLAG = ON THEN                                                                     | SCND_STAGE

466 M| 1      DO;                                                                                   | SCND_STAGE

467 M| 2         LOW_CL = SS_CL_MAT               + (F1 (SS_CL_MAT                - SS_CL_MAT      | SCND_STAGE
    S|                             LOW_ANGLE,LOW_M                 HIGH_ANGLE,LOW_M                 |

467 M| 2                            ));                                                             | SCND_STAGE
    S|           LOW_ANGLE,LOW_M                                                                    |

468 M| 2         LOW_CD = SS_CD_MAT               + (F1 (SS_CD_MAT                - SS_CD_MAT      | SCND_STAGE
    S|                             LOW_ANGLE,LOW_M                 HIGH_ANGLE,LOW_M                 |

468 M| 2                            ));                                                             | SCND_STAGE
    S|           LOW_ANGLE,LOW_M                                                                    |

469 M| 2         HIGH_CL = SS_CL_MAT               + (F1 (SS_CL_MAT                - SS_CL_MAT     | SCND_STAGE
    S|                             LOW_ANGLE,HIGH_M                 HIGH_ANGLE,HIGH_M               |

469 M| 2                             ));                                                            | SCND_STAGE
    S|           LOW_ANGLE,HIGH_M                                                                   |
```

```
STMT                                   SOURCE                                                CURRENT SCOPE

470 M| 2          HIGH_CD = SS_CD_MAT                 + (F1 (SS_CD_MAT                 - SS_CD_MAT  | SCND_STAGE
    S|                               LOW_ANGLE,HIGH_M                  HIGH_ANGLE,HIGH_M             |

470 M| 2                            ));                                                            | SCND_STAGE
    S|            LOW_ANGLE,HIGH_M                                                                 |

471 M| 2          DCL2_DM0 = ((HIGH_CL - LOW_CL) / (SS_M_VAL       - SS_M_VAL     ));              | SCND_STAGE
    S|                                                  HIGH_M            LOW_M                    |

472 M| 2          DCD2_DM0 = ((HIGH_CD - LOW_CD) / (SS_M_VAL       - SS_M_VAL     ));              | SCND_STAGE
    S|                                                  HIGH_M            LOW_M                    |

473 M| 1        END;                                                                               | SCND_STAGE

474 M| 1      ELSE                                                                                 | SCND_STAGE

474 M| 1        DO;                                                                                | SCND_STAGE

475 M| 2          DCL2_DM0 = 0.;                                                                   | SCND_STAGE

476 M| 2          DCD2_DM0 = 0.;                                                                   | SCND_STAGE

477 M| 1        END;                                                                               | SCND_STAGE

478 M| 1      F1 = SQRT(L_D_SCALE_FACTOR);                                                         | SCND_STAGE

479 M| 1      SS_CL = SS_CL F1;                                                                    | SCND_STAGE

480 M| 1      SS_CD = SS_CD / F1;                                                                  | SCND_STAGE

481 M| 1      DCL2_DM0 = DCL2_DM0 F1;                                                              | SCND_STAGE

482 M| 1      DCD2_DM0 = DCD2_DM0 / F1;                                                            | SCND_STAGE

483 M| 1      DCL2_DUA1 = DCL2_DUA1 F1;                                                            | SCND_STAGE

484 M| 1      DCD2_DUA1 = DCD2_DUA1 / F1;                                                          | SCND_STAGE

    C|  THE SCALING FACTOR (SQRT(F1)) IS TO ALLOW SHUTTLE LIFT/DRAG VALUES                         | SCND_STAGE
    C|  TO BE USED WHILE ASSUMING IMPROVED AERODYNAMICS OF A VEHICLE IN THE                        | SCND_STAGE
    C|  FUTURE                                                                                     | SCND_STAGE

485 M| 1      SS_LIFT = SS_CL SS_PLANFORM_AREA DYNAMIC_P0;                                         | SCND_STAGE

486 M| 1      SS_DRAG = SS_CD SS_PLANFORM_AREA DYNAMIC_P0;                                         | SCND_STAGE

487 M| 1      IF SS_DRAG < 0. THEN                                                                 | SCND_STAGE

488 M| 1        DO;                                                                                | SCND_STAGE

489 M| 2          SS_DRAG = 0.;                                                                    | SCND_STAGE

490 M| 2          SS_CD = 0.;                                                                      | SCND_STAGE

491 M| 2          DCD2_DM0 = 0.;                                                                   | SCND_STAGE
```

```
STMT                                SOURCE                                          CURRENT SCOPE

 492 M| 2             DCD2_DUA1 = 0.;                                               | SCND_STAGE

 493 M| 1          END;                                                             | SCND_STAGE

 494 M|       END;                                                                  | SCND_STAGE

 495 M| CLOSE SCND_STAGE;                                                           | SCND_STAGE

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
    DELIVERED_MASS, DELIVERED_PLANFORM_AREA, MAX_SS_ANGLE_INDEX, SS_ANGLE_OF_ATTACK_VAL, SS_M_VAL, SS_CL_MAT, SS_CD_MAT
    MAX_SS_M_INDEX, L_D_SCALE_FACTOR

OUTER VARIABLES USED
    RESHAPE_FLAG, SS_DRY_MASS*, SS_MAX_FUEL_LOAD, MAX_ROCKET_THRUST, SS_PLANFORM_AREA*, ANGLE_OF_ATTACK, I_SEARCH*, I_SEARCH, MO
    LOW_CL*, LOW_CD*, HIGH_CL*, HIGH_CD*, M_FIND*, M_FIND, HIGH_M*, LOW_M*, HIGH_M, F1*, LOW_M, F1, SS_CL*, LOW_CL, HIGH_CL, SS_CD*
    LOW_CD, HIGH_CD, DCL2_DUA1*, DCD2_DUA1*, DCL2_DM0*, DCD2_DM0*, SS_CL, SS_CD, DCL2_DM0, DCD2_DM0, DCL2_DUA1, DCD2_DUA1, SS_LIFT*
    SS_PLANFORM_AREA, DYNAMIC_P0, SS_DRAG*, SS_DRAG
```

```
STMT                              SOURCE                                        CURRENT SCOPE

    E|             .                                                            |
496 M|   IF RESHAPE_FLAG = OFF THEN                                             | VEHICLE

497 M|      DO;                                                                 | VEHICLE

    C| THIS IS CALLED TO COMPUTE VEHICLE FORCES, MAXIMUM THRUST, AND            | VEHICLE
    C| MAXIMUM FUEL FLOW                                                        | VEHICLE

    E|                .                                                         |
498 M| 1       IF STAGE_SEP = OFF THEN                                          | VEHICLE

499 M| 1          CALL FIRST_STAGE;                                             | VEHICLE

500 M| 1       CALL SCND_STAGE;                                                 | VEHICLE

501 M|      END;                                                                | VEHICLE

502 M|   ELSE                                                                   | VEHICLE

502 M|      DO;                                                                 | VEHICLE

    C| THIS IS CALLED IF THE VEHICLE GEOMETRY IS TO BE COMPUTED ALONG           | VEHICLE
    C| WITH EMPTY FUEL TANK VEHICLE MASS PROPERTIES                             | VEHICLE

503 M| 1       CALL FIRST_STAGE;                                                | VEHICLE

504 M| 1       CALL SCND_STAGE;                                                 | VEHICLE

    E|                .                                                         |
505 M| 1       RESHAPE_FLAG = OFF;                                              | VEHICLE

506 M|      END;                                                                | VEHICLE

507 M| CLOSE VEHICLE;                                                           | VEHICLE

**** B L O C K   S U M M A R Y ****

OUTER VARIABLES USED
   RESHAPE_FLAG, STAGE_SEP, RESHAPE_FLAG*
```

```
STMT                                   SOURCE                                         CURRENT SCOPE

 508 M| ENVIRONMENT:                                                                   | ENVIRONMENT

 508 M| PROCEDURE;                                                                     | ENVIRONMENT

 509 M|    DECLARE LOW_ALT INTEGER AUTOMATIC;                                          | ENVIRONMENT

 510 M|    DECLARE HIGH_ALT INTEGER AUTOMATIC;                                         | ENVIRONMENT

 511 M|    DECLARE ALTITUDE SCALAR DOUBLE AUTOMATIC;                                   | ENVIRONMENT

 512 M|    DECLARE REL_RO SCALAR DOUBLE AUTOMATIC;                                     | ENVIRONMENT

 513 M|    ALTITUDE = X_STORE         / (FEET_PER_METER ALT_METER_INTERVAL);           | ENVIRONMENT
     S|                  I_TIME:1                                                      |

 514 M|    LOW_ALT = FLOOR(ALTITUDE) + 1;                                              | ENVIRONMENT

 515 M|    IF LOW_ALT > MAX_ALT THEN                                                   | ENVIRONMENT

 516 M|       DO;                                                                      | ENVIRONMENT

 517 M| 1        WRITE(6) SKIP(2), COLUMN(30), 'ALTITUDE IS TOO HIGH', SKIP(2);        | ENVIRONMENT

 518 M| 1        REL_RO = ATM_DENS         ;                                           | ENVIRONMENT
     S|                          MAX_ALT+1                                             |

 519 M| 1        TO = ATM_TEMP          1.8;                                          | ENVIRONMENT
     S|                      MAX_ALT+1                                                 |

     C|  THESE VALUES ARE USED IF THE ALTITUDE IS ABOVE THE DATA RANGE                 | ENVIRONMENT

 520 M|       END;                                                                     | ENVIRONMENT

 521 M|    ELSE                                                                        | ENVIRONMENT

 521 M|       DO;                                                                      | ENVIRONMENT

 522 M| 1        IF LOW_ALT < 1 THEN                                                   | ENVIRONMENT

 523 M| 1           DO;                                                                | ENVIRONMENT

 524 M| 2              TO = ATM_TEMP  1.8;                                             | ENVIRONMENT
     S|                             1                                                  |

 525 M| 2              REL_RO = ATM_DENS ;                                             | ENVIRONMENT
     S|                                 1                                              |

     C|  THESE VALUES ARE USED IF THE ALTITUDE IS BELOW THE DATA RANGE                 | ENVIRONMENT

 526 M| 1           END;                                                               | ENVIRONMENT

 527 M| 1        ELSE                                                                  | ENVIRONMENT

 527 M| 1           DO;                                                                | ENVIRONMENT
```

```
STMT                                   SOURCE                                                    CURRENT SCOPE

528 M| 2          HIGH_ALT = LOW_ALT + 1;                                                       | ENVIRONMENT

529 M| 2          T0 = (ATM_TEMP        + ((ALTITUDE - LOW_ALT + 1) (ATM_TEMP         - ATM_TEMP | ENVIRONMENT
    S|                     LOW_ALT                                        HIGH_ALT               |

529 M| 2                ))) 1.8;                                                                | ENVIRONMENT
    S|            LOW_ALT                                                                        |

530 M| 2          DTO_DR = ((ATM_TEMP          - ATM_TEMP        ) 1.8) / (FEET_PER_METER      | ENVIRONMENT
    S|                             HIGH_ALT            LOW_ALT                                   |

530 M| 2          ALT_METER_INTERVAL);                                                          | ENVIRONMENT

531 M| 2          F1 = LOG(ATM_DENS         / ATM_DENS         );                               | ENVIRONMENT
    S|                          LOW_ALT           HIGH_ALT                                       |

532 M| 2          F2 = ATM_DENS        EXP(F1 (LOW_ALT - 1));                                   | ENVIRONMENT
    S|                       LOW_ALT                                                             |

533 M| 2          REL_RO = F2 EXP(-F1 ALTITUDE);                                                | ENVIRONMENT

    C|  THE DENSITY IS COMPUTED FROM AN EXPONENTIAL INTERPOLATION SCHEME                        | ENVIRONMENT

534 M| 2          DRO_DR = -(F1 REL_RO GROUND_RO) / (FEET_PER_METER ALT_METER_INTERVAL);        | ENVIRONMENT

535 M| 1        END;                                                                            | ENVIRONMENT

536 M|       END;                                                                               | ENVIRONMENT

537 M|    RO_0 = REL_RO GROUND_RO;                                                              | ENVIRONMENT

538 M|    P0 = RO_0 R_0 T0;                                                                     | ENVIRONMENT

539 M|    M0 = U0 / SQRT(GAMMA0 R_0 T0);                                                        | ENVIRONMENT

540 M|    G = (UNIVERSAL_G_CONSTANT EARTH_MASS) / (R R);                                        | ENVIRONMENT

541 M| CLOSE ENVIRONMENT;                                                                       | ENVIRONMENT

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
    X_STORE, FEET_PER_METER, ALT_METER_INTERVAL, MAX_ALT, ATM_DENS, ATM_TEMP, GROUND_RO, R_0, GAMMA0, UNIVERSAL_G_CONSTANT
    EARTH_MASS

OUTER VARIABLES USED
    I_TIME, T0*, DTO_DR*, F1*, F2*, F1, F2, DRO_DR*, RO_0*, P0*, RO_0, T0, M0*, U0, G*, R
```

```
STMT                                   SOURCE                                   CURRENT SCOPE

 542 M|    NOSE_ANGLE = P ;                                                     | MODEL_DRIVER
     S|                  1                                                      |

 543 M|    W_SCRJ = P ;                                                         | MODEL_DRIVER
     S|              2                                                          |

 544 M|    H_C = P ;                                                            | MODEL_DRIVER
     S|           3                                                             |

 545 M|    H_C_TJ = P ;                                                         | MODEL_DRIVER
     S|              4                                                          |

 546 M|    WING_SPAN = P ;                                                      | MODEL_DRIVER
     S|                 5                                                       |

 547 M|    DELTA_ANGLE = P ;                                                    | MODEL_DRIVER
     S|                   6                                                     |

 548 M|    HC_TANK_VOL_FRACTION = P ;                                           | MODEL_DRIVER
     S|                            7                                            |

 549 M|    FIRST_STAGE_LENGTH = P ;                                             | MODEL_DRIVER
     S|                          8                                              |

 550 M|    SS_MAX_FUEL_LOAD = P ;                                               | MODEL_DRIVER
     S|                        9                                                |

 551 M|    MAX_ROCKET_THRUST = P  ;                                             | MODEL_DRIVER
     S|                         10                                              |

     E|               .                                                         |
 552 M|    IF RESHAPE_FLAG = ON THEN                                            | MODEL_DRIVER

 553 M|       CALL VEHICLE;                                                     | MODEL_DRIVER

 554 M|    ELSE                                                                 | MODEL_DRIVER

 554 M|       DO;                                                               | MODEL_DRIVER

     E|                  .                                                      |
 555 M| 1          OBLIQUE_SHOCK_FLAG = OFF;                                    | MODEL_DRIVER

     E|                  .                                                      |
 556 M| 1          NORMAL_SHOCK_FLAG = OFF;                                     | MODEL_DRIVER

     E|                .                                                        |
 557 M| 1          EXPANSION_FLAG = OFF;                                        | MODEL_DRIVER

     E|                .                                                        |
 558 M| 1          SUBSONIC_FLAG = OFF;                                         | MODEL_DRIVER

 559 M| 1          DCL_DALPHA = 0.;                                             | MODEL_DRIVER

 560 M| 1          DCD_DCL2 = 0.;                                               | MODEL_DRIVER
```

```
STMT                               SOURCE                                              CURRENT SCOPE
 561 M| 1      CL = 0.;                                                              | MODEL_DRIVER
 562 M| 1      CD = 0.;                                                              | MODEL_DRIVER
 563 M| 1      DCL1_DM0 = 0.;                                                        | MODEL_DRIVER
 564 M| 1      DCL2_DM0 = 0.;                                                        | MODEL_DRIVER
 565 M| 1      DCD1_DM0 = 0.;                                                        | MODEL_DRIVER
 566 M| 1      DCD2_DM0 = 0.;                                                        | MODEL_DRIVER
 567 M| 1      DCL1_DAR = 0.;                                                        | MODEL_DRIVER
 568 M| 1      DCD1_DAR = 0.;                                                        | MODEL_DRIVER
 569 M| 1      DMCR_DM2 = 0.;                                                        | MODEL_DRIVER
 570 M| 1      DT0_DR = 0.;                                                          | MODEL_DRIVER
 571 M| 1      DR0_DR = 0.;                                                          | MODEL_DRIVER
 572 M| 1      R = X_STORE        + EARTH_RADIUS;                                    | MODEL_DRIVER
     S|                  I_TIME:1                                                    |
 573 M| 1      U_R = X_STORE        ;                                                | MODEL_DRIVER
     S|                    I_TIME:2                                                  |
 574 M| 1      U_THETA = R X_STORE        ;                                          | MODEL_DRIVER
     S|                          I_TIME:4                                            |
     E|                                                                              |
 575 M| 1      IF STATE_INTEGRATION_FLAG = ON THEN                                   | MODEL_DRIVER
 576 M| 1         DO;                                                                | MODEL_DRIVER
 577 M| 2            ANGLE_OF_ATTACK = U_ACTIVE        ;                             | MODEL_DRIVER
     S|                                      I_TIME:1                                |
 578 M| 2            CAP_PHI = 1. / (1. + (U_ACTIVE         U_ACTIVE        ));       | MODEL_DRIVER
     S|                                          I_TIME:2         I_TIME:2           |
 579 M| 1         END;                                                               | MODEL_DRIVER
 580 M| 1      ELSE                                                                  | MODEL_DRIVER
 580 M| 1         DO;                                                                | MODEL_DRIVER
 581 M| 2            ANGLE_OF_ATTACK = U_ACTIVE        ;                             | MODEL_DRIVER
     S|                                      L_TIME:1                                |
 582 M| 2            CAP_PHI = 1. / (1. + (U_ACTIVE         U_ACTIVE        ));       | MODEL_DRIVER
     S|                                          L_TIME:2         L_TIME:2           |
 583 M| 1         END;                                                               | MODEL_DRIVER
```

```
STMT                                SOURCE                                      CURRENT SCOPE

 584 M| 1        M2 = 0.;                                                        | MODEL_DRIVER

 585 M| 1        RO_2 = 0.;                                                      | MODEL_DRIVER

 586 M| 1        U2 = 0.;                                                        | MODEL_DRIVER

 587 M| 1        T2 = 0.;                                                        | MODEL_DRIVER

 588 M| 1        U_T_AIR = U_THETA - (R EARTH_OMEGA);                            | MODEL_DRIVER

     C| IT IS ASSUMED THAT THE ATMOSPHERE MOVES WITH THE EARTH'S SURFACE          | MODEL_DRIVER

 589 M| 1        VEHICLE_ANGLE = ARCTAN(U_R / U_T_AIR) + ANGLE_OF_ATTACK;        | MODEL_DRIVER

 590 M| 1        U0 = SQRT((U_R U_R) + (U_T_AIR U_T_AIR));                       | MODEL_DRIVER

 591 M| 1        CALL ENVIRONMENT;                                               | MODEL_DRIVER

 592 M| 1        DYNAMIC_P0 = (RO_0 U0 U0) / 2.;                                 | MODEL_DRIVER

     E|                 .                                                        |
 593 M| 1        IF STAGE_SEP = OFF THEN                                         | MODEL_DRIVER

 594 M| 1           DO;                                                          | MODEL_DRIVER

 595 M| 2              IF M0 < 1. THEN                                           | MODEL_DRIVER

 596 M| 2                 DO;                                                    | MODEL_DRIVER

     C| THE CHANGE IN FLOW PROPERTIES FOR SUBSONIC FLOW IS IGNORED DUE TO        | MODEL_DRIVER
     C| THE SMALL EXPECTED FLOW DIRECTION CHANGES                                | MODEL_DRIVER

     E|                          .                                               |
 597 M| 3                    SUBSONIC_FLAG = ON;                                 | MODEL_DRIVER

 598 M| 3                    P1 = P0;                                            | MODEL_DRIVER

 599 M| 3                    RO_1 = RO_0;                                        | MODEL_DRIVER

 600 M| 3                    T1 = T0;                                            | MODEL_DRIVER

 601 M| 3                    M1 = M0;                                            | MODEL_DRIVER

 602 M| 3                    U1 = U0;                                            | MODEL_DRIVER

 603 M| 2                 END;                                                   | MODEL_DRIVER

 604 M| 2              ELSE                                                      | MODEL_DRIVER

 604 M| 2                 DO;                                                    | MODEL_DRIVER

 605 M| 3                    M0_2 = M0 M0;                                       | MODEL_DRIVER

     C| THETA_NOSE, THE NOSE TURNING ANGLE IS SET EQUAL TO THE SUM OF THE        | MODEL_DRIVER
     C| ANGLE OF ATTACK AND VEHICLE CENTER LINE NOSE ANGLE.  IF THETA_NOSE       | MODEL_DRIVER
```

```
STMT                                        SOURCE                                                  CURRENT SCOPE

    C| IS GREATER THAN ZERO, AND THE FREE STREAM MACH NUMBER IS GREATER                           | MODEL_DRIVER
    C| THAN ONE, THEN A SEARCH IS MADE FOR A WEAK SHOCK SOLUTION.  IF A                           | MODEL_DRIVER
    C| SOLUTION EXISTS, THEN IT IS FOUND. OTHERWISE NORMAL SHOCK                                  | MODEL_DRIVER
    C| THEORY AND/OR SUBSONIC FLOW THEORY IS USED.                                                | MODEL_DRIVER

606 M| 3                 THETA_NOSE = ANGLE_OF_ATTACK + NOSE_ANGLE;                                   | MODEL_DRIVER

607 M| 3                 IF THETA_NOSE > 0. THEN                                                      | MODEL_DRIVER

608 M| 3                    DO;                                                                       | MODEL_DRIVER

    C| THE FOLLOWING IS FOR SHOCKED FLOW                                                          | MODEL_DRIVER

609 M| 4                       SIN_BETA_THETA_MAX = SQRT(((GAM2 M0_2) - 4. + SQRT(GAM2 ((GAM2 M0_2 M0_2 | MODEL_DRIVER

609 M| 4                       ) + (8. GAM3 M0_2) + 16.))) / (4. GAMMA0 M0_2));                       | MODEL_DRIVER

610 M| 4                       BETA_THETA_MAX = ARCSIN(SIN_BETA_THETA_MAX);                           | MODEL_DRIVER

611 M| 4                       TAN_THETA_MAX = (2. ((M0_2 SIN_BETA_THETA_MAX SIN_BETA_THETA_MAX) - 1.)) | MODEL_DRIVER

611 M| 4                       / (TAN(BETA_THETA_MAX) ((M0_2 (GAMMA0 + COS(2. BETA_THETA_MAX))) + 2.)); | MODEL_DRIVER

612 M| 4                       THETA_MAX = ARCTAN(TAN_THETA_MAX);                                     | MODEL_DRIVER

613 M| 4                       IF ((THETA_NOSE < THETA_MAX) OR (THETA_NOSE = THETA_MAX)) THEN         | MODEL_DRIVER

614 M| 4                          DO;                                                                 | MODEL_DRIVER

    C| THE FOLLOWING IS FOR OBLIQUE SHOCKS                                                        | MODEL_DRIVER

    E|                                .                                                           |
615 M| 5                             OBLIQUE_SHOCK_FLAG = ON;                                         | MODEL_DRIVER

616 M| 5                             EXTREMAL_ARRAY  = THETA_NOSE;                                    | MODEL_DRIVER
    S|                                             1                                              |

61. M| 5                             EXTREMAL_ARRAY  = ARCSIN(1. / M0);                               | MODEL_DRIVER
    S|                                             2                                              |

618 M| 5                             BETA_LOWER_BOUND = MAX([EXTREMAL_ARRAY]);                        | MODEL_DRIVER

619 M| 5                             BETA_UPPER_BOUND = BETA_THETA_MAX;                               | MODEL_DRIVER

620 M| 5                             L_BETA = TAN(THETA_NOSE);                                        | MODEL_DRIVER

621 M| 5                             DO FOR I_BETA = 1 TO MAX_BETA_CYCLES;                            | MODEL_DRIVER

622 M| 6                                BETA_NEW_BOUND = (BETA_UPPER_BOUND + BETA_LOWER_BOUND) / 2.;  | MODEL_DRIVER

    E|                                                                          2                  |
623 M| 6                                R_NEW_BOUND = (2. ((M0_2 (SIN(BETA_NEW_BOUND) )) - 1.)) / (TAN( | MODEL_DRIVER

623 M| 6                                BETA_NEW_BOUND) ((M0_2 (GAMMA0 + COS(2. BETA_NEW_BOUND))) + 2.) | MODEL_DRIVER
```

```
STMT                                  SOURCE                                              CURRENT SCOPE
 623 M| 6                       );                                                       | MODEL_DRIVER
 624 M| 6                       IF R_NEW_BOUND > L_BETA THEN                             | MODEL_DRIVER
 625 M| 6                          BETA_UPPER_BOUND = BETA_NEW_BOUND;                    | MODEL_DRIVER
 626 M| 6                       ELSE                                                     | MODEL_DRIVER
 626 M| 6                          BETA_LOWER_BOUND = BETA_NEW_BOUND;                    | MODEL_DRIVER
 627 M| 5                    END;                                                        | MODEL_DRIVER
 628 M| 5                    BETA_NOSE_SHOCK = (BETA_LOWER_BOUND + BETA_UPPER_BOUND) / 2.; | MODEL_DRIVER
     E|                                               2                                  |
 629 M| 5                    F4 = SIN(BETA_NOSE_SHOCK) ;                                 | MODEL_DRIVER
 630 M| 5                    M1_2 = (1. + (GAM1 M0_2 F4)) / (((GAMMA0 M0_2 F4) - GAM1) (SIN( | MCDEL_DRIVER
     E|                                                   2                              |
 630 M| 5                    BETA_NOSE_SHOCK - THETA_NOSE) ));                           | MODEL_DRIVER
 631 M| 5                    P1 = (1. + ((2. GAMMA0 ((M0_2 F4) - 1.)) / GAM2)) P0;       | MODEL_DRIVER
 632 M| 5                    T1 = (1. + ((2. GAM3 ((M0_2 F4) - 1.) ((GAMMA0 M0_2 F4) + 1.)) / ( | MODEL_DRIVER
 632 M| 5                    GAM2 GAM2 M0_2 F4))) T0;                                    | MODEL_DRIVER
 633 M| 4                 END;                                                           | MODEL_DRIVER
 634 M| 4              ELSE                                                              | MODEL_DRIVER
 634 M| 4                 DO;                                                            | MODEL_DRIVER
     C|  THE FOLLOWING IS FOR NORMAL SHOCKS                                              | MODEL_DRIVER
     E|                                                                                  |
 635 M| 5                    NORMAL_SHOCK_FLAG = ON;                                     | MODEL_DRIVER
 636 M| 5                    M1_2 = (1. + (GAM1 M0_2)) / ((GAMMA0 M0_2) - GAM1);         | MODEL_DRIVER
 637 M| 5                    P1 = (1. + (((2. GAMMA0) (M0_2 - 1.)) / GAM2)) P0;          | MODEL_DRIVER
 638 M| 5                    T1 = (1. + ((2. GAM3 ((GAMMA0 M0_2) + 1.) (M0_2 - 1.)) / (GAM2 | MODEL_DRIVER
 638 M| 5                    GAM2 M0_2))) T0;                                            | MODEL_DRIVER
 639 M| 4                 END;                                                           | MODEL_DRIVER
 640 M| 3           END;                                                                  | MODEL_DRIVER
 641 M| 3        ELSE                                                                     | MODEL_DRIVER
 641 M| 3           DO;                                                                   | MODEL_DRIVER
```

```
STMT                              SOURCE                                        CURRENT SCOPE

    C| THE FOLLOWING IS FOR PRANDTL-MEYER EXPANDED FLOW                        | MODEL_DRIVER
     E|                     .                                                  |
642 M| 4                EXPANSION_FLAG = ON;                                   | MODEL_DRIVER
643 M| 4                F4 = SQRT(GAM2 / GAM3);                                | MODEL_DRIVER
644 M| 4                F5 = SQRT(M0_2 - 1.);                                  | MODEL_DRIVER
645 M| 4                NU0 = (F4 ARCTAN(F5 / F4)) - ARCTAN(F5);               | MODEL_DRIVER
646 M| 4                NU1 = NU0 - THETA_NOSE;                                | MODEL_DRIVER
647 M| 4                LOW_M_2 = M0_2;                                        | MODEL_DRIVER
648 M| 4                HIGH_M_2 = 2. LOW_M_2;                                 | MODEL_DRIVER
     E|                  .                                                     |
649 M| 4                M_2_FLAG = ON;                                         | MODEL_DRIVER
     E|                           .                                            |
650 M| 4                DO WHILE M_2_FLAG = ON;                                | MODEL_DRIVER
651 M| 5                   F3 = SQRT(HIGH_M_2 - 1.);                           | MODEL_DRIVER
652 M| 5                   HIGH_NU = (F4 ARCTAN(F3 / F4)) - ARCTAN(F3);        | MODEL_DRIVER
653 M| 5                   IF HIGH_NU > NU1 THEN                               | MODEL_DRIVER
     E|                         .                                              |
654 M| 5                      M_2_FLAG = OFF;                                  | MODEL_DRIVER
655 M| 5                   ELSE                                                | MODEL_DRIVER
655 M| 5                      HIGH_M_2 = 2. HIGH_M_2;                          | MODEL_DRIVER
656 M| 4                END;                                                   | MODEL_DRIVER
657 M| 4                DO FOR I_BETA = 1 TO MAX_BETA_CYCLES;                  | MODEL_DRIVER
658 M| 5                   MID_M_2 = (HIGH_M_2 + LOW_M_2) / 2.;                | MODEL_DRIVER
659 M| 5                   F3 = SQRT(MID_M_2 - 1.);                            | MODEL_DRIVER
660 M| 5                   MID_NU = (F4 ARCTAN(F3 / F4)) - ARCTAN(F3);         | MODEL_DRIVER
661 M| 5                   IF MID_NU > NU1 THEN                                | MODEL_DRIVER
662 M| 5                      HIGH_M_2 = MID_M_2;                              | MODEL_DRIVER
663 M| 5                   ELSE                                                | MODEL_DRIVER
663 M| 5                      LOW_M_2 = MID_M_2;                               | MODEL_DRIVER
664 M| 4                END;                                                   | MODEL_DRIVER
```

```
STMT                                  SOURCE                                         CURRENT SCOPE

 665 M| 4                   M1_2 = MID_M_2;                                          | MODEL_DRIVER

 666 M| 4                   T1 = (1. + (GAM1 M0_2)) / (1. + (GAM1 M1_2));            | MODEL_DRIVER

     E|                             GAMMA0/GAM3                                      |
 667 M| 4                   P1 = (T1           ) P0;                                 | MODEL_DRIVER

 668 M| 4                   T1 = T1 T0;                                              | MODEL_DRIVER

 669 M| 3                END;                                                        | MODEL_DRIVER

 670 M| 3             M1 = SQRT(M1_2);                                               | MODEL_DRIVER

 671 M| 3             RO_1 = P1 / (R_0 T1);                                          | MODEL_DRIVER

 672 M| 3             U1 = M1 SQRT(GAMMA0 R_0 T1);                                   | MODEL_DRIVER

 673 M| 2          END;                                                              | MODEL_DRIVER

     C|  THE CHANGES IN FLOW PROPERTIES IN THE SUBSONIC FLOW PAST THE                | MODEL_DRIVER
     C|  VEHICLE NOSE ARE IGNORED                                                    | MODEL_DRIVER

 674 M| 2          M2 = M1;                                                          | MODEL_DRIVER

 675 M| 2          U2 = U1;                                                          | MODEL_DRIVER

 676 M| 2          RO_2 = RO_1;                                                      | MODEL_DRIVER

 677 M| 2          T2 = T1;                                                          | MODEL_DRIVER

     C|  IT IS ASSUMED NO FLOW FIELD INTERACTIONS AFFECT FLUID PROPERTIES            | MODEL_DRIVER
     C|  BETWEEN THE VEHICLE NOSE AND PROPULSION INLETS                              | MODEL_DRIVER

 678 M| 1       END;                                                                 | MODEL_DRIVER

 679 M| 1    SIN_VEHICLE_ANGLE = SIN(VEHICLE_ANGLE);                                 | MODEL_DRIVER

 680 M| 1    COS_VEHICLE_ANGLE = COS(VEHICLE_ANGLE);                                 | MODEL_DRIVER

 681 M| 1    CALL VEHICLE;                                                           | MODEL_DRIVER

     E|           .                                                                  |
 682 M| 1    IF STAGE_SEP = OFF THEN                                                 | MODEL_DRIVER

 683 M| 1       ROCKET_MAX_T = 0.;                                                   | MODEL_DRIVER

 684 M| 1    ELSE                                                                    | MODEL_DRIVER

 684 M| 1       DO;                                                                  | MODEL_DRIVER

 685 M| 2          ROCKET_MAX_T = MAX_ROCKET_THRUST;                                 | MODEL_DRIVER

 686 M| 2          SCRAMJET_THRUST = 0.;                                             | MODEL_DRIVER

 687 M| 2          TURBOJET_THRUST = 0.;                                             | MODEL_DRIVER
```

```
STMT                                  SOURCE                                              CURRENT SCOPE
 688 M| 2            FS_LIFT = 0.;                                                       | MODEL_DRIVER
 689 M| 2            FS_DRAG = 0.;                                                       | MODEL_DRIVER
 690 M| 2            SCRJ_MASS_CAPTURE_RATIO = 0.;                                       | MODEL_DRIVER
 691 M| 1         END;                                                                   | MODEL_DRIVER
 692 M| 1      LIFT = FS_LIFT + SS_LIFT;                                                 | MODEL_DRIVER
 693 M| 1      DRAG = FS_DRAG + SS_DRAG;                                                 | MODEL_DRIVER
 694 M| 1      F1 = SIN(ANGLE_OF_ATTACK);                                                | MODEL_DRIVER
 695 M| 1      F2 = COS(ANGLE_OF_ATTACK);                                                | MODEL_DRIVER
 696 M| 1      P_AERO = (LIFT F1) - (DRAG F2);                                           | MODEL_DRIVER
 697 M| 1      N_AERO = (LIFT F2) + (DRAG F1);                                           | MODEL_DRIVER
     C|  P_AERO AND N_AERO CONSTITUTE THE AERODYNAMIC FORCES IN BODY X AND Y              | MODEL_DRIVER
     C|  AXES RESPECTIVELY                                                               | MODEL_DRIVER
 698 M| 1      MD_MASS = SS_DRY_MASS + X_STORE         + X_STORE         + X_STORE        ; | MODEL_DRIVER
     S|                                     I_TIME:5          I_TIME:6          I_TIME:7  |
     E|              .                                                                   |
 699 M| 1      IF STAGE_SEP = OFF THEN                                                   | MODEL_DRIVER
 700 M| 1         MD_MASS = MD_MASS + FIRST_STAGE_DRY_MASS;                              | MODEL_DRIVER
 701 M| 1      ROCKET_THRUST = CAP_PHI ROCKET_MAX_T;                                     | MODEL_DRIVER
     E|              .                                                                   |
 702 M| 1      IF STAGE_SEP = OFF THEN                                                   | MODEL_DRIVER
 703 M| 1         DO;                                                                    | MODEL_DRIVER
 704 M| 2            TJ_FUEL_FLOW = TURBOJET_THRUST / (TURBOJET_ISP G0);                 | MODEL_DRIVER
 705 M| 2            SCRJ_FUEL_FLOW = SCRAMJET_THRUST / (SCRAMJET_ISP G0);               | MODEL_DRIVER
 706 M| 1         END;                                                                   | MODEL_DRIVER
 707 M| 1      ELSE                                                                      | MODEL_DRIVER
 707 M| 1         DO;                                                                    | MODEL_DRIVER
 708 M| 2            TJ_FUEL_FLOW = 0.;                                                  | MODEL_DRIVER
 709 M| 2            SCRJ_FUEL_FLOW = 0.;                                                | MODEL_DRIVER
 710 M| 1         END;                                                                   | MODEL_DRIVER
 711 M| 1      SS_FUEL_FLOW = ROCKET THRUST / (ROCKET ISP G0);                           | MODEL_DRIVER
```

```
STMT                                       SOURCE                                                   CURRENT SCOPE

 712 M| 1       NET_X_FORCE = TURBOJET_THRUST + SCRAMJET_THRUST + ROCKET_THRUST + P_AERO;            | MODEL_DRIVER
 713 M| 1       NET_R_FORCE = (NET_X_FORCE SIN_VEHICLE_ANGLE) + (N_AERO COS_VEHICLE_ANGLE) - (MD_MASS G);  | MODEL_DRIVER
 714 M| 1       NET_THETA_FORCE = (NET_X_FORCE COS_VEHICLE_ANGLE) - (N_AERO SIN_VEHICLE_ANGLE);      | MODEL_DRIVER
 715 M| 1       G_LOAD = SQRT((NET_X_FORCE NET_X_FORCE) + (N_AERO N_AERO)) / (MD_MASS G0);           | MODEL_DRIVER
 716 M|      END;                                                                                     | MODEL_DRIVER
 717 M| CLOSE MODEL_DRIVER;                                                                         | MODEL_DRIVER

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
   P, X_STORE, EARTH_RADIUS, U_ACTIVE, EARTH_OMEGA, GAM2, GAM3, GAMMA0, MAX_BETA_CYCLES, GAM1, R_0, G0, ROCKET_ISP

OUTER VARIABLES USED
   RESHAPE_FLAG, OBLIQUE_SHOCK_FLAG*, NORMAL_SHOCK_FLAG*, EXPANSION_FLAG*, SUBSONIC_FLAG*, DCL_DALPHA*, DCD_DCL2*, CL*, CD*
   DCL1_DM0*, DCL2_DM0*, DCD1_DM0*, DCD2_DM0*, DCL1_DAR*, DCD1_DAR*, DMCR_DM2*, DT0_DR*, DRO_DR*, I_TIME, STATE_INTEGRATION_FLAG
   CAP_PHI*, L_TIME, M2*, RO_2*, U2*, T2*, VEHICLE_ANGLE*, DYNAMIC_P0*, RO_0, STAGE_SEP, M0, T0, M0_2*, THETA_NOSE*, THETA_NOSE
   M0_2, BETA_NOSE_SHOCK*, BETA_NOSE_SHOCK, SIN_VEHICLE_ANGLE*, VEHICLE_ANGLE, COS_VEHICLE_ANGLE*, SCRAMJET_THRUST*
   TURBOJET_THRUST*, SCRJ_MASS_CAPTURE_RATIO*, LIFT*, DRAG*, P_AERO*, LIFT, DRAG, N_AERO*, MD_MASS*, SS_DRY_MASS, MD_MASS
   FIRST_STAGE_DRY_MASS, ROCKET_THRUST*, CAP_PHI, TJ_FUEL_FLOW*, TURBOJET_THRUST, TURBOJET_ISP, SCRJ_FUEL_FLOW*, SCRAMJET_THRUST
   SCRAMJET_ISP, SS_FUEL_FLOW*, ROCKET_THRUST, NET_X_FORCE*, P_AERO, NET_R_FORCE*, NET_X_FORCE, SIN_VEHICLE_ANGLE, N_AERO
   COS_VEHICLE_ANGLE, G, NET_THETA_FORCE*, G_LOAD*
```

```
 STMT                                        SOURCE                                                         CURRENT SCOPE
 718 M| THESIS_ALGORITHM:                                                                                   | THESIS_ALGORITHM
 718 M| PROCEDURE;                                                                                          | THESIS_ALGORITHM
 719 M|    DECLARE I_FD INTEGER STATIC;                                                                     | THESIS_ALGORITHM
 720 M|    DECLARE FIRST_DERIV_FLAG BIT(1) STATIC;                                                          | THESIS_ALGORITHM
 721 M|    DECLARE PARTIAL_DERIV_FLAG BIT(1) STATIC;                                                        | THESIS_ALGORITHM
 722 M|    DECLARE DP_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                                               | THESIS_ALGORITHM
 723 M|    DECLARE DN_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                                               | THESIS_ALGORITHM
 724 M|    DECLARE RK_VAL_N MATRIX(10, NUM_CONSTRAINTS) DOUBLE STATIC;                                      | THESIS_ALGORITHM
 725 M|    DECLARE RK_VAL_N_PLUS_1 MATRIX(10, NUM_CONSTRAINTS) DOUBLE STATIC;                               | THESIS_ALGORITHM
     C| THE ABOVE TWO VARIABLES HAVE THE NUMBER OF ROWS EQUAL MAX(                                          | THESIS_ALGORITHM
     C| NUM_STATES+1,NUM_CONSTANT_PARAMETERS,NUM_CONSTRAINTS);                                             | THESIS_ALGORITHM
     C| THE ABOVE VARIABLES REQUIRE AN INITIALIZATION/DIMENSION MATCH                                       | THESIS_ALGORITHM
 726 M|    DECLARE NET_MASS ARRAY(STEP_DIM + 1) SCALAR DOUBLE STATIC;                                       | THESIS_ALGORITHM
 727 M|    DECLARE L_X_STORE ARRAY(STEP_DIM + 1) VECTOR(NUM_STATES) DOUBLE STATIC;                          | THESIS_ALGORITHM
 728 M|    DECLARE DM_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;                               | THESIS_ALGORITHM
 729 M|    DECLARE H_SUB_U ARRAY((STEP_DIM + 2) / 2) VECTOR(NUM_CONTROLS) DOUBLE STATIC;                    | THESIS_ALGORITHM
 730 M|    DECLARE DELTA_U ARRAY((STEP_DIM + 2) / 2) VECTOR(NUM_CONTROLS) DOUBLE STATIC;                    | THESIS_ALGORITHM
 731 M|    DECLARE DELTA_V VECTOR(NUM_CONSTANT_PARAMETERS + NUM_TRANS_PTS) DOUBLE STATIC;                   | THESIS_ALGORITHM
 732 M|    DECLARE G_MAT ARRAY((STEP_DIM + 2) / 2) MATRIX(NUM_STATES, NUM_CONTROLS) DOUBLE STATIC;          | THESIS_ALGORITHM
 733 M|    DECLARE CAP_LAMBDA_1 ARRAY((STEP_DIM + 2) / 2) MATRIX(NUM_STATES, NUM_CONSTRAINTS) DOUBLE STATIC | THESIS_ALGORITHM
 733 M|    ;                                                                                                | THESIS_ALGORITHM
 734 M|    DECLARE CAP_LAMBDA_2 MATRIX(NUM_CONSTANT_PARAMETERS, NUM_CONSTRAINTS) DOUBLE STATIC;             | THESIS_ALGORITHM
 735 M|    DECLARE PSI VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                                               | THESIS_ALGORITHM
 736 M|    DECLARE I_PSI_PSI MATRIX(NUM_CONSTRAINTS, NUM_CONSTRAINTS) DOUBLE STATIC;                        | THESIS_ALGORITHM
 737 M|    DECLARE DELTA_OMEGA_I_TIME ARRAY(NUM_TRANS_PTS) INTEGER STATIC;                                  | THESIS_ALGORITHM
 738 M|    DECLARE I_PSI_J VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                                           | THESIS_ALGORITHM
 739 M|    DECLARE X_DOT VECTOR(NUM_STATES) DOUBLE STATIC;                                                  | THESIS_ALGORITHM
 740 M|    DECLARE F MATRIX(NUM_STATES, NUM_STATES) DOUBLE STATIC;                                          | THESIS_ALGORITHM
 741 M|    DECLARE K MATRIX(NUM_STATES, NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                             | THESIS_ALGORITHM
```

```
STMT                                   SOURCE                                                                    CURRENT SCOPE
 742 M| DECLARE LAMBDA_DOT VECTOR(NUM_STATES) DOUBLE STATIC;                                               | THESIS_ALGORITHM
 743 M| DECLARE CAP_LAMBDA_1_DOT MATRIX(NUM_STATES, NUM_CONSTRAINTS) DOUBLE STATIC;                         | THESIS_ALGORITHM
 744 M| DECLARE CAP_LAMBDA_2_DOT MATRIX(NUM_CONSTANT_PARAMETERS, NUM_CONSTRAINTS) DOUBLE STATIC;            | THESIS_ALGORITHM
 745 M| DECLARE I_PSI_J_DOT VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                                          | THESIS_ALGORITHM
 746 M| DECLARE I_PSI_PSI_DOT MATRIX(NUM_CONSTRAINTS, NUM_CONSTRAINTS) DOUBLE STATIC;                       | THESIS_ALGORITHM
 747 M| DECLARE SGV_DOT VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                                      | THESIS_ALGORITHM
 748 M| DECLARE LAMBDA ARRAY(1001) VECTOR(NUM_STATES) DOUBLE STATIC;                                        | THESIS_ALGORITHM
 749 M| DECLARE M MATRIX(NUM_CONSTRAINTS, NUM_CONSTANT_PARAMETERS + NUM_TRANS_PTS) DOUBLE STATIC;           | THESIS_ALGORITHM
 750 M| DECLARE SMALL_G_VEC VECTOR(NUM_CONSTANT_PARAMETERS + NUM_TRANS_PTS) DOUBLE STATIC;                  | THESIS_ALGORITHM
 751 M| DECLARE U_J_OLD_TIME ARRAY((STEP_DIM + 2) / 2) SCALAR DOUBLE STATIC;                                | THESIS_ALGORITHM
 752 M| DECLARE U_NEW_TIME ARRAY(STEP_DIM + 1) SCALAR DOUBLE STATIC;                                        | THESIS_ALGORITHM
 753 M| DECLARE TIME_STEP SCALAR DOUBLE STATIC;                                                             | THESIS_ALGORITHM
 754 M| DECLARE FINAL_TIME_STEP SCALAR DOUBLE STATIC;                                                       | THESIS_ALGORITHM
 755 M| DECLARE TIME_INTERVAL SCALAR DOUBLE STATIC;                                                         | THESIS_ALGORITHM
 756 M| DECLARE C_SUB_PSI SCALAR DOUBLE STATIC;                                                             | THESIS_ALGORITHM
 757 M| DECLARE DELTA_COST SCALAR DOUBLE STATIC;                                                            | THESIS_ALGORITHM
 758 M| DECLARE DJS SCALAR DOUBLE STATIC;                                                                   | THESIS_ALGORITHM
 759 M| DECLARE I_J_J SCALAR DOUBLE STATIC;                                                                 | THESIS_ALGORITHM
 760 M| DECLARE I_J_J_DOT SCALAR DOUBLE STATIC;                                                             | THESIS_ALGORITHM
 761 M| DECLARE LAMBDA_FLAG BIT(1) STATIC;                                                                  | THESIS_ALGORITHM
 762 M| DECLARE CAP_LAMBDA_1_FLAG BIT(1) STATIC;                                                            | THESIS_ALGORITHM
 763 M| DECLARE CAP_LAMBDA_2_FLAG BIT(1) STATIC;                                                            | THESIS_ALGORITHM
 764 M| DECLARE SGV_FLAG BIT(1) STATIC;                                                                     | THESIS_ALGORITHM
 765 M| DECLARE I_J_J_FLAG BIT(1) STATIC;                                                                   | THESIS_ALGORITHM
 766 M| DECLARE I_PSI_J_FLAG BIT(1) STATIC;                                                                 | THESIS_ALGORITHM
 767 M| DECLARE I_PSI_PSI_FLAG BIT(1) STATIC;                                                               | THESIS_ALGORITHM
 768 M| DECLARE RK_ROWS INTEGER STATIC;                                                                     | THESIS_ALGORITHM
 769 M| DECLARE RK_COLUMNS INTEGER STATIC;                                                                  | THESIS_ALGORITHM
```

```
STMT                                SOURCE                                                CURRENT SCOPE
770 M|  DECLARE CONFIG_INDEX INTEGER STATIC;                                             | THESIS_ALGORITHM
771 M|  DECLARE MASS_FLOW_FINAL_STEP SCALAR DOUBLE STATIC;                               | THESIS_ALGORITHM
772 M|  DECLARE M1_FLOW_FINAL_STEP SCALAR DOUBLE STATIC;                                 | THESIS_ALGORITHM
773 M|  DECLARE M2_FLOW_FINAL_STEP SCALAR DOUBLE STATIC;                                 | THESIS_ALGORITHM
774 M|  DECLARE M3_FLOW_FINAL_STEP SCALAR DOUBLE STATIC;                                 | THESIS_ALGORITHM
775 M|  DECLARE I_LOOP INTEGER STATIC;                                                   | THESIS_ALGORITHM
776 M|  DECLARE I_STORE INTEGER STATIC;                                                  | THESIS_ALGORITHM
777 M|  DECLARE J_STORE INTEGER STATIC;                                                  | THESIS_ALGORITHM
778 M|  DECLARE PSI_MAG SCALAR DOUBLE STATIC;                                            | THESIS_ALGORITHM
779 M|  DECLARE ITER_FLAG BIT(1) STATIC;                                                 | THESIS_ALGORITHM
780 M|  DECLARE IIT INTEGER STATIC;                                                      | THESIS_ALGORITHM
781 M|  DECLARE COST SCALAR DOUBLE STATIC;                                               | THESIS_ALGORITHM
782 M|  DECLARE OVER_ITER_FLAG BIT(1) STATIC;                                            | THESIS_ALGORITHM
783 M|  DECLARE I_TIME_STORE INTEGER STATIC;                                             | THESIS_ALGORITHM
784 M|  DECLARE OVER_STEP BIT(1) STATIC;                                                 | THESIS_ALGORITHM
785 M|  DECLARE J_TIME INTEGER STATIC;                                                   | THESIS_ALGORITHM
786 M|  DECLARE FD_COUNT INTEGER STATIC;                                                 | THESIS_ALGORITHM
787 M|  DECLARE INTEG_L SCALAR DOUBLE STATIC;                                            | THESIS_ALGORITHM
788 M|  DECLARE L_FINAL SCALAR DOUBLE STATIC;                                            | THESIS_ALGORITHM
789 M|  DECLARE I_CL INTEGER STATIC;                                                     | THESIS_ALGORITHM
790 M|  DECLARE FD_FLAG BIT(1) STATIC;                                                   | THESIS_ALGORITHM
791 M|  DECLARE PRESENT_TIME_STEP SCALAR DOUBLE STATIC;                                  | THESIS_ALGORITHM
792 M|  DECLARE OIT_FLAG BIT(1) STATIC;                                                  | THESIS_ALGORITHM
793 M|  DECLARE T_FLAG BIT(1) AUTOMATIC;                                                 | THESIS_ALGORITHM
794 M|  DECLARE HELD_FS_MASS SCALAR DOUBLE STATIC;                                       | THESIS_ALGORITHM
795 M|  DECLARE U_TIME INTEGER STATIC;                                                   | THESIS_ALGORITHM
796 M|  DECLARE U MATRIX(NUM_CONTROLS, NUM_CONTROLS) DOUBLE STATIC;                      | THESIS_ALGORITHM
797 M|  DECLARE DJS_SCALE SCALAR DOUBLE STATIC;                                          | THESIS_ALGORITHM
```

```
STMT   SOURCE                                                                    CURRENT SCOPE
798 M| DECLARE HOLD_X ARRAY(STEP_DIM + 1) VECTOR(NUM_STATES) DOUBLE STATIC;     | THESIS_ALGORITHM
799 M| DECLARE HOLD_M ARRAY(STEP_DIM + 1) SCALAR DOUBLE STATIC;                 | THESIS_ALGORITHM
800 M| DECLARE HOLD_L_X ARRAY(STEP_DIM + 1) VECTOR(NUM_STATES) DOUBLE STATIC;   | THESIS_ALGORITHM
801 M| DECLARE HOLD_P VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;            | THESIS_ALGORITHM
802 M| DECLARE HOLD_U_T ARRAY(STEP_DIM + 1) SCALAR DOUBLE STATIC;               | THESIS_ALGORITHM
803 M| DECLARE HOLD_T ARRAY(NUM_TRANS_PTS) INTEGER STATIC;                      | THESIS_ALGORITHM
804 M| DECLARE HOLD_U ARRAY(STEP_DIM + 1) VECTOR(NUM_CONTROLS) DOUBLE STATIC;   | THESIS_ALGORITHM
805 M| DECLARE HOLD_POS_TIME ARRAY(NUM_TRANS_PTS) SCALAR DOUBLE STATIC;         | THESIS_ALGORITHM
806 M| DECLARE HOLD_NEG_TIME ARRAY(NUM_TRANS_PTS) SCALAR DOUBLE STATIC;         | THESIS_ALGORITHM
807 M| DECLARE STEP_I INTEGER STATIC;                                           | THESIS_ALGORITHM
808 M| DECLARE FIRST_PASS_PSI_MAG SCALAR DOUBLE STATIC;                         | THESIS_ALGORITHM
809 M| DECLARE L_SUB_U_1 VECTOR(NUM_CONTROLS) DOUBLE STATIC;                    | THESIS_ALGORITHM
810 M| DECLARE L_SUB_U_3 VECTOR(NUM_CONTROLS) DOUBLE STATIC;                    | THESIS_ALGORITHM
811 M| DECLARE L_SUB_U_L VECTOR(NUM_CONTROLS) DOUBLE STATIC;                    | THESIS_ALGORITHM
```

```
 STMT                                  SOURCE                                            CURRENT SCOPE

 812 M| U_COMPUTE:                                                                       | U_COMPUTE
 812 M| PROCEDURE;                                                                       | U_COMPUTE
 813 M|    DECLARE U_A INTEGER AUTOMATIC;                                                | U_COMPUTE
 814 M|    DECLARE U_B INTEGER AUTOMATIC;                                                | U_COMPUTE
 815 M|    DECLARE U_I INTEGER AUTOMATIC;                                                | U_COMPUTE
 816 M|    DECLARE U_SET BIT(1) AUTOMATIC;                                               | U_COMPUTE
 817 M|    DECLARE U_VAL SCALAR DOUBLE AUTOMATIC;                                        | U_COMPUTE
 818 M|    U_I = (2 NUM_TRANS_PTS) + 3;                                                  | U_COMPUTE
 819 M|    U_A = U_TIME - U_I;                                                           | U_COMPUTE
 820 M|    IF U_A < 2 THEN                                                               | U_COMPUTE
 821 M|       U_A = 2;                                                                   | U_COMPUTE
 822 M|    U_B = U_TIME + U_I;                                                           | U_COMPUTE
 823 M|    IF U_B > FLOOR((FINAL_STEP + 2) / 2) THEN                                     | U_COMPUTE
 824 M|       U_B = FLOOR((FINAL_STEP + 2) / 2);                                         | U_COMPUTE
     E|         .                                                                       |
 825 M|    U_SET = ON;                                                                   | U_COMPUTE
     E|                                 .                                               |
 826 M|    DO FOR U_I = U_A TO U_B WHILE U_SET = ON;                                     | U_COMPUTE
 827 M| 1     IF ((U_TIME_KEEP    > U_J_OLD_TIME      ) OR (U_TIME_KEEP    = U_J_OLD_TIME      )) THEN | U_COMPUTE
     S|                    U_I                U_TIME                  U_I                U_TIME         |
 828 M| 1        DO;                                                                     | U_COMPUTE
     E|             .                                                                   |
 829 M| 2           U_SET = OFF;                                                         | U_COMPUTE
 830 M| 2           U_VAL = (U_TIME_KEEP      - U_J_OLD_TIME      ) / (U_TIME_KEEP      - U_TIME_KEEP   ); | U_COMPUTE
     S|                                 U_I-1               U_TIME                U_I-1             U_I    |
 831 M| 2           U     = W          + ((W        - W          ) U_VAL);            | U_COMPUTE
     S|              1,1    U_I-1:1,1      U_I:1,1    U_I-1:1,1                        |
 832 M| 2           U     = W          + ((W        - W          ) U_VAL);            | U_COMPUTE
     S|              2,2    U_I-1:2,2      U_I:2,2    U_I-1:2,2                        |
 833 M| 1        END;                                                                    | U_COMPUTE
 834 M|    END;                                                                          | U_COMPUTE
```

```
STMT                              SOURCE                                        CURRENT SCOPE

 835 M| CLOSE U_COMPUTE;                                                        | U_COMPUTE

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
    NUM_TRANS_PTS, FINAL_STEP, U_TIME_KEEP, W

OUTER VARIABLES USED
    U_TIME, U_J_OLD_TIME, U*
```

```
STMT                                  SOURCE                                        CURRENT SCOPE

836 M| STATE_DERIVS:                                                                | STATE_DERIVS

836 M| PROCEDURE;                                                                   | STATE_DERIVS

837 M|    DECLARE VEHICLE_MASS SCALAR DOUBLE AUTOMATIC;                             | STATE_DERIVS

838 M|    DECLARE X_R SCALAR DOUBLE AUTOMATIC;                                      | STATE_DERIVS

839 M|    VEHICLE_MASS = X_STORE        + X_STORE        + X_STORE        + SS_DRY_MASS; | STATE_DERIVS
    S|                          I_TIME:5         I_TIME:6         I_TIME:7          |

    E|                                                                              |
840 M|    IF STAGE_SEP = OFF THEN                                                   | STATE_DERIVS

841 M|       VEHICLE_MASS = VEHICLE_MASS + FIRST_STAGE_DRY_MASS;                    | STATE_DERIVS

842 M|    X_R = X_STORE        + EARTH_RADIUS;                                      | STATE_DERIVS
    S|                 I_TIME:1                                                     |

843 M|    X_DOT  = X_STORE        ;                                                 | STATE_DERIVS
    S|         1          I_TIME:2                                                  |

844 M|    X_DOT  = (X_R X_STORE         X_STORE        ) + (NET_R_FORCE / VEHICLE_MASS); | STATE_DERIVS
    S|         2               I_TIME:4        I_TIME:4                             |

845 M|    X_DOT  = X_STORE        ;                                                 | STATE_DERIVS
    S|         3          I_TIME:4                                                  |

846 M|    X_DOT  = ((-2. X_STORE         X_STORE        ) / X_R) + (NET_THETA_FORCE / (VEHICLE_MASS X_R)); | STATE_DERIVS
    S|         4               I_TIME:2        I_TIME:4                             |

847 M|    X_DOT  = -TJ_FUEL_FLOW;                                                   | STATE_DERIVS
    S|         5                                                                    |

848 M|    X_DOT  = -SCRJ_FUEL_FLOW;                                                 | STATE_DERIVS
    S|         6                                                                    |

849 M|    X_DOT  = -SS_FUEL_FLOW;                                                   | STATE_DERIVS
    S|         7                                                                    |

850 M| CLOSE STATE_DERIVS;                                                          | STATE_DERIVS

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
   X_STORE, EARTH_RADIUS

OUTER VARIABLES USED
   I_TIME, SS_DRY_MASS, STAGE_SEP, FIRST_STAGE_DRY_MASS, X_DOT*, NET_R_FORCE, NET_THETA_FORCE, TJ_FUEL_FLOW, SCRJ_FUEL_FLOW
   SS_FUEL_FLOW
```

```
STMT                                    SOURCE                                              CURRENT SCOPE

 851 M| HAM_SUB_U:                                                                          | HAM_SUB_U

 851 M| PROCEDURE;                                                                          | HAM_SUB_U

 852 M|    IF I_TIME ¬= 1 THEN                                                              | HAM_SUB_U

 853 M|       DO;                                                                           | HAM_SUB_U

 854 M| 1        IF I_CL = 1 THEN                                                           | HAM_SUB_U

     E|             -                -            *            -                            |
 855 M| 1           H_SUB_U        = (LAMBDA       G_MAT       ) + L_SUB_U_1;               | HAM_SUB_U
     S|                    J_TIME:         J_TIME:      J_TIME:                             |

 856 M| 1        ELSE                                                                       | HAM_SUB_U

     E|             -                -            *            -                            |
 856 M| 1           H_SUB_U        = (LAMBDA       G_MAT       ) + L_SUB_U_3;               | HAM_SUB_U
     S|                    J_TIME:         J_TIME:      J_TIME:                             |

 857 M|       END;                                                                          | HAM_SUB_U

 858 M|    ELSE                                                                             | HAM_SUB_U

     E|       -          -        *          -                                              |
 858 M|       H_SUB_U  = (LAMBDA  G_MAT  ) + L_SUB_U_L;                                     | HAM_SUB_U
     S|              1:         1:     1:                                                   |

 859 M| CLOSE HAM_SUB_U;                                                                    | HAM_SUB_U

**** B L O C K   S U M M A R Y ****

OUTER VARIABLES USED
   I_TIME, I_CL, J_TIME, H_SUB_U*, LAMBDA, G_MAT, L_SUB_U_1, L_SUB_U_3, L_SUB_U_L
```

```
STMT                                    SOURCE                                                       CURRENT SCOPE

 860 M| DELTA_U_V_CALC:                                                                            | DELTA_U_V_CALC
 860 M| PROCEDURE;                                                                                 | DELTA_U_V_CALC
 861 M|    DECLARE B MATRIX(NUM_CONSTRAINTS, NUM_CONSTRAINTS) DOUBLE STATIC;                       | DELTA_U_V_CALC
 862 M|    DECLARE I_VEC_1 VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                                  | DELTA_U_V_CALC
 863 M|    DECLARE I_VEC_2 VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                                  | DELTA_U_V_CALC
 864 M|    DECLARE I_VEC_3 VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                                  | DELTA_U_V_CALC
 865 M|    DECLARE I_MAT MATRIX(NUM_CONSTANT_PARAMETERS + NUM_TRANS_PTS, NUM_CONSTRAINTS) DOUBLE STATIC; | DELTA_U_V_CALC
 866 M|    DECLARE J_MAT MATRIX(NUM_CONTROLS, NUM_CONSTRAINTS) DOUBLE STATIC;                      | DELTA_U_V_CALC
 867 M|    DECLARE C_SUB_J SCALAR DOUBLE STATIC;                                                   | DELTA_U_V_CALC
 868 M|    DECLARE PSI_VAL SCALAR DOUBLE AUTOMATIC;                                                | DELTA_U_V_CALC
 869 M|    DECLARE J_DU INTEGER STATIC;                                                            | DELTA_U_V_CALC
 870 M|    DECLARE IN_I INTEGER AUTOMATIC;                                                         | DELTA_U_V_CALC
 871 M|    DECLARE IN_J INTEGER AUTOMATIC;                                                         | DELTA_U_V_CALC
 872 M|    DECLARE IN_K INTEGER AUTOMATIC;                                                         | DELTA_U_V_CALC
 873 M|    DECLARE C_SUB_PSI_KEEP SCALAR DOUBLE STATIC;                                            | DELTA_U_V_CALC
 874 M|    DECLARE B_INT_1 ARRAY(NUM_CONSTRAINTS) SCALAR DOUBLE AUTOMATIC;                         | DELTA_U_V_CALC
 875 M|    DECLARE B_INT_2 ARRAY(NUM_CONSTRAINTS) SCALAR DOUBLE AUTOMATIC;                         | DELTA_U_V_CALC
 876 M|    DECLARE MAX_B SCALAR DOUBLE AUTOMATIC;                                                  | DELTA_U_V_CALC
 877 M|    DECLARE B_HOLD SCALAR DOUBLE AUTOMATIC;                                                 | DELTA_U_V_CALC
 878 M|    IF STEP_I = 1 THEN                                                                      | DELTA_U_V_CALC
 879 M|       DO;                                                                                  | DELTA_U_V_CALC
     E|          *       *        * *          *                                                   |
 880 M| 1        B = I_PSI_PSI + (M V TRANSPOSE(M));                                               | DELTA_U_V_CALC
 881 M| 1        DO FOR IN_K = 1 TO NUM_CONSTRAINTS;                                               | DELTA_U_V_CALC
 882 M| 2           B_INT_1     = IN_K;                                                            | DELTA_U_V_CALC
     S|                  IN_K                                                                      |
 883 M| 2           B_INT_2     = IN_K;                                                            | DELTA_U_V_CALC
     S|                  IN_K                                                                      |
 884 M| 2           MAX_B = B          ;                                                          | DELTA_U_V_CALC
     S|                    IN_K,IN_K                                                               |
```

```
STMT                                  SOURCE                                          CURRENT SCOPE

 885 M| 2         DO FOR IN_I = IN_K TO NUM_CONSTRAINTS;                             | DELTA_U_V_CALC

 886 M| 3            DO FOR IN_J = IN_K TO NUM_CONSTRAINTS;                          | DELTA_U_V_CALC

 887 M| 4               IF ABS(MAX_B) < ABS(B         ) THEN                         | DELTA_U_V_CALC
     S|                                      IN_I,IN_J                               |

 888 M| 4                  DO;                                                       | DELTA_U_V_CALC

 889 M| 5                     MAX_B = B         ;                                    | DELTA_U_V_CALC
     S|                                IN_I,IN_J                                     |

 890 M| 5                     B_INT_1     = IN_I;                                    | DELTA_U_V_CALC
     S|                              IN_K                                            |

 891 M| 5                     B_INT_2     = IN_J;                                    | DELTA_U_V_CALC
     S|                              IN_K                                            |

 892 M| 4                  END;                                                      | DELTA_U_V_CALC

 893 M| 3            END;                                                            | DELTA_U_V_CALC

 894 M| 2         END;                                                               | DELTA_U_V_CALC

 895 M| 2         IN_J = B_INT_1    ;                                                | DELTA_U_V_CALC
     S|                        IN_K                                                  |

 896 M| 2         IF B_INT_1     > IN_K THEN                                         | DELTA_U_V_CALC
     S|                    IN_K                                                      |

 897 M| 2            DO FOR IN_I = 1 TO NUM_CONSTRAINTS;                             | DELTA_U_V_CALC

 898 M| 3               B_HOLD = -B         ;                                        | DELTA_U_V_CALC
     S|                           IN_K,IN_I                                          |

 899 M| 3               B           = B         ;                                    | DELTA_U_V_CALC
     S|                  IN_K,IN_I    IN_J,IN_I                                      |

 900 M| 3               B           = B_HOLD;                                        | DELTA_U_V_CALC
     S|                  IN_J,IN_I                                                   |

 901 M| 2            END;                                                            | DELTA_U_V_CALC

 902 M| 2         IN_I = B_INT_2    ;                                                | DELTA_U_V_CALC
     S|                        IN_K                                                  |

 903 M| 2         IF B_INT_2     > IN_K THEN                                         | DELTA_U_V_CALC
     S|                    IN_K                                                      |

 904 M| 2            DO FOR IN_J = 1 TO NUM_CONSTRAINTS;                             | DELTA_U_V_CALC

 905 M| 3               B_HOLD = -B         ;                                        | DELTA_U_V_CALC
     S|                           IN_J,IN_K                                          |
```

```
STMT                                   SOURCE                                          CURRENT SCOPE

906 M| 3            B           = B          ;                                        | DELTA_U_V_CALC
    S|               IN_J,IN_K    IN_J,IN_I                                           |

907 M| 3            B           = B_HOLD;                                             | DELTA_U_V_CALC
    S|               IN_J,IN_I                                                        |

908 M| 2         END;                                                                 | DELTA_U_V_CALC

909 M| 2      DO FOR IN_I = 1 TO NUM_CONSTRAINTS;                                     | DELTA_U_V_CALC

910 M| 3         IF IN_I ¬= IN_K THEN                                                 | DELTA_U_V_CALC

911 M| 3            B           = -B          / B          ;                          | DELTA_U_V_CALC
    S|               IN_I,IN_K     IN_I,IN_K     IN_K,IN_K                            |

912 M| 2      END;                                                                    | DELTA_U_V_CALC

913 M| 2      DO FOR IN_I = 1 TO NUM_CONSTRAINTS;                                     | DELTA_U_V_CALC

914 M| 3         IF IN_I ¬= IN_K THEN                                                 | DELTA_U_V_CALC

915 M| 3            DO FOR IN_J = 1 TO NUM_CONSTRAINTS;                               | DELTA_U_V_CALC

916 M| 4               IF IN_J ¬= IN_K THEN                                           | DELTA_U_V_CALC

917 M| 4                  B           = (B          B          ) + B          ;       | DELTA_U_V_CALC
    S|                     IN_I,IN_J      IN_I,IN_K  IN_K,IN_J      IN_I,IN_J         |

918 M| 3            END;                                                              | DELTA_U_V_CALC

919 M| 2      END;                                                                    | DELTA_U_V_CALC

920 M| 2      DO FOR IN_J = 1 TO NUM_CONSTRAINTS;                                     | DELTA_U_V_CALC

921 M| 3         IF IN_J ¬= IN_K THEN                                                 | DELTA_U_V_CALC

922 M| 3            B           = B          / B          ;                           | DELTA_U_V_CALC
    S|               IN_K,IN_J    IN_K,IN_J    IN_K,IN_K                              |

923 M| 2      END;                                                                    | DELTA_U_V_CALC

924 M| 2      B           = 1. / B          ;                                         | DELTA_U_V_CALC
    S|         IN_K,IN_K          IN_K,IN_K                                           |

925 M| 1   END;                                                                       | DELTA_U_V_CALC

926 M| 1   DO WHILE IN_K > 1.;                                                        | DELTA_U_V_CALC

927 M| 2      IN_K = IN_K - 1;                                                        | DELTA_U_V_CALC

928 M| 2      IN_I = B_INT_1     ;                                                    | DELTA_U_V_CALC
    S|                    IN_K                                                        |

929 M| 2      IF IN_I > IN_K THEN                                                     | DELTA_U_V_CALC
```

```
STMT                                  SOURCE                                            CURRENT SCOPE
 930 M| 2          DO FOR IN_J = 1 TO NUM_CONSTRAINTS;                                  | DELTA_U_V_CALC
 931 M| 3             B_HOLD = B          ;                                             | DELTA_U_V_CALC
     S|                        IN_J,IN_K                                                |
 932 M| 3             B          = -B          ;                                        | DELTA_U_V_CALC
     S|                IN_J,IN_K     IN_J,IN_I                                          |
 933 M| 3             B          = B_HOLD;                                              | DELTA_U_V_CALC
     S|                IN_J,IN_I                                                        |
 934 M| 2          END;                                                                 | DELTA_U_V_CALC
 935 M| 2       IN_J = B_INT_2    ;                                                      | DELTA_U_V_CALC
     S|                     IN_K                                                        |
 936 M| 2       IF IN_J > IN_K THEN                                                     | DELTA_U_V_CALC
 937 M| 2          DO FOR IN_I = 1 TO NUM_CONSTRAINTS;                                  | DELTA_U_V_CALC
 938 M| 3             B_HOLD = B          ;                                             | DELTA_U_V_CALC
     S|                        IN_K,IN_I                                                |
 939 M| 3             B          = -B          ;                                        | DELTA_U_V_CALC
     S|                IN_K,IN_I     IN_J,IN_I                                          |
 940 M| 3             B          = B_HOLD;                                              | DELTA_U_V_CALC
     S|                IN_J,IN_I                                                        |
 941 M| 2          END;                                                                 | DELTA_U_V_CALC
 942 M| 1    END;                                                                       | DELTA_U_V_CALC
 943 M| 1    IF ITERATION = 1 THEN                                                      | DELTA_U_V_CALC
 944 M| 1       C_SUB_PSI_KEEP = PSI_START;                                             | DELTA_U_V_CALC
 945 M| 1    ELSE                                                                       | DELTA_U_V_CALC
 945 M| 1       DO;                                                                     | DELTA_U_V_CALC
 946 M| 2          PSI_VAL = PSI_START SQRT(FIRST_PSI_MAG / PSI_MAG);                   | DELTA_U_V_CALC
 947 M| 2          IF PSI_VAL > 1. THEN                                                 | DELTA_U_V_CALC
 948 M| 2             PSI_VAL = 1.;                                                     | DELTA_U_V_CALC
 949 M| 2          IF PSI_VAL < PSI_START THEN                                          | DELTA_U_V_CALC
 950 M| 2             C_SUB_PSI_KEEP = PSI_START;                                       | DELTA_U_V_CALC
 951 M| 2          ELSE                                                                 | DELTA_U_V_CALC
 951 M| 2             C_SUB_PSI_KEEP = PSI_VAL;                                         | DELTA_U_V_CALC
```

```
 STMT                                       SOURCE                                                        CURRENT SCOPE
 952 M| 1         END;                                                                                   | DELTA_U_V_CALC
     E|         -      * -                                                                               |
 953 M| 1       I_VEC_1 = B PSI;                                                                         | DELTA_U_V_CALC
     E|         -      * *     -                                                                         |
 954 M| 1       I_VEC_3 = M V SMALL_G_VEC;                                                               | DELTA_U_V_CALC
     E|         -      *    -          -                                                                 |
 955 M| 1       I_VEC_2 = B (I_PSI_J + I_VEC_3);                                                         | DELTA_U_V_CALC
     E|         *     *          *                                                                       |
 956 M| 1       I_MAT = V TRANSPOSE(M);                                                                  | DELTA_U_V_CALC
 957 M|       END;                                                                                       | DELTA_U_V_CALC
 958 M|     C_SUB_PSI = STEP_SCALE_PSI C_SUB_PSI_KEEP;                                                   | DELTA_U_V_CALC
 959 M|     DJS_SCALE = DJS STEP_SCALE_J;                                                                | DELTA_U_V_CALC
 960 M|     IF FIRST_PASS_PSI_MAG < (PSI_COST_MIX FIRST_PSI_MAG) THEN                                    | DELTA_U_V_CALC
     E|                                             -          -         -                       -      |
 961 M|       C_SUB_J = (-DJS_SCALE - (C_SUB_PSI ((I_PSI_J + I_VEC_3) . I_VEC_1))) / (I_J_J - ((I_PSI_J + | DELTA_U_V_CALC
     E|       -          -           -          *     -                                                  |
 961 M|       I_VEC_3) . I_VEC_2) + (SMALL_G_VEC . (V SMALL_G_VEC)));                                    | DELTA_U_V_CALC
 962 M|     ELSE                                                                                         | DELTA_U_V_CALC
 962 M|       C_SUB_J = 0.;                                                                              | DELTA_U_V_CALC
 963 M|     DO FOR J_DU = 1 TO FLOOR(FINAL_STEP / 2);                                                    | DELTA_U_V_CALC
     E|       *                  *             *                                                         |
 964 M| 1     J_MAT = TRANSPOSE(G_MAT      ) CAP_LAMBDA_1      ;                                         | DELTA_U_V_CALC
     S|                             J_DU:                J_DU:                                           |
 965 M| 1     U_TIME = J_DU;                                                                             | DELTA_U_V_CALC
 966 M| 1     CALL U_COMPUTE;                                                                            | DELTA_U_V_CALC
     E|         -            *             *      -                -                *      -            |
 967 M| 1     DELTA_U       = U ((-C_SUB_PSI (J_MAT I_VEC_1)) - (C_SUB_J (H_SUB_U       - (J_MAT I_VEC_2)))); | DELTA_U_V_CALC
     S|              J_DU:                                              J_DU:                            |
 968 M|     END;                                                                                         | DELTA_U_V_CALC
     E|       -                   *     -               *      -             *     -                     |
 969 M|     DELTA_V = (-C_SUB_PSI I_MAT I_VEC_1) - (C_SUB_J ((V SMALL_G_VEC) - (I_MAT I_VEC_2)));        | DELTA_U_V_CALC
 970 M| CLOSE DELTA_U_V_CALC;                                                                            | DELTA_U_V_CALC

**** B L O C K   S U M M A R Y ****
```

```
STMT                                   SOURCE                                   CURRENT SCOPE

OUTER PROCEDURES CALLED
    U_COMPUTE

COMPOOL VARIABLES USED
    NUM_CONSTRAINTS, NUM_CONSTANT_PARAMETERS, NUM_TRANS_PTS, NUM_CONTROLS, V, ITERATION, PSI_START, FIRST_PSI_MAG, STEP_SCALE_PSI
    STEP_SCALE_J, PSI_COST_MIX, FINAL_STEP

OUTER VARIABLES USED
    STEP_I, I_FSI_PSI, M, PSI_MAG, PSI, SMALL_G_VEC, I_PSI_J, C_SUB_PSI*, DJS_SCALE*, DJS, FIRST_PASS_PSI_MAG, DJS_SCALE, C_SUB_PSI
    I_J_J, G_MAT, CAP_LAMBDA_1, U_TIME*, DELTA_U*, U, H_SUB_U, DELTA_V*
```

```
STMT                              SOURCE                                             CURRENT SCOPE
 971 M| RUNGE_KUTTA:                                                                 | RUNGE_KUTTA
 971 M| PROCEDURE;                                                                   | RUNGE_KUTTA
 972 M|    DECLARE K0 MATRIX(10, NUM_CONSTRAINTS) DOUBLE STATIC;                     | RUNGE_KUTTA
 973 M|    DECLARE K1 MATRIX(10, NUM_CONSTRAINTS) DOUBLE STATIC;                     | RUNGE_KUTTA
 974 M|    DECLARE K2 MATRIX(10, NUM_CONSTRAINTS) DOUBLE STATIC;                     | RUNGE_KUTTA
 975 M|    DECLARE K3 MATRIX(10, NUM_CONSTRAINTS) DOUBLE STATIC;                     | RUNGE_KUTTA
 976 M|    DECLARE FIRST_RK_VAL_N MATRIX(10, NUM_CONSTRAINTS) DOUBLE STATIC;         | RUNGE_KUTTA
     C|  THE ABOVE VARIABLES HAVE THE NUMBER OF ROWS EQUAL MAX(                      | RUNGE_KUTTA
     C|  NUM_STATES+1,NUM_CONSTANT_PARAMETERS,NUM_CONSTRAINTS)                       | RUNGE_KUTTA
 977 M|    DECLARE F1 MATRIX(NUM_STATES, NUM_STATES) DOUBLE STATIC;                  | RUNGE_KUTTA
 978 M|    DECLARE F2 MATRIX(NUM_STATES, NUM_STATES) DOUBLE STATIC;                  | RUNGE_KUTTA
 979 M|    DECLARE F3 MATRIX(NUM_STATES, NUM_STATES) DOUBLE STATIC;                  | RUNGE_KUTTA
 980 M|    DECLARE F4 MATRIX(NUM_STATES, NUM_STATES) DOUBLE STATIC;                  | RUNGE_KUTTA
 981 M|    DECLARE F5 MATRIX(NUM_STATES, NUM_STATES) DOUBLE STATIC;                  | RUNGE_KUTTA
 982 M|    DECLARE KA MATRIX(NUM_STATES, NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;     | RUNGE_KUTTA
 983 M|    DECLARE KB MATRIX(NUM_STATES, NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;     | RUNGE_KUTTA
 984 M|    DECLARE KC MATRIX(NUM_STATES, NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;     | RUNGE_KUTTA
 985 M|    DECLARE KD MATRIX(NUM_STATES, NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;     | RUNGE_KUTTA
 986 M|    DECLARE KE MATRIX(NUM_STATES, NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;     | RUNGE_KUTTA
 987 M|    DECLARE M_1 MATRIX(NUM_CONSTRAINTS, NUM_CONTROLS) DOUBLE AUTOMATIC;       | RUNGE_KUTTA
 988 M|    DECLARE I_RK INTEGER STATIC;                                              | RUNGE_KUTTA
 989 M|    DECLARE J_RK INTEGER STATIC;                                              | RUNGE_KUTTA
 990 M|    DECLARE RK_D_VAL SCALAR DOUBLE STATIC;                                    | RUNGE_KUTTA
 991 M|    DECLARE RK_STEP INTEGER STATIC;                                           | RUNGE_KUTTA
 992 M|    DECLARE VAL_1 INTEGER STATIC;                                             | RUNGE_KUTTA
 993 M|    DECLARE START_RK_COLUMNS INTEGER STATIC;                                  | RUNGE_KUTTA
```

```
STMT                                  SOURCE                                          CURRENT SCOPE
 994 M| RK_DERIV:                                                                     | RK_DERIV
 994 M| PROCEDURE;                                                                    | RK_DERIV
 995 M|    DECLARE L_SUB_X VECTOR(NUM_STATES) DOUBLE STATIC;                          | RK_DERIV
 996 M|    DECLARE L_SUB_P VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;             | RK_DERIV
 997 M|    DECLARE L_SUB_P_1 VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;           | RK_DERIV
 998 M|    DECLARE L_SUB_P_3 VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;           | RK_DERIV
 999 M|    DECLARE L_SUB_P_5 VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;           | RK_DERIV
1000 M|    DECLARE DFR_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;        | RK_DERIV
1001 M|    DECLARE DFTHETA_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;    | RK_DERIV
1002 M|    DECLARE DM1DOT_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;     | RK_DERIV
1003 M|    DECLARE DM2DOT_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;     | RK_DERIV
1004 M|    DECLARE DM3DOT_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;     | RK_DERIV
1005 M|    DECLARE DP_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;         | RK_DERIV
1006 M|    DECLARE DN_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;         | RK_DERIV
1007 M|    DECLARE DT_DP ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC;         | RK_DERIV
1008 M|    DECLARE DMASSFLUX_DP1 ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC; | RK_DERIV
1009 M|    DECLARE DMASSFLUX_DP2 ARRAY(NUM_CONSTANT_PARAMETERS) SCALAR DOUBLE STATIC; | RK_DERIV
1010 M|    DECLARE DFR_DR SCALAR DOUBLE STATIC;                                       | RK_DERIV
1011 M|    DECLARE DFTHETA_DR SCALAR DOUBLE STATIC;                                   | RK_DERIV
1012 M|    DECLARE DM1DOT_DR SCALAR DOUBLE STATIC;                                    | RK_DERIV
1013 M|    DECLARE DM2DOT_DR SCALAR DOUBLE STATIC;                                    | RK_DERIV
1014 M|    DECLARE DP_DR SCALAR DOUBLE STATIC;                                        | RK_DERIV
1015 M|    DECLARE DN_DR SCALAR DOUBLE STATIC;                                        | RK_DERIV
1016 M|    DECLARE DT_DR SCALAR DOUBLE STATIC;                                        | RK_DERIV
1017 M|    DECLARE DMASSFLUX_DR1 SCALAR DOUBLE STATIC;                                | RK_DERIV
1018 M|    DECLARE DMASSFLUX_DR2 SCALAR DOUBLE STATIC;                                | RK_DERIV
1019 M|    DECLARE DP_DUR SCALAR DOUBLE STATIC;                                       | RK_DERIV
1020 M|    DECLARE DN_DUR SCALAR DOUBLE STATIC;                                       | RK_DERIV
```

```
STMT                                  SOURCE                                              CURRENT SCOPE
1021 M|   DECLARE DT_DUR SCALAR DOUBLE STATIC;                                           | RK_DERIV
1022 M|   DECLARE DMASSFLUX_DUR1 SCALAR DOUBLE STATIC;                                   | RK_DERIV
1023 M|   DECLARE DMASSFLUX_DUR2 SCALAR DOUBLE STATIC;                                   | RK_DERIV
1024 M|   DECLARE DFR_DMI SCALAR DOUBLE STATIC;                                          | RK_DERIV
1025 M|   DECLARE DFR_DUR SCALAR DOUBLE STATIC;                                          | RK_DERIV
1026 M|   DECLARE DFTHETA_DUR SCALAR DOUBLE STATIC;                                      | RK_DERIV
1027 M|   DECLARE DM1DOT_DUR SCALAR DOUBLE STATIC;                                       | RK_DERIV
1028 M|   DECLARE DM2DOT_DUR SCALAR DOUBLE STATIC;                                       | RK_DERIV
1029 M|   DECLARE DFR_DTDOT SCALAR DOUBLE STATIC;                                        | RK_DERIV
1030 M|   DECLARE DFTHETA_DTDOT SCALAR DOUBLE STATIC;                                    | RK_DERIV
1031 M|   DECLARE DM1DOT_DTDOT SCALAR DOUBLE STATIC;                                     | RK_DERIV
1032 M|   DECLARE DM2DOT_DTDOT SCALAR DOUBLE STATIC;                                     | RK_DERIV
1033 M|   DECLARE DP_DTDOT SCALAR DOUBLE STATIC;                                         | RK_DERIV
1034 M|   DECLARE DN_DTDOT SCALAR DOUBLE STATIC;                                         | RK_DERIV
1035 M|   DECLARE DT_DTDOT SCALAR DOUBLE STATIC;                                         | RK_DERIV
1036 M|   DECLARE DMASSFLUX_DTDOT1 SCALAR DOUBLE STATIC;                                 | RK_DERIV
1037 M|   DECLARE DMASSFLUX_DTDOT2 SCALAR DOUBLE STATIC;                                 | RK_DERIV
1038 M|   DECLARE DDELTA_DR SCALAR DOUBLE STATIC;                                        | RK_DERIV
1039 M|   DECLARE DDELTA_DUR SCALAR DOUBLE STATIC;                                       | RK_DERIV
1040 M|   DECLARE DDELTA_DTDOT SCALAR DOUBLE STATIC;                                     | RK_DERIV
1041 M|   DECLARE S2 SCALAR DOUBLE STATIC;                                               | RK_DERIV
1042 M|   DECLARE S3 SCALAR DOUBLE STATIC;                                               | RK_DERIV
1043 M|   DECLARE S5 SCALAR DOUBLE STATIC;                                               | RK_DERIV
1044 M|   DECLARE S6 SCALAR DOUBLE STATIC;                                               | RK_DERIV
1045 M|   DECLARE COS_DELTA SCALAR DOUBLE STATIC;                                        | RK_DERIV
1046 M|   DECLARE SIN_DELTA SCALAR DOUBLE STATIC;                                        | RK_DERIV
1047 M|   DECLARE SG SCALAR DOUBLE STATIC;                                               | RK_DERIV
1048 M|   DECLARE L_CONST SCALAR DOUBLE STATIC;                                          | RK_DERIV
```

```
STMT                                       SOURCE                                           CURRENT SCOPE
1049 M| DECLARE LAMBDA_HOLD VECTOR(NUM_STATES) DOUBLE STATIC;                              | RK_DERIV
1050 M| DECLARE CAP_LAMBDA_1_HOLD MATRIX(NUM_STATES, NUM_CONSTRAINTS) DOUBLE STATIC;        | RK_DERIV
1051 M| DECLARE KEEP_MASS SCALAR DOUBLE STATIC;                                            | RK_DERIV
1052 M| DECLARE SR SCALAR DOUBLE STATIC;                                                   | RK_DERIV
1053 M| DECLARE K_RK INTEGER STATIC;                                                       | RK_DERIV
1054 M| DECLARE L_RK INTEGER AUTOMATIC;                                                    | RK_DERIV
1055 M| DECLARE I_FK INTEGER STATIC;                                                       | RK_DERIV
1056 M| DECLARE T_VAL SCALAR DOUBLE STATIC;                                                | RK_DERIV
1057 M| DECLARE SPEED_OF_SOUND_2 SCALAR DOUBLE STATIC;                                     | RK_DERIV
1058 M| DECLARE DM0_DR SCALAR DOUBLE STATIC;                                               | RK_DERIV
1059 M| DECLARE DM0_DUR SCALAR DOUBLE STATIC;                                              | RK_DERIV
1060 M| DECLARE DM0_DTDOT SCALAR DOUBLE STATIC;                                            | RK_DERIV
1061 M| DECLARE DM2_DBETA SCALAR DOUBLE STATIC;                                            | RK_DERIV
1062 M| DECLARE DT2_DBETA SCALAR DOUBLE STATIC;                                            | RK_DERIV
1063 M| DECLARE DRO2_DBETA SCALAR DOUBLE STATIC;                                           | RK_DERIV
1064 M| DECLARE DU2_DM2 SCALAR DOUBLE STATIC;                                              | RK_DERIV
1065 M| DECLARE DU2_DT2 SCALAR DOUBLE STATIC;                                              | RK_DERIV
1066 M| DECLARE SIN_BETA SCALAR DOUBLE STATIC;                                             | RK_DERIV
1067 M| DECLARE COS_BETA SCALAR DOUBLE STATIC;                                             | RK_DERIV
1068 M| DECLARE SIN_2_BETA SCALAR DOUBLE STATIC;                                           | RK_DERIV
1069 M| DECLARE SIN_B_T SCALAR DOUBLE STATIC;                                              | RK_DERIV
1070 M| DECLARE COS_B_T SCALAR DOUBLE STATIC;                                              | RK_DERIV
1071 M| DECLARE OSC1 SCALAR DOUBLE STATIC;                                                 | RK_DERIV
1072 M| DECLARE DT2_DM0 SCALAR DOUBLE STATIC;                                              | RK_DERIV
1073 M| DECLARE DT2_DT0 SCALAR DOUBLE STATIC;                                              | RK_DERIV
1074 M| DECLARE DRO2_DM0 SCALAR DOUBLE STATIC;                                             | RK_DERIV
1075 M| DECLARE DRO2_DRO_0 SCALAR DOUBLE STATIC;                                           | RK_DERIV
1076 M| DECLARE DBETA_DR SCALAR DOUBLE STATIC;                                             | RK_DERIV
```

```
STMT                              SOURCE                                  CURRENT SCOPE
1077 M|   DECLARE DBETA_DUR SCALAR DOUBLE STATIC;                          | RK_DERIV
1078 M|   DECLARE DBETA_DTDOT SCALAR DOUBLE STATIC;                        | RK_DERIV
1079 M|   DECLARE DM2_DR SCALAR DOUBLE STATIC;                             | RK_DERIV
1080 M|   DECLARE DM2_DUR SCALAR DOUBLE STATIC;                            | RK_DERIV
1081 M|   DECLARE DM2_DTDOT SCALAR DOUBLE STATIC;                          | RK_DERIV
1082 M|   DECLARE DT2_DR SCALAR DOUBLE STATIC;                             | RK_DERIV
1083 M|   DECLARE DT2_DUR SCALAR DOUBLE STATIC;                            | RK_DERIV
1084 M|   DECLARE DT2_DTDOT SCALAR DOUBLE STATIC;                          | RK_DERIV
1085 M|   DECLARE DU2_DR SCALAR DOUBLE STATIC;                             | RK_DERIV
1086 M|   DECLARE DU2_DUR SCALAR DOUBLE STATIC;                            | RK_DERIV
1087 M|   DECLARE DU2_DTDOT SCALAR DOUBLE STATIC;                          | RK_DERIV
1088 M|   DECLARE DRO2_DR SCALAR DOUBLE STATIC;                            | RK_DERIV
1089 M|   DECLARE DRO2_DUR SCALAR DOUBLE STATIC;                           | RK_DERIV
1090 M|   DECLARE DRO2_DTDOT SCALAR DOUBLE STATIC;                         | RK_DERIV
1091 M|   DECLARE DISP1_DM2 SCALAR DOUBLE STATIC;                          | RK_DERIV
1092 M|   DECLARE DISP2_DM2 SCALAR DOUBLE STATIC;                          | RK_DERIV
1093 M|   DECLARE DT1_DR SCALAR DOUBLE STATIC;                             | RK_DERIV
1094 M|   DECLARE DT1_DUR SCALAR DOUBLE STATIC;                            | RK_DERIV
1095 M|   DECLARE DT1_DTDOT SCALAR DOUBLE STATIC;                          | RK_DERIV
1096 M|   DECLARE DTH2_DR SCALAR DOUBLE STATIC;                            | RK_DERIV
1097 M|   DECLARE DTH2_DUR SCALAR DOUBLE STATIC;                           | RK_DERIV
1098 M|   DECLARE DTH2_DTDOT SCALAR DOUBLE STATIC;                         | RK_DERIV
1099 M|   DECLARE DCL1_DR SCALAR DOUBLE STATIC;                            | RK_DERIV
1100 M|   DECLARE DCL2_DR SCALAR DOUBLE STATIC;                            | RK_DERIV
1101 M|   DECLARE DCD1_DR SCALAR DOUBLE STATIC;                            | RK_DERIV
1102 M|   DECLARE DCD2_DR SCALAR DOUBLE STATIC;                            | RK_DERIV
1103 M|   DECLARE DCL1_DUR SCALAR DOUBLE STATIC;                           | RK_DERIV
1104 M|   DECLARE DCL2_DUR SCALAR DOUBLE STATIC;                           | RK_DERIV
```

```
STMT                                 SOURCE                                                  CURRENT SCOPE
1105 M|  DECLARE DCD1_DUR SCALAR DOUBLE STATIC;                                             | RK_DERIV
1106 M|  DECLARE DCD2_DUR SCALAR DOUBLE STATIC;                                             | RK_DERIV
1107 M|  DECLARE DCL1_DTDOT SCALAR DOUBLE STATIC;                                           | RK_DERIV
1108 M|  DECLARE DCL2_DTDOT SCALAR DOUBLE STATIC;                                           | RK_DERIV
1109 M|  DECLARE DCD1_DTDOT SCALAR DOUBLE STATIC;                                           | RK_DERIV
1110 M|  DECLARE DCD2_DTDOT SCALAR DOUBLE STATIC;                                           | RK_DERIV
1111 M|  DECLARE V0_2 SCALAR DOUBLE STATIC;                                                 | RK_DERIV
1112 M|  DECLARE DL_DR SCALAR DOUBLE STATIC;                                                | RK_DERIV
1113 M|  DECLARE DL_DUR SCALAR DOUBLE STATIC;                                               | RK_DERIV
1114 M|  DECLARE DL_DTDOT SCALAR DOUBLE STATIC;                                             | RK_DERIV
1115 M|  DECLARE DD_DR SCALAR DOUBLE STATIC;                                                | RK_DERIV
1116 M|  DECLARE DD_DUR SCALAR DOUBLE STATIC;                                               | RK_DERIV
1117 M|  DECLARE DD_DTDOT SCALAR DOUBLE STATIC;                                             | RK_DERIV
1118 M|  DECLARE SIN_A SCALAR DOUBLE STATIC;                                                | RK_DERIV
1119 M|  DECLARE COS_A SCALAR DOUBLE STATIC;                                                | RK_DERIV
1120 M|  DECLARE DT2_DM2 SCALAR DOUBLE STATIC;                                              | RK_DERIV
1121 M|  DECLARE DRO2_DM2 SCALAR DOUBLE STATIC;                                             | RK_DERIV
1122 M|  DECLARE DBETA_DM0 SCALAR DOUBLE STATIC;                                            | RK_DERIV
1123 M|  DECLARE DM2_DM0 SCALAR DOUBLE STATIC;                                              | RK_DERIV
1124 M|  DECLARE M2_2 SCALAR DOUBLE STATIC;                                                 | RK_DERIV
1125 M|  DECLARE DAR_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                     | RK_DERIV
1126 M|  DECLARE DAW1_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                    | RK_DERIV
1127 M|  DECLARE DAW2_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                    | RK_DERIV
1128 M|  DECLARE DTH1_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                    | RK_DERIV
1129 M|  DECLARE DTH2_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                    | RK_DERIV
1130 M|  DECLARE DTH3_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                    | RK_DERIV
1131 M|  DECLARE DAE1_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                    | RK_DERIV
1132 M|  DECLARE DAE2_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;                    | RK_DERIV
```

```
STMT                                  SOURCE                                          CURRENT SCOPE
1133 M|   DECLARE DG_DPI VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC;               | RK_DERIV
1134 M|   DECLARE DRO2_DP1 SCALAR DOUBLE STATIC;                                      | RK_DERIV
1135 M|   DECLARE DBETA_DP1 SCALAR DOUBLE STATIC;                                     | RK_DERIV
1136 M|   DECLARE DT2_DP1 SCALAR DOUBLE STATIC;                                       | RK_DERIV
1137 M|   DECLARE DM2_DP1 SCALAR DOUBLE STATIC;                                       | RK_DERIV
1138 M|   DECLARE DISP1_DP1 SCALAR DOUBLE STATIC;                                     | RK_DERIV
1139 M|   DECLARE DISP2_DP1 SCALAR DOUBLE STATIC;                                     | RK_DERIV
1140 M|   DECLARE DMCR_DP1 SCALAR DOUBLE STATIC;                                      | RK_DERIV
1141 M|   DECLARE DU2_DP1 SCALAR DOUBLE STATIC;                                       | RK_DERIV
1142 M|   DECLARE DL_DPI SCALAR DOUBLE AUTOMATIC;                                     | RK_DERIV
1143 M|   DECLARE DD_DPI SCALAR DOUBLE AUTOMATIC;                                     | RK_DERIV
1144 M|   DECLARE DM1DOT_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                      | RK_DERIV
1145 M|   DECLARE DM2DOT_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                      | RK_DERIV
1146 M|   DECLARE DM3DOT_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                      | RK_DERIV
1147 M|   DECLARE DFR_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                         | RK_DERIV
1148 M|   DECLARE DFTHETA_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                     | RK_DERIV
1149 M|   DECLARE DT_DUA VECTOR(NUM_CONTROLS) DOUBLE STATIC;                          | RK_DERIV
1150 M|   DECLARE L_SUB_U VECTOR(NUM_CONTROLS) DOUBLE STATIC;                         | RK_DERIV
1151 M|   DECLARE DL_DUA1 SCALAR DOUBLE STATIC;                                       | RK_DERIV
1152 M|   DECLARE DD_DUA1 SCALAR DOUBLE STATIC;                                       | RK_DERIV
1153 M|   DECLARE DPHI_DUA2 SCALAR DOUBLE STATIC;                                     | RK_DERIV
1154 M|   DECLARE NET_THRUST SCALAR DOUBLE STATIC;                                    | RK_DERIV
     E|          .                                                                    |
1155 M|   IF STATE_INTEGRATION_FLAG = ON THEN                                         | RK_DERIV
1156 M|      DO;                                                                      | RK_DERIV
1157 M| 1       IF I_RK = (NUM_STATES + 1) THEN                                        | RK_DERIV
1158 M| 1          DO;                                                                 | RK_DERIV
1159 M| 2             RK_D_VAL = 0.;                                                   | RK_DERIV
```

```
STMT                                    SOURCE                                          CURRENT SCOPE

1160 M| 2             IF DYNAMIC_PO > Q_D THEN                                         | RK_DERIV

     E|                                                 2                               |
1161 M| 2               RK_D_VAL = CAP_Q ((DYNAMIC_PO - Q_D) );                        | RK_DERIV

1162 M| 2             IF G_LOAD > G_D THEN                                             | RK_DERIV

     E|                                                                   2             |
1163 M| 2               RK_D_VAL = RK_D_VAL + (CAP_CA (((G_LOAD - G_D) GO) ));         | RK_DERIV

1164 M| 1           END;                                                               | RK_DERIV

1165 M| 1         ELSE                                                                 | RK_DERIV

1165 M| 1           RK_D_VAL = X_DOT     ;                                             | RK_DERIV
     S|                        I_RK                                                    |

1166 M|           END;                                                                 | RK_DERIV

1167 M|        ELSE                                                                    | RK_DERIV

1167 M|           DO;                                                                  | RK_DERIV

1168 M| 1         RK_STEP = RK_STEP + 1;                                               | RK_DERIV

     E|                .                                                               |
1169 M| 1         IF FIRST_DERIV_FLAG = ON THEN                                        | RK_DERIV

1170 M| 1           DO;                                                                | RK_DERIV

     E|                  .                                                             |
1171 M| 2             FIRST_DERIV_FLAG = OFF;                                          | RK_DERIV

     E|                   .            .                                               |
1172 M| 2             IF (LAMBDA_FLAG OR CAP_LAMBDA_1_FLAG) = ON THEN                  | RK_DERIV

1173 M| 2               DO FOR K_RK = 1 TO NUM_STATES;                                 | RK_DERIV

     E|                          .                                                     |
1174 M| 3                 IF LAMBDA_FLAG = ON THEN                                     | RK_DERIV

1175 M| 3                   LAMBDA_HOLD     = RK_VAL_N      ;                          | RK_DERIV
     S|                              K_RK             K_RK,1                           |

1176 M| 3                 ELSE                                                         | RK_DERIV

1176 M| 3                   DO FOR L_RK = 1 TO NUM_CONSTRAINTS;                        | RK_DERIV

1177 M| 4                     CAP_LAMBDA_1_HOLD           = RK_VAL_N           ;       | RK_DERIV
     S|                                      K_RK,L_RK             K_RK,L_RK           |

1178 M| 3                   END;                                                       | RK_DERIV

1179 M| 2               END;                                                           | RK_DERIV
```

```
STMT                                      SOURCE                                          CURRENT SCOPE
     E|           .                                                                       |
1180 M| 2      IF PARTIAL_DERIV_FLAG = ON THEN                                            | RK_DERIV
1181 M| 2         DO;                                                                     | RK_DERIV
     E|                .                                                                  |
1182 M| 3            PARTIAL_DERIV_FLAG = OFF;                                            | RK_DERIV
1183 M| 3            IF RK_STEP > VAL_1 THEN                                              | RK_DERIV
1184 M| 3               DO;                                                               | RK_DERIV
     E|                        .                  .                   .                   |
1185 M| 4                  IF (LAMBDA_FLAG OR CAP_LAMBDA_1_FLAG OR CAP_LAMBDA_2_FLAG) = ON THEN | RK_DERIV
1186 M| 4                     I_TIME = I_TIME - 1;                                        | RK_DERIV
1187 M| 4                  ELSE                                                           | RK_DERIV
1187 M| 4                     I_TIME = I_TIME - 2;                                        | RK_DERIV
1188 M| 4                  J_TIME = FLOOR((I_TIME + 2) / 2);                              | RK_DERIV
1189 M| 3               END;                                                              | RK_DERIV
     C|  COSTATE DERIVATIVES FOLLOW                                                       | RK_DERIV
     E|                .                                                                  |
1190 M| 3            IF LAMBDA_FLAG = ON THEN                                             | RK_DERIV
1191 M| 3               DO;                                                               | RK_DERIV
     E|                                       .                                           |
1192 M| 4                  IF ((I_CL = 2) AND (FD_FLAG = ON)) THEN                        | RK_DERIV
     E|                        .                                                          |
1193 M| 4                     FD_FLAG = OFF;                                              | RK_DERIV
1194 M| 4                  ELSE                                                           | RK_DERIV
1194 M| 4                     DO;                                                         | RK_DERIV
1195 M| 5                        SR = X_STORE         + EARTH_RADIUS;                     | RK_DERIV
     S|                                  I_TIME:1                                         |
1196 M| 5                        L_TIME = I_TIME - 1;                                     | RK_DERIV
1197 M| 5                        IF L_TIME = 0 THEN                                       | RK_DERIV
1198 M| 5                           L_TIME = 1;                                           | RK_DERIV
     E|                              .                                                    |
1199 M| 5                        OIT_FLAG = OFF;                                          | RK_DERIV
```

```
 STMT                              SOURCE                                               CURRENT SCOPE
      E|                   .                                                          |
 1200 M| 5              T_FLAG = OFF;                                                  | RK_DERIV
 1201 M| 5              IF MOD(I_TIME, 4) = 1 THEN                                     | RK_DERIV
 1202 M| 5                 DO FOR I_FK = 1 TO NUM_TRANS_PTS;                           | RK_DERIV
 1203 M| 6                    IF OMEGA_I_TIME      = I_TIME THEN                       | RK_DERIV
      S|                                     I_FK                                      |
 1204 M| 6                       DO;                                                   | RK_DERIV
      E|                            .                                                  |
 1205 M| 7                          T_FLAG = ON;                                       | RK_DERIV
 1206 M| 7                          IF OMEGA_I_TIME      = I_TIME_STORE THEN           | RK_DERIV
      S|                                          I_FK                                 |
      E|                                .                                              |
 1207 M| 7                             OIT_FLAG = ON;                                  | RK_DERIV
 1208 M| 6                       END;                                                  | RK_DERIV
 1209 M| 5                 END;                                                        | RK_DERIV
      E|                      .              .                                         |
 1210 M| 5              IF (T_FLAG AND NOT OIT_FLAG) = ON THEN                         | RK_DERIV
 1211 M| 5                 L_TIME = I_TIME;                                            | RK_DERIV
      E|                 -                                                             |
 1212 M| 5              L_SUB_X = 0.;                                                  | RK_DERIV
      E|                 -                                                             |
 1213 M| 5              L_SUB_P = 0.;                                                  | RK_DERIV
 1214 M| 5              CALL MODEL_DRIVER;                                             | RK_DERIV
 1215 M| 5              NET_THRUST = TURBOJET_THRUST + SCRAMJET_THRUST + ROCKET_THRUST; | RK_DERIV
 1216 M| 5              L_CONST = 2. CAP_CA (G_LOAD - G_D) G0 G0;                      | RK_DERIV
 1217 M| 5              FD_COUNT = FD_COUNT + 1;                                       | RK_DERIV
 1218 M| 5              IF FD_COUNT = 5 THEN                                           | RK_DERIV
 1219 M| 5                 FD_COUNT = 0;                                               | RK_DERIV
      E|                                                        .                      |
 1220 M| 5              IF ((I_TIME = OMEGA_I_TIME ) AND (OIT_FLAG = ON)) THEN         | RK_DERIV
      S|                                          3                                    |
 1221 M| 5                 DO;                                                         | RK_DERIV
```

```
STMT                                  SOURCE                                          CURRENT SCOPE

1222 M|  6                KEEP_MASS = NET_MASS      ;                               | RK_DERIV
     S|                                    I_TIME                                   |

1223 M|  6                NET_MASS       = NET_MASS       - HELD_FS_MASS;           | RK_DERIV
     S|                           I_TIME           I_TIME                           |

1224 M|  5             END;                                                         | RK_DERIV

1225 M|  5          IF G_LOAD > G_D THEN                                            | RK_DERIV

1226 M|  5             DO FOR I_FK = 5 TO 7;                                        | RK_DERIV

1227 M|  6                L_SUB_X     = -(G_LOAD L_CONST) / MD_MASS;                | RK_DERIV
     S|                          I_FK                                               |

1228 M|  5             END;                                                         | RK_DERIV

1229 M|  5          MO_2 = MO MO;                                                   | RK_DERIV

1230 M|  5          T_VAL = X_STORE          - EARTH_OMEGA;                         | RK_DERIV
     S|                          I_TIME:4                                           |

1231 M|  5          SPEED_OF_SOUND_2 = GAMMA0 R_0 T0;                               | RK_DERIV

1232 M|  5          DM0_DR = ((SR T_VAL T_VAL) / (M0 SPEED_OF_SOUND_2)) - ((M0 DT0_DR) | RK_DERIV

1232 M|  5          / (2. T0));                                                     | RK_DERIV

1233 M|  5          DM0_DUR = X_STORE          / (M0 SPEED_OF_SOUND_2);             | RK_DERIV
     S|                           I_TIME:2                                          |

1234 M|  5          DM0_DTDOT = (SR SR T_VAL) / (M0 SPEED_OF_SOUND_2);              | RK_DERIV

1235 M|  5          DM2_DBETA = 0.;                                                 | RK_DERIV

1236 M|  5          DT2_DBETA = 0.;                                                 | RK_DERIV

1237 M|  5          DRO2_DBETA = 0.;                                                | RK_DERIV

1238 M|  5          DT2_DM2 = 0.;                                                   | RK_DERIV

1239 M|  5          DRO2_DM2 = 0.;                                                  | RK_DERIV

1240 M|  5          DBETA_DM0 = 0.;                                                 | RK_DERIV

1241 M|  5          DM2_DM0 = 0.;                                                   | RK_DERIV

     E|                                                                             |
1242 M|  5          IF STAGE_SEP = OFF THEN                                         | RK_DERIV

1243 M|  5             DO;                                                          | RK_DERIV

1244 M|  6                DU2_DM2 = SQRT(GAMMA0 R_0 T2);                            | RK_DERIV

1245 M|  6                IF T2 ¬= 0. THEN                                          | RK_DERIV
```

```
STMT                          SOURCE                                          CURRENT SCOPE
1246 M| 6        DU2_DT2 = (M2 DU2_DM2) / (2. T2);                          | RK_DERIV
     E|                                                                     |
1247 M| 6     IF OBLIQUE_SHOCK_FLAG = ON THEN                               | RK_DERIV
1248 M| 6        DO;                                                        | RK_DERIV
1249 M| 7           SIN_BETA = SIN(BETA_NOSE_SHOCK);                        | RK_DERIV
1250 M| 7           COS_BETA = COS(BETA_NOSE_SHOCK);                        | RK_DERIV
1251 M| 7           SIN_2_BETA = SIN_BETA SIN_BETA;                         | RK_DERIV
1252 M| 7           SIN_B_T = SIN(BETA_NOSE_SHOCK - THETA_NOSE);            | RK_DERIV
1253 M| 7           COS_B_T = COS(BETA_NOSE_SHOCK - THETA_NOSE);            | RK_DERIV
1254 M| 7           OSC1 = SQRT(((GAMMA0 M0_2 SIN_2_BETA) - GAM1) / (1. +   | RK_DERIV
1254 M| 7           (GAM1 M0_2 SIN_2_BETA)));                               | RK_DERIV
1255 M| 7           DBETA_DM0 = (2. M0 GAM2 SIN_BETA COS_BETA) / ((M0_2     | RK_DERIV
1255 M| 7           M0_2 SIN_2_BETA ((2. GAMMA0 SIN_2_BETA) - GAM2)) + (    | RK_DERIV
1255 M| 7           M0_2 ((4. SIN_2_BETA) - GAM2)) - 2.);                   | RK_DERIV
1256 M| 7           IF SIN_B_T ¬= 0. THEN                                   | RK_DERIV
1257 M| 7              DO;                                                  | RK_DERIV
1258 M| 8                 DM2_DBETA = (-(GAM2 GAM2 M0_2 SIN_BETA COS_BETA   | RK_DERIV
1258 M| 8                 OSC1) / (4. SIN_B_T (((GAMMA0 M0_2 SIN_2_BETA) -  | RK_DERIV
     E|                        2                                            |
1258 M| 8                 GAM1) ))) - (COS_B_T / (SIN_B_T SIN_B_T OSC1));   | RK_DERIV
1259 M| 8                 DM2_DM0 = -(GAM2 GAM2 M0 SIN_2_BETA OSC1) / (4.   | RK_DERIV
     E|                                                           2         |
1259 M| 8                 SIN_B_T (((GAMMA0 M0_2 SIN_2_BETA) - GAM1) ));    | RK_DERIV
1260 M| 7              END;                                                 | RK_DERIV
1261 M| 7           IF ((SIN_2_BETA ¬= 0.) OR (SIN_BETA ¬= 0.)) THEN        | RK_DERIV
1262 M| 7              DT2_DBETA = (4. T0 GAM3 COS_BETA ((GAMMA0 M0_2 M0_2  | RK_DERIV
1262 M| 7              SIN_2_BETA SIN_2_BETA) + 1.)) / (GAM2 GAM2 M0_2      | RK_DERIV
1262 M| 7              SIN_2_BETA SIN_BETA);                                | RK_DERIV
1263 M| 7           IF COS_BETA ¬= 0. THEN                                  | RK_DERIV
```

```
STMT                          SOURCE                                                CURRENT SCOPE
1264 M| 7           DT2_DM0 = (DT2_DBETA SIN_BETA) / (COS_BETA M0);               | RK_DERIV
1265 M| 7         DT2_DT0 = T2 / T0;                                              | RK_DERIV
1266 M| 7         DRO2_DBETA = (4. RO_0 GAM2 M0_2 SIN_BETA COS_BETA) / (          | RK_DERIV
     E|                                              2                              |
1266 M| 7         ((GAM3 M0_2 SIN_2_BETA) + 2.) );                                | RK_DERIV
1267 M| 7         IF COS_BETA ¬= 0. THEN                                          | RK_DERIV
1268 M| 7           DRO2_DM0 = (DRO2_DBETA SIN_BETA) / (COS_BETA M0);             | RK_DERIV
1269 M| 7         DRO2_DRO_0 = RO_2 / RO_0;                                       | RK_DERIV
1270 M| 6       END;                                                              | RK_DERIV
     E|                  .                                                          |
1271 M| 6     IF NORMAL_SHOCK_FLAG = ON THEN                                      | RK_DERIV
1272 M| 6       DO;                                                               | RK_DERIV
1273 M| 7         DM2_DM0 = -(GAM2 GAM2 M0 SQRT(((GAMMA0 M0_2) - GAM1) /          | RK_DERIV
     E|                                                                   2         |
1273 M| 7         (1. + (GAM1 M0_2)))) / (4. (((GAMMA0 M0_2) - GAM1) ));          | RK_DERIV
1274 M| 7         DT2_DM0 = (4. T0 GAM3 ((GAMMA0 M0_2 M0_2) + 1.)) / (            | RK_DERIV
1274 M| 7         GAM2 GAM2 M0_2 M0);                                             | RK_DERIV
1275 M| 7         DT2_DT0 = T2 / T0;                                              | RK_DERIV
     E|                                                                   2         |
1276 M| 7         DRO2_DM0 = (4. RO_0 GAM2 M0) / (((GAM3 M0_2) + 2.) );           | RK_DERIV
1277 M| 7         DRO2_DRO_0 = RO_2 / RO_0;                                       | RK_DERIV
1278 M| 6       END;                                                              | RK_DERIV
     E|                  .                                                          |
1279 M| 6     IF ((EXPANSION_FLAG = ON) AND ((M2_2 - 1.) > 0.)) THEN              | RK_DERIV
1280 M| 6       DO;                                                               | RK_DERIV
1281 M| 7         M2_2 = M2 M2;                                                   | RK_DERIV
1282 M| 7         DM2_DM0 = SQRT((M0_2 - 1.) / (M2_2 - 1.)) (M2 / M0) ((          | RK_DERIV
1282 M| 7         2. + (GAM3 M2_2)) / (2. + (GAM3 M0_2)));                        | RK_DERIV
1283 M| 7         DT2_DM0 = (T0 GAM3 M0) / (1. + (GAM1 M2_2));                    | RK_DERIV
1284 M| 7         DT2_DT0 = T2 / T0;                                              | RK_DERIV
```

```
STMT                  SOURCE                                                         CURRENT SCOPE
1285 M| 7                 DRO2_DM0 = (RO_2 M0) / (1. + (GAM1 M0_2));                  | RK_DERIV
1286 M| 7                 DRO2_DRO_0 = RO_2 / RO_0;                                   | RK_DERIV
1287 M| 7                 DT2_DM2 = -(T2 GAM3 M2) / (1. + (GAM1 M2_2));               | RK_DERIV
1288 M| 7                 DRO2_DM2 = -(RO_2 M2) / (1. + (GAM1 M2_2));                 | RK_DERIV
1289 M| 6              END;                                                           | RK_DERIV
     E|                                                                               |
1290 M| 6           IF SUBSONIC_FLAG = ON THEN                                        | RK_DERIV
1291 M| 6              DO;                                                            | RK_DERIV
1292 M| 7                 DM2_DM0 = 1.;                                               | RK_DERIV
1293 M| 7                 DT2_DM0 = 0.;                                               | RK_DERIV
1294 M| 7                 DT2_DT0 = 1.;                                               | RK_DERIV
1295 M| 7                 DRO2_DM0 = 0.;                                              | RK_DERIV
1296 M| 7                 DRO2_DRO_0 = 1.;                                            | RK_DERIV
1297 M| 6              END;                                                           | RK_DERIV
1298 M| 6           DBETA_DR = DBETA_DM0 DM0_DR;                                      | RK_DERIV
1299 M| 6           DBETA_DUR = DBETA_DM0 DM0_DUR;                                    | RK_DERIV
1300 M| 6           DBETA_DTDOT = DBETA_DM0 DM0_DTDOT;                                | RK_DERIV
1301 M| 6           DM2_DR = (DM2_DM0 DM0_DR) + (DM2_DBETA DBETA_DR);                 | RK_DERIV
1302 M| 6           DM2_DUR = (DM2_DM0 DM0_DUR) + (DM2_DBETA DBETA_DUR);              | RK_DERIV
1303 M| 6           DM2_DTDOT = (DM2_DM0 DM0_DTDOT) + (DM2_DBETA DBETA_DTDOT);        | RK_DERIV
1304 M| 6           DT2_DR = (DT2_DM0 DM0_DR) + (DT2_DBETA DBETA_DR) + (DT2_DT0       | RK_DERIV
1304 M| 6           DT0_DR) + (DT2_DM2 DM2_DR);                                       | RK_DERIV
1305 M| 6           DT2_DUR = (DT2_DM0 DM0_DUR) + (DT2_DBETA DBETA_DUR) + (           | RK_DERIV
1305 M| 6           DT2_DM2 DM2_DUR);                                                 | RK_DERIV
1306 M| 6           DT2_DTDOT = (DT2_DM0 DM0_DTDOT) + (DT2_DBETA DBETA_DTDOT) +       | RK_DERIV
1306 M| 6           (DT2_DM2 DM2_DTDOT);                                              | RK_DERIV
1307 M| 6           DU2_DR = (DU2_DM2 DM2_DR) + (DU2_DT2 DT2_DR);                     | RK_DERIV
1308 M| 6           DU2_DUR = (DU2_DM2 DM2_DUR) + (DU2_DT2 DT2_DUR);                  | RK_DERIV
```

```
STMT                    SOURCE                                                        CURRENT SCOPE
1309 M| 6          DU2_DTDOT = (DU2_DM2 DM2_DTDOT) + (DU2_DT2 DT2_DTDOT);            | RK_DERIV
1310 M| 6          DRO2_DR = (DRO2_DM0 DM0_DR) + (DRO2_DBETA DBETA_DR) + (           | RK_DERIV
1310 M| 6          DRO2_DRO_0 DRO_DR) + (DRO2_DM2 DM2_DR);                           | RK_DERIV
1311 M| 6          DRO2_DUR = (DRO2_DM0 DM0_DUR) + (DRO2_DBETA DBETA_DUR) + (        | RK_DERIV
1311 M| 6          DRO2_DM2 DM2_DUR);                                                | RK_DERIV
1312 M| 6          DRO2_DTDOT = (DRO2_DM0 DM0_DTDOT) + (DRO2_DBETA DBETA_DTDOT)      | RK_DERIV
1312 M| 6          + (DRO2_DM2 DM2_DTDOT);                                           | RK_DERIV
1313 M| 6          DISP1_DM2 = -300. - (200. M2);                                    | RK_DERIV
1314 M| 6          IF M2 > 0. THEN                                                   | RK_DERIV
     E|                                     .52                 .6                   |
1315 M| 6             DISP2_DM2 = EXP(-1.73 (M2    )) ((24000. (M2   )) - (13494.   | RK_DERIV
     E|                  1.12                                                        |
1315 M| 6             (M2     )));                                                   | RK_DERIV
1316 M| 5       END;                                                                 | RK_DERIV
1317 M| 5    IF ((RO_2 ¬= 0.) AND (U2 ¬= 0.)) THEN                                   | RK_DERIV
1318 M| 5       DO;                                                                  | RK_DERIV
     E|                    .                                                         |
1319 M| 6          IF ((TURBOJET_POWER = ON) AND (TURBOJET_ISP ¬= 0.)) THEN          | RK_DERIV
1320 M| 6             DO;                                                            | RK_DERIV
1321 M| 7                DT1_DR = TURBOJET_THRUST ((DRO2_DR / RO_2) + (DU2_DR /      | RK_DERIV
1321 M| 7                U2) + ((DISP1_DM2 DM2_DR) / TURBOJET_ISP));                 | RK_DERIV
1322 M| 7                DT1_DUR = TURBOJET_THRUST ((DRO2_DUR / RO_2) + (            | RK_DERIV
1322 M| 7                DU2_DUR / U2) + ((DISP1_DM2 DM2_DUR) / TURBOJET_ISP));      | RK_DERIV
1323 M| 7                DT1_DTDOT = TURBOJET_THRUST ((DRO2_DTDOT / RO_2) + (        | RK_DERIV
1323 M| 7                DU2_DTDOT / U2) + ((DISP1_DM2 DM2_DTDOT) /                  | RK_DERIV
1323 M| 7                TURBOJET_ISP));                                             | RK_DERIV
1324 M| 6             END;                                                           | RK_DERIV
1325 M| 6          ELSE                                                              | RK_DERIV
1325 M| 6             DO;                                                            | RK_DERIV
```

```
STMT                         SOURCE                                                CURRENT SCOPE
1326 M| 7          DT1_DR = 0.;                                                    | RK_DERIV
1327 M| 7          DT1_DUR = 0.;                                                   | RK_DERIV
1328 M| 7          DT1_DTDOT = 0.;                                                 | RK_DERIV
1329 M| 6       END;                                                               | RK_DERIV
     E|                                                                            |
1330 M| 6    IF ((SCRAMJET_POWER = ON) AND (SCRAMJET_ISP ¬= 0.) AND (              | RK_DERIV
1330 M| 6    SCRJ_MASS_CAPTURE_RATIO > 0.)) THEN                                   | RK_DERIV
1331 M| 6       DO;                                                                | RK_DERIV
1332 M| 7          S2 = (SCRAMJET_THRUST DMCR_DM2) /                               | RK_DERIV
1332 M| 7          SCRJ_MASS_CAPTURE_RATIO;                                        | RK_DERIV
1333 M| 7          DTH2_DR = (SCRAMJET_THRUST ((DRO2_DR / RO_2) + (DU2_DR          | RK_DERIV
1333 M| 7          / U2) + ((DISP2_DM2 DM2_DR) / SCRAMJET_ISP))) + (S2             | RK_DERIV
1333 M| 7          DM2_DR);                                                        | RK_DERIV
1334 M| 7          DTH2_DUR = (SCRAMJET_THRUST ((DRO2_DUR / RO_2) + (              | RK_DERIV
1334 M| 7          DU2_DUR / U2) + ((DISP2_DM2 DM2_DUR) / SCRAMJET_ISP)))          | RK_DERIV
1334 M| 7          + (S2 DM2_DUR);                                                 | RK_DERIV
1335 M| 7          DTH2_DTDOT = (SCRAMJET_THRUST ((DRO2_DTDOT / RO_2) + (          | RK_DERIV
1335 M| 7          DU2_DTDOT / U2) + ((DISP2_DM2 DM2_DTDOT) /                      | RK_DERIV
1335 M| 7          SCRAMJET_ISP))) + (S2 DM2_DTDOT);                               | RK_DERIV
1336 M| 6       END;                                                               | RK_DERIV
1337 M| 6    ELSE                                                                  | RK_DERIV
1337 M| 6       DO;                                                                | RK_DERIV
1338 M| 7          DTH2_DR = 0.;                                                   | RK_DERIV
1339 M| 7          DTH2_DUR = 0.;                                                  | RK_DERIV
1340 M| 7          DTH2_DTDOT = 0.;                                                | RK_DERIV
1341 M| 6       END;                                                               | RK_DERIV
1342 M| 5    END;                                                                  | RK_DERIV
1343 M| 5 DCL1_DR = DCL1_DM0 DM0_DR;                                               | RK_DERIV
```

```
STMT                                   SOURCE                                          CURRENT SCOPE

1344 M| 5     DCL2_DR = DCL2_DM0 DM0_DR;                                              | RK_DERIV
1345 M| 5     DCD1_DR = DCD1_DM0 DM0_DR;                                              | RK_DERIV
1346 M| 5     DCD2_DR = DCD2_DM0 DM0_DR;                                              | RK_DERIV
1347 M| 5     DCL1_DUR = DCL1_DM0 DM0_DUR;                                            | RK_DERIV
1348 M| 5     DCL2_DUR = DCL2_DM0 DM0_DUR;                                            | RK_DERIV
1349 M| 5     DCD1_DUR = DCD1_DM0 DM0_DUR;                                            | RK_DERIV
1350 M| 5     DCD2_DUR = DCD2_DM0 DM0_DUR;                                            | RK_DERIV
1351 M| 5     DCL1_DTDOT = DCL1_DM0 DM0_DTDOT;                                        | RK_DERIV
1352 M| 5     DCL2_DTDOT = DCL2_DM0 DM0_DTDOT;                                        | RK_DERIV
1353 M| 5     DCD1_DTDOT = DCD1_DM0 DM0_DTDOT;                                        | RK_DERIV
1354 M| 5     DCD2_DTDOT = DCD2_DM0 DM0_DTDOT;                                        | RK_DERIV
     E|                                                              2                |
1355 M| 5     V0_2 = (X_STORE         X_STORE         ) + ((SR T_VAL) );              | RK_DERIV
     S|                  I_TIME:2        I_TIME:2                                     |
1356 M| 5     DL_DR = (LIFT ((DRO_DR / RO_C) + ((2. SR T_VAL T_VAL) / V0_2))) +       | RK_DERIV
1356 M| 5     (((DCL1_DR WING_AREA) + (DCL2_DR SS_PLANFORM_AREA)) DYNAMIC_P0);        | RK_DERIV
1357 M| 5     DL_DUR = ((2. X_STORE         LIFT) / V0_2) + (((DCL1_DUR              | RK_DERIV
     S|                        I_TIME:2                                               |
1357 M| 5     WING_AREA) + (DCL2_DUR SS_PLANFORM_AREA)) DYNAMIC_P0);                  | RK_DERIV
1358 M| 5     DL_DTDOT = ((2. SR SR T_VAL LIFT) / V0_2) + (((DCL1_DTDOT               | RK_DERIV
1358 M| 5     WING_AREA) + (DCL2_DTDOT SS_PLANFORM_AREA)) DYNAMIC_P0);                | RK_DERIV
1359 M| 5     DD_DR = (DRAG ((DRO_DR / RO_0) + ((2. SR T_VAL T_VAL) / V0_2))) +       | RK_DERIV
1359 M| 5     (((DCD1_DR WING_AREA) + (DCD2_DR SS_PLANFORM_AREA)) DYNAMIC_P0);        | RK_DERIV
1360 M| 5     DD_DUR = ((2. X_STORE         DRAG) / V0_2) + (((DCD1_DUR              | RK_DERIV
     S|                        I_TIME:2                                               |
1360 M| 5     WING_AREA) + (DCD2_DUR SS_PLANFORM_AREA)) DYNAMIC_P0);                  | RK_DERIV
1361 M| 5     DD_DTDOT = ((2. SR SR T_VAL DRAG) / V0_2) + (((DCD1_DTDOT               | RK_DERIV
1361 M| 5     WING_AREA) + (DCD2_DTDOT SS_PLANFORM_AREA)) DYNAMIC_P0);                | RK_DERIV
1362 M| 5     SIN_A = SIN(U_ACTIVE         );                                         | RK_DERIV
     S|                       L_TIME:1                                                |
```

```
STMT                                SOURCE                                      CURRENT SCOPE

1363 M| 5              COS_A = COS(U_ACTIVE          );                          | RK_DERIV
     S|                                   L_TIME:1                               |

1364 M| 5              DP_DR = (DL_DR SIN_A) - (DD_DR COS_A);                   | RK_DERIV

1365 M| 5              DP_DUR = (DL_DUR SIN_A) - (DD_DUR COS_A);                | RK_DERIV

1366 M| 5              DP_DTDOT = (DL_DTDOT SIN_A) - (DD_DTDOT COS_A);          | RK_DERIV

1367 M| 5              DN_DR = (DL_DR COS_A) + (DD_DR SIN_A);                   | RK_DERIV

1368 M| 5              DN_DUR = (DL_DUR COS_A) + (DD_DUR SIN_A);                | RK_DERIV

1369 M| 5              DN_DTDOT = (DL_DTDOT COS_A) + (DD_DTDOT SIN_A);          | RK_DERIV

     E|                       .                                                  |
1370 M| 5              IF STAGE_SEP = OFF THEN                                  | RK_DERIV

1371 M| 5                 DO;                                                   | RK_DERIV

1372 M| 6                    DMASSFLUX_DR1 = (DRO2_DR U2) + (RO_2 DU2_DR);      | RK_DERIV

1373 M| 6                    DMASSFLUX_DUR1 = (DRO2_DUR U2) + (RO_2 DU2_DUR);   | RK_DERIV

1374 M| 6                    DMASSFLUX_DTDOT1 = (DRC2_DTDOT U2) + (RO_2 DU2_DTDOT);  | RK_DERIV

1375 M| 6                    DMASSFLUX_DR2 = (DRO2_DR U2 SCRJ_MASS_CAPTURE_RATIO) + (RO_2 | RK_DERIV

1375 M| 6                    DU2_DR SCRJ_MASS_CAPTURE_RATIO) + (RO_2 U2 DMCR_DM2 DM2_DR); | RK_DERIV

1376 M| 6                    DMASSFLUX_DUR2 = (DRO2_DUR U2 SCRJ_MASS_CAPTURE_RATIO) + (   | RK_DERIV

1376 M| 6                    RO_2 DU2_DUR SCRJ_MASS_CAPTURE_RATIO) + (RO_2 U2 DMCR_DM2    | RK_DERIV

1376 M| 6                    DM2_DUR);                                          | RK_DERIV

1377 M| 6                    DMASSFLUX_DTDOT2 = (DRO2_DTDOT U2 SCRJ_MASS_CAPTURE_RATIO) + | RK_DERIV

1377 M| 6                    (RO_2 DU2_DTDOT SCRJ_MASS_CAPTURE_RATIO) + (RO_2 U2 DMCR_DM2 | RK_DERIV

1377 M| 6                    DM2_DTDOT);                                        | RK_DERIV

1378 M| 5                 END;                                                  | RK_DERIV

     E|                                                     2                    |
1379 M| 5              OSC1 = L_CONST / (G_LOAD ((MD_MASS G0) ));               | RK_DERIV

1380 M| 5              DT_DR = DT1_DR + DTH2_DR;                                | RK_DERIV

1381 M| 5              DT_DUR = DT1_DUR + DTH2_DUR;                             | RK_DERIV

1382 M| 5              DT_DTDOT = DT1_DTDOT + DTH2_DTDOT;                       | RK_DERIV

1383 M| 5              IF G_LOAD > G_D THEN                                     | RK_DERIV
```

```
STMT                                  SOURCE                                          CURRENT SCOPE

1384 M| 5                   DO;                                                       | RK_DERIV

1385 M| 6                      L_SUB_X  = OSC1 ((NET_X_FORCE (DT_DR + DP_DR)) + (N_AERO  | RK_DERIV
     S|                               1                                                |

1385 M| 6                      DN_DR));                                                | RK_DERIV

1386 M| 6                      L_SUB_X  = OSC1 ((NET_X_FORCE (DT_DUR + DP_DUR)) + (N_AERO | RK_DERIV
     S|                               2                                                |

1386 M| 6                      DN_DUR));                                               | RK_DERIV

1387 M| 6                      L_SUB_X  = OSC1 ((NET_X_FORCE (DT_DTDOT + DP_DTDOT)) + (   | RK_DERIV
     S|                               4                                                |

1387 M| 6                      N_AERO DN_DTDOT));                                      | RK_DERIV

1388 M| 5                   END;                                                      | RK_DERIV

     E|                  -                                                            |
1389 M| 5                DG_DPI = 0.;                                                 | RK_DERIV

     E|                   -                                                           |
1390 M| 5                DAR_DPI = 0.;                                                | RK_DERIV

     E|                   -                                                           |
1391 M| 5                DAW1_DPI = 0.;                                               | RK_DERIV

     E|                   -                                                           |
1392 M| 5                DAW2_DPI = 0.;                                               | RK_DERIV

     E|                   -                                                           |
1393 M| 5                DTH1_DPI = 0.;                                               | RK_DERIV

     E|                   -                                                           |
1394 M| 5                DTH2_DPI = 0.;                                               | RK_DERIV

     E|                   -                                                           |
1395 M| 5                DTH3_DPI = 0.;                                               | RK_DERIV

     E|                   -                                                           |
1396 M| 5                DAE1_DPI = 0.;                                               | RK_DERIV

     E|                   -                                                           |
1397 M| 5                DAE2_DPI = 0.;                                               | RK_DERIV

1398 M| 5                [DMASSFLUX_DP1] = 0.;                                        | RK_DERIV

1399 M| 5                [DMASSFLUX_DP2] = 0.;                                        | RK_DERIV

1400 M| 5                DO FOR I_FK = 1 TO NUM_CONSTANT_PARAMETERS;                  | RK_DERIV

1401 M| 6                   IF I_FK = 9 THEN                                          | RK_DERIV
```

```
STMT                                  SOURCE                                      CURRENT SCOPE

1402 M| 6          DAW2_DPI  = 2.226778E-05 DELIVERED_PLANFORM_AREA ((1. + (     | RK_DERIV
     S|                    9                                                     |
     E|                          -.3333                                          |
1402 M| 6          3.34E-05 P ))       );                                        | RK_DERIV
     S|                      9                                                   |
     E|                                                                          |
1403 M| 6       IF STAGE_SEP = OFF THEN                                          | RK_DERIV
1404 M| 6          DO;                                                           | RK_DERIV
1405 M| 7             IF I_FK = 1 THEN                                           | RK_DERIV
1406 M| 7                DO;                                                     | RK_DERIV
1407 M| 8                   DRO2_DP1 = 0.;                                       | RK_DERIV
1408 M| 8                   DT2_DP1 = 0.;                                        | RK_DERIV
1409 M| 8                   DM2_DP1 = 0.;                                        | RK_DERIV
     E|                                                                          |
1410 M| 8                   IF OBLIQUE_SHOCK_FLAG = ON THEN                      | RK_DERIV
1411 M| 8                      DO;                                               | RK_DERIV
1412 M| 9                         DBETA_DP1 = ((((M0_2 (GAMMA0 + 1. - (2.        | RK_DERIV
     E|                                                   2                      |
1412 M| 9                         SIN_2_BETA))) + 2.)  ) SIN_2_BETA) / (2. (COS( | RK_DERIV
     E|                                                      2                   |
1412 M| 9                         P  + U_ACTIVE          ) ) (2. + (M0_2 (GAM2 - | RK_DERIV
     S|                          1            L_TIME:1                           |
1412 M| 9                         (4. SIN_2_BETA))) - (M0_2 M0_2 GAM3            | RK_DERIV
1412 M| 9                         SIN_2_BETA SIN_2_BETA)));                      | RK_DERIV
1413 M| 9                         DRO2_DP1 = (4. RO_0 GAM2 M0_2 SIN_BETA         | RK_DERIV
1413 M| 9                         COS_BETA DBETA_DP1) / (((GAM3 M0_2 SIN_2_BETA  | RK_DERIV
     E|                                  2                                       |
1413 M| 9                         ) + 2.)  );                                    | RK_DERIV
1414 M| 9                         IF ((SIN_B_T ¬= 0.) AND (M2 ¬= 0.)) THEN       | RK_DERIV
1415 M| 9                            DM2_DP1 = (((1. - DBETA_DP1) M2 COS_B_T) /  | RK_DERIV
1415 M| 9                            SIN_B_T) - ((M0_2 SIN_BETA COS_BETA GAM2    | RK_DERIV
1415 M| 9                            GAM2 DBETA_DP1) / (4. M2 ((SIN_B_T ((       | RK_DERIV
```

```
STMT                                   SOURCE                                         CURRENT SCOPE
     E|                                                       2                      |
1415 M| 9                         GAMMA0 M0_2 SIN_2_BETA) - GAM1)) )));              | RK_DERIV
1416 M| 9                       IF ((SIN_2_BETA ¬= 0.) AND (SIN_BETA ¬= 0.))         | RK_DERIV
1416 M| 9                       THEN                                                 | RK_DERIV
1417 M| 9                         DT2_DP1 = (4. TO GAM3 DBETA_DP1 COS_BETA (         | RK_DERIV
1417 M| 9                         (GAMMA0 M0_2 M0_2 SIN_2_BETA SIN_2_BETA) +         | RK_DERIV
1417 M| 9                         1.)) / (GAM2 GAM2 M0_2 SIN_2_BETA SIN_BETA         | RK_DERIV
1417 M| 9                         );                                                 | RK_DERIV
1418 M| 8                     END;                                                   | RK_DERIV
     E|                        .                                                     |
1419 M| 8                   IF EXPANSION_FLAG = ON THEN                              | RK_DERIV
1420 M| 8                     DO;                                                    | RK_DERIV
1421 M| 9                       IF (M2_2 - 1.) > 0. THEN                             | RK_DERIV
1422 M| 9                         DM2_DP1 = -(M2 (GAM2 + (GAM3 (M2_2 - 1.)))         | RK_DERIV
1422 M| 9                         ) / (2. SQRT(M2_2 - 1.));                          | RK_DERIV
1423 M| 9                       DT2_DP1 = -(T2 GAM3 M2 DM2_DP1) / (1. + (GAM1        | RK_DERIV
1423 M| 9                       M2_2));                                              | RK_DERIV
1424 M| 9                       DRO2_DP1 = -(RO_2 M2 DM2_DP1) / (1. + (GAM1          | RK_DERIV
1424 M| 9                       M2_2));                                              | RK_DERIV
1425 M| 8                     END;                                                   | RK_DERIV
1426 M| 8                   DISP1_DP1 = DISP1_DM2 DM2_DP1;                           | RK_DERIV
1427 M| 8                   DISP2_DP1 = DISP2_DM2 DM2_DP1;                           | RK_DERIV
1428 M| 8                   DMCR_DP1 = DMCR_DM2 DM2_DP1;                             | RK_DERIV
1429 M| 8                   DU2_DP1 = (DU2_DM2 DM2_DP1) + (DU2_DT2 DT2_DP1);         | RK_DERIV
     E|                              .                                               |
1430 M| 8                   IF ((TURBOJET_POWER = ON) AND (RO_2 ¬= 0.) AND (U2       | RK_DERIV
1430 M| 8                   ¬= 0.) AND (TURBOJET_ISP ¬= 0.)) THEN                    | RK_DERIV
1431 M| 8                     DTH1_DPI  = TURBOJET_THRUST ((DRO2_DP1 / RO_2) +       | RK_DERIV
     S|                              1                                               |
1431 M| 8                     (DU2_DP1 / U2) + (DISP1_DP1 / TURBOJET_ISP));          | RK_DERIV
```

```
STMT                                SOURCE                                               CURRENT SCOPE
1432 M|  8         IF ((SCRJ_MASS_CAPTURE_RATIO > 0.) AND (             | RK_DERIV
      E|                                                                 |
1432 M|  8         SCRAMJET_POWER = ON) AND (RO_2 ~= 0.) AND (U2 ~= 0.  | RK_DERIV
1432 M|  8         ) AND (SCRAMJET_ISP ~= 0.) AND (                     | RK_DERIV
1432 M|  8         SCRJ_MASS_CAPTURE_RATIO ~= 0.)) THEN                  | RK_DERIV
1433 M|  8            DTH2_DPI  = SCRAMJET_THRUST ((DRO2_DP1 / RO_2) +  | RK_DERIV
      S|                    1                                            |
1433 M|  8            (DU2_DP1 / U2) + (DISP2_DP1 / SCRAMJET_ISP) + (   | RK_DERIV
1433 M|  8            DMCR_DP1 / SCRJ_MASS_CAPTURE_RATIO));             | RK_DERIV
1434 M|  8         DMASSFLUX_DP1  = P  P ((RO_2 DU2_DP1) + (DRO2_DP1    | RK_DERIV
      S|                          1   2  4                               |
1434 M|  8         U2));                                                 | RK_DERIV
1435 M|  8         DMASSFLUX_DP2  = P  P ((RO_2 DU2_DP1                 | RK_DERIV
      S|                          1   2  3                               |
1435 M|  8         SCRJ_MASS_CAPTURE_RATIO) + (DRO2_DP1 U2               | RK_DERIV
1435 M|  8         SCRJ_MASS_CAPTURE_RATIO) + (RO_2 U2 DMCR_DP1));       | RK_DERIV
1436 M|  7      END;                                                     | RK_DERIV
1437 M|  7   IF I_FK = 2 THEN                                            | RK_DERIV
1438 M|  7      DO;                                                      | RK_DERIV
1439 M|  8         DAE1_DPI  = P ;                                       | RK_DERIV
      S|                 2    4                                          |
1440 M|  8         DAE2_DPI  = P ;                                       | RK_DERIV
      S|                 2    3                                          |
1441 M|  7      END;                                                     | RK_DERIV
1442 M|  7   IF I_FK = 3 THEN                                            | RK_DERIV
1443 M|  7      DAE2_DPI  = P ;                                          | RK_DERIV
      S|              3    2                                             |
1444 M|  7   IF I_FK = 4 THEN                                            | RK_DERIV
1445 M|  7      DAE1_DPI  = P ;                                          | RK_DERIV
      S|              4    2                                             |
1446 M|  7   IF I_FK = 5 THEN                                            | RK_DERIV
```

```
STMT                                        SOURCE                                    CURRENT SCOPE

1447 M| 7                  DAW1_DPI  = (2. WING_AREA) / P ;                          | RK_DERIV
     S|                            5                      5                          |

1448 M| 7               IF I_FK = 6 THEN                                             | RK_DERIV

1449 M| 7                  DO;                                                       | RK_DERIV

     E|                                                    2                         |
1450 M| 8                     DAR_DPI  = 4. / (COS(P ) );                            | RK_DERIV
     S|                             6              6                                 |

     E|                                                                2             |
1451 M| 8                     DAW1_DPI  = -(P  P ) / (4. (SIN(P ) ));                | RK_DERIV
     S|                              6      5  5               6                     |

1452 M| 7                  END;                                                      | RK_DERIV

     E|                       .                                                      |
1453 M| 7               IF TURBOJET_POWER = ON THEN                                  | RK_DERIV

1454 M| 7                  DTH1_DPI     = DTH1_DPI     + ((TURBOJET_THRUST           | RK_DERIV
     S|                          I_FK           I_FK                                 |

1454 M| 7                  DAE1_DPI     ) / (P  P ));                                | RK_DERIV
     S|                          I_FK        2  4                                    |

     E|                       .                                                      |
1455 M| 7               IF SCRAMJET_POWER = ON THEN                                  | RK_DERIV

1456 M| 7                  DTH2_DPI     = DTH2_DPI     + ((SCRAMJET_THRUST           | RK_DERIV
     S|                          I_FK           I_FK                                 |

1456 M| 7                  DAE2_DPI     ) / (P  P ));                                | RK_DERIV
     S|                          I_FK        2  3                                    |

1457 M| 7               DMASSFLUX_DP1     = DMASSFLUX_DP1     + (RO_2 U2 DAE1_DPI    | RK_DERIV
     S|                             I_FK                I_FK                         |

1457 M| 7                    );                                                      | RK_DERIV
     S|                 I_FK                                                         |

1458 M| 7               DMASSFLUX_DP2     = DMASSFLUX_DP2     + (RO_2 U2             | RK_DERIV
     S|                             I_FK                I_FK                         |

1458 M| 7               SCRJ_MASS_CAPTURE_RATIO DAE2_DPI     );                      | RK_DERIV
     S|                                              I_FK                            |

1459 M| 6            END;                                                            | RK_DERIV

1460 M| 6         ELSE IF I_FK = 10 THEN                                             | RK_DERIV

1461 M| 6            DTH3_DPI   = CAP_PHI;                                           | RK_DERIV
     S|                      10                                                      |
```

```
STMT                                   SOURCE                                                CURRENT SCOPE

1452 M| 6               DT_DP      = DTH1_DPI      + DTH2_DPI      + DTH3_DPI      ;         | RK_DERIV
     S|                      I_FK           I_FK            I_FK            I_FK            |

1463 M| 6               DL_DPI = DYNAMIC_P0 ((DCL1_DAR DAR_DPI      WING_AREA) + (CL      | RK_DERIV
     S|                                                  I_FK                            |

1463 M| 6               DAW1_DPI     ) + (SS_CL DAW2_DPI     ));                          | RK_DERIV
     S|                         I_FK                   I_FK                              |

1464 M| 6               DD_DPI = DYNAMIC_P0 ((DCD1_DAR DAR_DPI      WING_AREA) + (CD      | RK_DERIV
     S|                                                  I_FK                            |

1464 M| 6               DAW1_DPI     ) + (SS_CD DAW2_DPI     ));                          | RK_DERIV
     S|                         I_FK                   I_FK                              |

1465 M| 6               DP_DP      = (DL_DPI SIN_A) - (DD_DPI COS_A);                     | RK_DERIV
     S|                      I_FK                                                        |

1466 M| 6               DN_DP      = (DL_DPI COS_A) + (DD_DPI SIN_A);                     | RK_DERIV
     S|                      I_FK                                                        |

     E|                           .                                                      |
1467 M| 6               IF ((STAGE_SEP = OFF) OR (I_FK > FS_FIXED_PARAMETERS)) THEN       | RK_DERIV

1468 M| 6                  DG_DPI      = ((((NET_THRUST + P_AERO) (DT_DP      + DP_DP     | RK_DERIV
     S|                          I_FK                          I_FK         I_FK         |

     E|                                                            2                     |
1468 M| 6                  )) + (N_AERO DN_DP      )) / (G_LOAD ((MD_MASS G0) ))) - ((    | RK_DERIV
     S|                                       I_FK                                       |

1468 M| 6                  G_LOAD DM_DP      ) / MD_MASS);                                | RK_DERIV
     S|                               I_FK                                               |

1469 M| 6               IF G_LOAD > G_D THEN                                              | RK_DERIV

1470 M| 6                  L_SUB_P      = DG_DPI      L_CONST;                            | RK_DERIV
     S|                          I_FK          I_FK                                      |

1471 M| 5            END;                                                                 | RK_DERIV

     E|                -                                                                 |
1472 M| 5            DM1DOT_DUA = 0.;                                                     | RK_DERIV

     E|                -                                                                 |
1473 M| 5            DM2DOT_DUA = 0.;                                                     | RK_DERIV

     E|                -                                                                 |
1474 M| 5            DM3DOT_DUA = 0.;                                                     | RK_DERIV

1475 M| 5            DT_DUA  = DT_DP ;                                                    | RK_DERIV
     S|                    1         1                                                   |

1476 M| 5            DPHI_DUA2 = -2. U_ACTIVE          CAP_PHI CAP_PHI;                   | RK_DERIV
     S|                                  L_TIME:2                                        |
```

```
STMT                          SOURCE                                                        CURRENT SCOPE

1477 M| 5    DT_DUA  = (NET_THRUST DPHI_DUA2) / CAP_PHI;                                | RK_DERIV
     S|            2                                                                      |

1478 M| 5    DL_DUA1 = DYNAMIC_P0 ((DCL_DALPHA DEGREES_PER_RADIAN WING_AREA) +          | RK_DERIV

1478 M| 5    (DCL2_DUA1 SS_PLANFORM_AREA));                                             | RK_DERIV

1479 M| 5    DD_DUA1 = DYNAMIC_P0 ((2. CL DCD_DCL2 DCL_DALPHA                           | RK_DERIV

1479 M| 5    DEGREES_PER_RADIAN WING_AREA) + (DCD2_DUA1 SS_PLANFORM_AREA));             | RK_DERIV

1480 M| 5    DP_DUA  = (DL_DUA1 SIN_A) - (DD_DUA1 COS_A) + N_AERO;                      | RK_DERIV
     S|            1                                                                      |

1481 M| 5    DN_DUA  = (DD_DUA1 SIN_A) + (DL_DUA1 COS_A) - P_AERO;                      | RK_DERIV
     S|            1                                                                      |

     E|              .                                                                    |
1482 M| 5    IF STAGE_SEP = OFF THEN                                                    | RK_DERIV

1483 M| 5       DO;                                                                     | RK_DERIV

     E|                        .                                                          |
1484 M| 6          IF ((TURBOJET_POWER = ON) AND (TURBOJET_ISP ¬= 0.)) THEN             | RK_DERIV

1485 M| 6             DO;                                                               | RK_DERIV

1486 M| 7                DM1DOT_DUA  = ((TURBOJET_THRUST DISP1_DM2 DM2_DP1) / (        | RK_DERIV
     S|                           1                                                       |

1486 M| 7                TURBOJET_ISP TURBOJET_ISP G0)) - (DTH1_DPI  / (               | RK_DERIV
     S|                                                           1                       |

1486 M| 7                TURBOJET_ISP G0));                                             | RK_DERIV

1487 M| 7                DM1DOT_DUA  = -(TURBOJET_THRUST DPHI_DUA2) / (                 | RK_DERIV
     S|                           2                                                       |

1487 M| 7                TURBOJET_ISP G0 CAP_PHI);                                      | RK_DERIV

1488 M| 6             END;                                                              | RK_DERIV

     E|                        .                                                          |
1489 M| 6          IF ((SCRAMJET_POWER = ON) AND (SCRAMJET_ISP ¬= 0.)) THEN             | RK_DERIV

1490 M| 6             DO;                                                               | RK_DERIV

1491 M| 7                DM2DOT_DUA  = ((SCRAMJET_THRUST DISP2_DM2 DM2_DP1) / (        | RK_DERIV
     S|                           1                                                       |

1491 M| 7                SCRAMJET_ISP SCRAMJET_ISP G0)) - (DTH2_DPI  / (               | RK_DERIV
     S|                                                           1                       |

1491 M| 7                SCRAMJET_ISP G0));                                             | RK_DERIV
```

```
STMT                                   SOURCE                                          CURRENT SCOPE

1492 M|  7                    DM2DOT_DUA  = -(SCRAMJET_THRUST DPHI_DUA2) / (          | RK_DERIV
     S|                                2                                              |

1492 M|  7                    SCRAMJET_ISP G0 CAP_PHI);                               | RK_DERIV

1493 M|  6                 END;                                                       | RK_DERIV

1494 M|  5           END;                                                             | RK_DERIV

1495 M|  5        ELSE                                                                | RK_DERIV

1495 M|  5           DM3DOT_DUA  = -(ROCKET_THRUST DPHI_DUA2) / (ROCKET_ISP G0        | RK_DERIV
     S|                       2                                                       |

1495 M|  5           CAP_PHI);                                                        | RK_DERIV

     E|                        2                                                      |
1496 M|  5        S3 = ((SR T_VAL) ) + (X_STORE         X_STORE         );             | RK_DERIV
     S|                                   I_TIME:2        I_TIME:2                    |

1497 M|  5        DDELTA_DR = -(X_STORE         T_VAL) / S3;                          | RK_DERIV
     S|                            I_TIME:2                                           |

1498 M|  5        DDELTA_DUR = (SR T_VAL) / S3;                                       | RK_DERIV

1499 M|  5        DDELTA_DTDOT = -(SR X_STORE         ) / S3;                         | RK_DERIV
     S|                                  I_TIME:2                                     |

1500 M|  5        S2 = ARCTAN(X_STORE         / (SR T_VAL)) + U_ACTIVE        ;        | RK_DERIV
     S|                         I_TIME:2                         L_TIME:1             |

1501 M|  5        COS_DELTA = COS(S2);                                                | RK_DERIV

1502 M|  5        SIN_DELTA = SIN(S2);                                                | RK_DERIV

1503 M|  5        SG = (UNIVERSAL_G_CONSTANT EARTH_MASS) / (SR SR);                   | RK_DERIV

1504 M|  5        DFR_DMI = -SG;                                                      | RK_DERIV

1505 M|  5        S5 = DT_DR + DP_DR - (N_AERO DDELTA_DR);                            | RK_DERIV

1506 M|  5        S6 = DN_DR + ((NET_THRUST + P_AERO) DDELTA_DR);                     | RK_DERIV

1507 M|  5        DFR_DR = (S5 SIN_DELTA) + (S6 COS_DELTA) + ((2. SG NET_MASS       ) | RK_DERIV
     S|                                                                     I_TIME   |

1507 M|  5        / SR);                                                              | RK_DERIV

1508 M|  5        DFTHETA_DR = (S5 COS_DELTA) - (S6 SIN_DELTA);                       | RK_DERIV

1509 M|  5        S5 = DT_DUR + DP_DUR - (N_AERO DDELTA_DUR);                         | RK_DERIV

1510 M|  5        S6 = DN_DUR + ((NET_THRUST + P_AERO) DDELTA_DUR);                   | RK_DERIV

1511 M|  5        DFR_DUR = (S5 SIN_DELTA) + (S6 COS_DELTA);                          | RK_DERIV
```

```
STMT                       SOURCE                                          CURRENT SCOPE

1512 M| 5       DFTHETA_DUR = (S5 COS_DELTA) - (S6 SIN_DELTA);             | RK_DERIV
1513 M| 5       S5 = DT_DTDOT + DP_DTDOT - (N_AERO DDELTA_DTDOT);          | RK_DERIV
1514 M| 5       S6 = DN_DTDOT + ((NET_THRUST + P_AERO) DDELTA_DTDOT);      | RK_DERIV
1515 M| 5       DFR_DTDOT = (S5 SIN_DELTA) + (S6 COS_DELTA);               | RK_DERIV
1516 M| 5       DFTHETA_DTDOT = (S5 COS_DELTA) - (S6 SIN_DELTA);           | RK_DERIV
     E|                  .                                                 |
1517 M| 5       IF STAGE_SEP = OFF THEN                                    | RK_DERIV
1518 M| 5          DO;                                                     | RK_DERIV
     E|                      .                                             |
1519 M| 6             IF TURBOJET_POWER = ON THEN                          | RK_DERIV
1520 M| 6                DO;                                               | RK_DERIV
1521 M| 7                   S5 = -P  P  TJ_MAX_FUEL_AIR_RATIO CAP_PHI;     | RK_DERIV
     S|                          2  4                                      |
1522 M| 7                   DM1DOT_DR = S5 DMASSFLUX_DR1;                  | RK_DERIV
1523 M| 7                   DM1DOT_DUR = S5 DMASSFLUX_DUR1;                | RK_DERIV
1524 M| 7                   DM1DOT_DTDOT = S5 DMASSFLUX_DTDOT1;            | RK_DERIV
1525 M| 5                END;                                              | RK_DERIV
1526 M| 6             ELSE                                                 | RK_DERIV
1526 M| 6                DO;                                               | RK_DERIV
1527 M| 7                   DM1DOT_DR = 0.;                                | RK_DERIV
1528 M| 7                   DM1DOT_DUR = 0.;                               | RK_DERIV
1529 M| 7                   DM1DOT_DTDOT = 0.;                             | RK_DERIV
1530 M| 6                END;                                              | RK_DERIV
     E|                      .                                             |
1531 M| 6             IF SCRAMJET_POWER = ON THEN                          | RK_DERIV
1532 M| 6                DO;                                               | RK_DERIV
1533 M| 7                   S5 = -P  P  SCRJ_MAX_FUEL_AIR_RATIO CAP_PHI;   | RK_DERIV
     S|                          2  3                                      |
1534 M| 7                   DM2DOT_DR = S5 DMASSFLUX_DR2;                  | RK_DERIV
1535 M| 7                   DM2DOT_DUR = S5 DMASSFLUX_DUR2;                | RK_DERIV
```

```
STMT                                     SOURCE                                          CURRENT SCOPE
1536 M| 7                        DM2DOT_DTDOT = S5 DMASSFLUX_DTDOT2;                   | RK_DERIV
1537 M| 6                     END;                                                     | RK_DERIV
1538 M| 6                  ELSE                                                        | RK_DERIV
1538 M| 6                     DO;                                                      | RK_DERIV
1539 M| 7                        DM2DOT_DR = 0.;                                       | RK_DERIV
1540 M| 7                        DM2DOT_DUR = 0.;                                      | RK_DERIV
1541 M| 7                        DM2DOT_DTDOT = 0.;                                    | RK_DERIV
1542 M| 6                     END;                                                     | RK_DERIV
1543 M| 5               END;                                                           | RK_DERIV
1544 M| 5            ELSE                                                              | RK_DERIV
1544 M| 5               DO;                                                            | RK_DERIV
1545 M| 6                  DM1DOT_DR = 0.;                                             | RK_DERIV
1546 M| 6                  DM1DOT_DUR = 0.;                                            | RK_DERIV
1547 M| 6                  DM1DOT_DTDOT = 0.;                                          | RK_DERIV
1548 M| 6                  DM2DOT_DR = 0.;                                             | RK_DERIV
1549 M| 6                  DM2DOT_DUR = 0.;                                            | RK_DERIV
1550 M| 6                  DM2DOT_DTDOT = 0.;                                          | RK_DERIV
1551 M| 5               END;                                                           | RK_DERIV
     E|            *                                                                   |
1552 M| 5          F = 0;                                                              | RK_DERIV
1553 M| 5          F    = 1.;                                                          | RK_DERIV
     S|             1,2                                                                |
1554 M| 5          F    = 1.;                                                          | RK_DERIV
     S|             3,4                                                                |
1555 M| 5          F    = (X_STORE           X_STORE          ) + (DFR_DR / NET_MASS   | RK_DERIV
     S|             2,1           I_TIME:4          I_TIME:4                           |
1555 M| 5                 );                                                           | RK_DERIV
     S|            I_TIME                                                              |
1556 M| 5          F    = DFR_DUR / NET_MASS       ;                                   | RK_DERIV
     S|             2,2                      I_TIME                                    |
```

```
STMT                                        SOURCE                                           CURRENT SCOPE

1557 M| 5          F    = (2. SR X_STORE          ) + (DFR_DTDOT / NET_MASS        );       | RK_DERIV
     S|             2,4                 I_TIME:4                         I_TIME            |

1558 M| 5          F    = (DFR_DMI - (NET_R_FORCE / NET_MASS        )) / NET_MASS          | RK_DERIV
     S|             2.5                                       I_TIME              I_TIME   |

1558 M| 5          ;                                                                         | RK_DERIV

1559 M| 5          F    = F   ;                                                              | RK_DERIV
     S|             2,6    2,5                                                               |

1560 M| 5          F    = F   ;                                                              | RK_DERIV
     S|             2,7    2,5                                                               |

1561 M| 5          F    = ((((2. X_STORE          X_STORE          ) - (NET_THETA_FORCE     | RK_DERIV
     S|             4,1                 I_TIME:2         I_TIME:4                            |

1561 M| 5          / NET_MASS       )) / SR) + (DFTHETA_DR / NET_MASS       )) / SR;         | RK_DERIV
     S|                  I_TIME                                  I_TIME                      |

1562 M| 5          F    = ((DFTHETA_DUR / NET_MASS        ) - (2. X_STORE          )) /     | RK_DERIV
     S|             4,2                           I_TIME                   I_TIME:4          |

1562 M| 5          SR;                                                                       | RK_DERIV

1563 M| 5          F    = (DFTHETA_DTDOT / (NET_MASS         SR)) - ((2. X_STORE           | RK_DERIV
     S|             4,4                              I_TIME                       I_TIME:  |

1563 M| 5           ) / SR);                                                                 | RK_DERIV
     S|            2                                                                         |

1564 M| 5          F    = -NET_THETA_FORCE / (NET_MASS         NET_MASS         SR);         | RK_DERIV
     S|             4,5                                I_TIME           I_TIME               |

1565 M| 5          F    = F   ;                                                              | RK_DERIV
     S|             4,6    4,5                                                               |

1566 M| 5          F    = F   ;                                                              | RK_DERIV
     S|             4,7    4,5                                                               |

1567 M| 5          F    = DM1DOT_DR;                                                         | RK_DERIV
     S|             5,1                                                                      |

1568 M| 5          F    = DM1DOT_DUR;                                                        | RK_DERIV
     S|             5,2                                                                      |

1569 M| 5          F    = DM1DOT_DTDOT;                                                      | RK_DERIV
     S|             5,4                                                                      |

1570 M| 5          F    = DM2DOT_DR;                                                         | RK_DERIV
     S|             6,1                                                                      |

1571 M| 5          F    = DM2DOT_DUR;                                                        | RK_DERIV
     S|             6,2                                                                      |
```

```
STMT                                          SOURCE                                           CURRENT SCOPE

1572 M| 5              F      = DM2DOT_DTDOT;                                                  | RK_DERIV
     S|                 6,4                                                                   |

     E|                *                                                                      |
1573 M| 5              K = 0.;                                                                | RK_DERIV

1574 M| 5              DO FOR I_FK = 1 TO NUM_CONSTANT_PARAMETERS;                            | RK_DERIV

     E|                        .                                                              |
1575 M| 6                 IF ((STAGE_SEP = OFF) OR (I_FK > FS_FIXED_PARAMETERS)) THEN         | RK_DERIV

1576 M| 6                    DO;                                                              | RK_DERIV

1577 M| 7                       S5 = DT_DP     + DP_DP     ;                                  | RK_DERIV
     S|                                   I_FK        I_FK                                    |

1578 M| 7                       DFR_DP      = (S5 SIN_DELTA) + (DN_DP      COS_DELTA) - (SG  | RK_DERIV
     S|                               I_FK                           I_FK                     |

1578 M| 7                       DM_DP     );                                                  | RK_DERIV
     S|                              I_FK                                                     |

1579 M| 7                       DFTHETA_DP      = (S5 COS_DELTA) - (DN_DP      SIN_DELTA);    | RK_DERIV
     S|                                   I_FK                           I_FK                 |

1580 M| 6                    END;                                                             | RK_DERIV

1581 M| 6                 ELSE                                                                | RK_DERIV

1581 M| 6                    DO;                                                              | RK_DERIV

1582 M| 7                       DFR_DP      = 0.;                                             | RK_DERIV
     S|                               I_FK                                                    |

1583 M| 7                       DFTHETA_DP      = 0.;                                         | RK_DERIV
     S|                                   I_FK                                                |

1584 M| 6                    END;                                                             | RK_DERIV

     E|                         .                                                             |
1585 M| 6                 IF STAGE_SEP = OFF THEN                                             | RK_DERIV

1586 M| 6                    DO;                                                              | RK_DERIV

     E|                                .                                                      |
1587 M| 7                       IF TURBOJET_POWER = ON THEN                                   | RK_DERIV

1588 M| 7                          DM1DOT_DP      = -TJ_MAX_FUEL_AIR_RATIO CAP_PHI            | RK_DERIV
     S|                                     I_FK                                              |

1588 M| 7                          DMASSFLUX_DP1     ;                                        | RK_DERIV
     S|                                         I_FK                                          |

1589 M| 7                       ELSE                                                          | RK_DERIV
```

```
STMT                            SOURCE                                      CURRENT SCOPE

1589 M| 7                 DM1DOT_DP      = 0.;                              | RK_DERIV
     S|                            I_FK                                     :

     E|                        .                                            |
1590 M| 7              IF SCRAMJET_POWER = ON THEN                          | RK_DERIV

1591 M| 7                 DM2DOT_DP      = -SCRJ_MAX_FUEL_AIR_RATIO CAP_PHI | RK_DERIV
     S|                            I_FK                                     |

1591 M| 7                 DMASSFLUX_DP2    ;                                | RK_DERIV
     S|                                I_FK                                 |

1592 M| 7              ELSE                                                 | RK_DERIV

1592 M| 7                 DM2DOT_DP      = 0.;                              | RK_DERIV
     S|                            I_FK                                     |

1593 M| 7              DM3DOT_DP      = 0.;                                 | RK_DERIV
     S|                         I_FK                                        |

1594 M| 6           END;                                                    | RK_DERIV

1595 M| 6        ELSE                                                       | RK_DERIV

1595 M| 6           DO;                                                     | RK_DERIV

1596 M| 7              DM1DOT_DP      = 0.;                                 | RK_DERIV
     S|                         I_FK                                        |

1597 M| 7              DM2DOT_DP      = 0.;                                 | RK_DERIV
     S|                         I_FK                                        |

1598 M| 7              IF I_FK = 10 THEN                                    | RK_DERIV

1599 M| 7                 DM3DOT_DP      = -CAP_PHI / (ROCKET_ISP G0);      | RK_DERIV
     S|                            I_FK                                     |

1600 M| 7              ELSE                                                 | RK_DERIV

1600 M| 7                 DM3DOT_DP      = 0.;                              | RK_DERIV
     S|                            I_FK                                     |

1601 M| 6           END;                                                    | RK_DERIV

     E|                   .                                                 |
1602 M| 6        IF ((STAGE_SEP = OFF) OR (I_FK > FS_FIXED_PARAMETERS)) THEN | RK_DERIV

1603 M| 6           DO;                                                     | RK_DERIV

1604 M| 7              K        = (DFR_DP      - ((NET_R_FORCE DM_DP     ) / | RK_DERIV
     S|                 2,I_FK           I_FK                      I_FK     |

1604 M| 7              NET_MASS      )) / NET_MASS      ;                   | RK_DERIV
     S|                       I_TIME              I_TIME                    |
```

```
STMT                                    SOURCE                                      CURRENT SCOPE

1605 M| 7              K        = (DFTHETA_DP      - ((NET_THETA_FORCE DM_DP     )  | RK_DERIV
     S|                 4,I_FK              I_FK                        I_FK        |

1605 M| 7              / NET_MASS       )) / (NET_MASS       SR);                   | RK_DERIV
     S|                         I_TIME             I_TIME                           |

1606 M| 7              K        = DM1DOT_DP    ;                                    | RK_DERIV
     S|                 5,I_FK            I_FK                                      |

1607 M| 7              K        = DM2DOT_DP    ;                                    | RK_DERIV
     S|                 6,I_FK            I_FK                                      |

1608 M| 7              K        = DM3DOT_DP    ;                                    | RK_DERIV
     S|                 7,I_FK            I_FK                                      |

1609 M| 6           END;                                                            | RK_DERIV

1610 M| 5        END;                                                               | RK_DERIV

1611 M| 5        DFR_DUA  = ((DT_DUA  + DP_DUA ) SIN_VEHICLE_ANGLE) + (DN_DUA       | RK_DERIV
     S|                 1          1        1                             1         |

1611 M| 5        COS_VEHICLE_ANGLE) + NET_THETA_FORCE;                              | RK_DERIV

1612 M| 5        DFTHETA_DUA  = ((DT_DUA  + DP_DUA ) COS_VEHICLE_ANGLE) - (DN_DUA   | RK_DERIV
     S|                     1          1        1                             1     |

1612 M| 5        SIN_VEHICLE_ANGLE) - NET_R_FORCE - (NET_MASS       G);             | RK_DERIV
     S|                                                 I_TIME                      |

1613 M| 5        DFR_DUA  = DT_DUA  SIN_VEHICLE_ANGLE;                              | RK_DERIV
     S|                 2         2                                                 |

1614 M| 5        DFTHETA_DUA  = DT_DUA  COS_VEHICLE_ANGLE;                          | RK_DERIV
     S|                     2         2                                             |

1615 M| 5        IF ((FD_COUNT = 1) OR (FD_COUNT = 3) OR (I_TIME = 1)) THEN         | RK_DERIV

1616 M| 5           DO;                                                             | RK_DERIV

1617 M| 6              DO FOR I_FK = 1 TO NUM_CONTROLS;                             | RK_DERIV

1618 M| 7                 G_MAT             = DFR_DUA     / NET_MASS      ;         | RK_DERIV
     S|                      J_TIME:2,I_FK           I_FK          I_TIME           |

1619 M| 7                 G_MAT             = DFTHETA_DUA     / (NET_MASS      SR   | RK_DERIV
     S|                      J_TIME:4,I_FK               I_FK          I_TIME       |

1619 M| 7                 );                                                        | RK_DERIV

1620 M| 7                 G_MAT             = DM1DOT_DUA    ;                       | RK_DERIV
     S|                      J_TIME:5,I_FK              I_FK                        |

1621 M| 7                 G_MAT             = DM2DOT_DUA    ;                       | RK_DERIV
     S|                      I_TIME:6,I_FK              I_FK                        |
```

```
STMT                                      SOURCE                                       CURRENT SCOPE

1622 M|  7                    G_MAT                = DM3DOT_DUA     ;                | RK_DERIV
     S|                            J_TIME:7,I_FK               I_FK                  |

1623 M|  7                    IF G_LOAD > G_D THEN                                   | RK_DERIV

1624 M|  7                       L_SUB_U      = ((((DT_DUA      + DP_DUA     ) (NET_THRUST | RK_DERIV
     S|                                 I_FK             I_FK          I_FK          |

1624 M|  7                       + P_AERO)) + (DN_DUA      N_AERO)) L_CONST) / (G_LOAD ( | RK_DERIV
     S|                                                I_FK                          |

     E|                                                 2                            |
1624 M|  7                       (NET_MASS          GO) ));                          | RK_DERIV
     S|                                   I_TIME                                     |

1625 M|  7                    ELSE                                                   | RK_DERIV

1625 M|  7                       L_SUB_U      = 0.;                                  | RK_DERIV
     S|                                 I_FK                                         |

1626 M|  6                 END;                                                      | RK_DERIV

1627 M|  5              END;                                                         | RK_DERIV

1628 M|  5           IF FD_COUNT = 1 THEN                                            | RK_DERIV

1629 M|  5              DO;                                                          | RK_DERIV

     E|                    *    *                                                    |
1630 M|  6                 F1 = F;                                                   | RK_DERIV

     E|                    *    *                                                    |
1631 M|  6                 KA = K;                                                   | RK_DERIV

     E|                        -         -                                           |
1632 M|  6                 L_SUB_P_1 = L_SUB_P;                                      | RK_DERIV

     E|                        -         -                                           |
1633 M|  6                 L_SUB_U_1 = L_SUB_U;                                      | RK_DERIV

1634 M|  5              END;                                                         | RK_DERIV

1635 M|  5           IF FD_COUNT = 2 THEN                                            | RK_DERIV

1636 M|  5              DO;                                                          | RK_DERIV

     E|                    *    *                                                    |
1637 M|  6                 F2 = F;                                                   | RK_DERIV

     E|                    *    *                                                    |
1638 M|  6                 KB = K;                                                   | RK_DERIV

1639 M|  5              END;                                                         | RK_DERIV

1640 M|  5           IF FD_COUNT = 3 THEN                                            | RK_DERIV
```

```
STMT                          SOURCE                                        CURRENT SCOPE
1641 M| 5              DO;                                                  | RK_DERIV
                          *    *                                            |
     E|                                                                     
1642 M| 6                 F3 = F;                                           | RK_DERIV
                          *    *                                            |
     E|
1643 M| 6                 KC = K;                                           | RK_DERIV
                              -                -                            |
     E|
1644 M| 6                 L_SUB_P_3 = L_SUB_P;                              | RK_DERIV
                              -                -                            |
     E|
1645 M| 6                 L_SUB_U_3 = L_SUB_U;                              | RK_DERIV
1646 M| 5              END;                                                 | RK_DERIV
1647 M| 5           IF FD_COUNT = 4 THEN                                    | RK_DERIV
1648 M| 5              DO;                                                  | RK_DERIV
                          *    *                                            |
     E|
1649 M| 6                 F4 = F;                                           | RK_DERIV
                          *    *                                            |
     E|
1650 M| 6                 KD = K;                                           | RK_DERIV
1651 M| 5              END;                                                 | RK_DERIV
1652 M| 5           IF FD_COUNT = 0 THEN                                    | RK_DERIV
1653 M| 5              DO;                                                  | RK_DERIV
                          *    *                                            |
     E|
1654 M| 6                 F5 = F;                                           | RK_DERIV
                          *    *                                            |
     E|
1655 M| 6                 KE = K;                                           | RK_DERIV
                              -                -                            |
     E|
1656 M| 6                 L_SUB_P_5 = L_SUB_P;                              | RK_DERIV
1657 M| 6                 IF I_TIME = 1 THEN                                | RK_DERIV
                                  -              -                          |
     E|
1658 M| 6                    L_SUB_U_L = L_SUB_U;                           | RK_DERIV
1659 M| 5              END;                                                 | RK_DERIV
                  .                                                         |
     E|
1660 M| 5           IF ((OIT_FLAG = ON) AND (OMEGA_I_TIME  = I_TIME)) THEN  | RK_DERIV
     S|                                                  3                  |
1661 M| 5              NET_MASS       = KEEP_MASS;                          | RK_DERIV
     S|                        I_TIME                                       |
```

```
STMT                                  SOURCE                                         CURRENT SCOPE

1662 M| 4              END;                                                          | RK_DERIV

1663 M| 3            END;                                                            | RK_DERIV

1664 M| 2          END;                                                              | RK_DERIV

     E|              .                                                               |
1665 M| 2        IF LAMBDA_FLAG = ON THEN                                            | RK_DERIV

     E|             -                 *     -         -                -             |
1666 M| 2          LAMBDA_DOT = -(TRANSPOSE(F) LAMBDA_HOLD) - L_X_STORE        - L_SUB_X; | RK_DERIV
     S|                                                        I_TIME;               |

     E|                 .                                                            |
1667 M| 2        IF CAP_LAMBDA_1_FLAG = ON THEN                                      | RK_DERIV

1668 M| 2          DO;                                                               | RK_DERIV

1669 M| 3            IF I_FD = 1 THEN                                                | RK_DERIV

     E|                     *                    *         *                         |
1670 M| 3              CAP_LAMBDA_1_DOT = -TRANSPOSE(F1) CAP_LAMBDA_1_HOLD;          | RK_DERIV

1671 M| 3            IF I_FD = 2 THEN                                                | RK_DERIV

     E|                     *                    *         *                         |
1672 M| 3              CAP_LAMBDA_1_DOT = -TRANSPOSE(F2) CAP_LAMBDA_1_HOLD;          | RK_DERIV

1673 M| 3            IF I_FD = 3 THEN                                                | RK_DERIV

     E|                     *                    *         *                         |
1674 M| 3              CAP_LAMBDA_1_DOT = -TRANSPOSE(F3) CAP_LAMBDA_1_HOLD;          | RK_DERIV

1675 M| 3            IF I_FD = 4 THEN                                                | RK_DERIV

     E|                     *                    *         *                         |
1676 M| 3              CAP_LAMBDA_1_DOT = -TRANSPOSE(F4) CAP_LAMBDA_1_HOLD;          | RK_DERIV

1677 M| 3            IF I_FD = 5 THEN                                                | RK_DERIV

     E|                     *                    *         *                         |
1678 M| 3              CAP_LAMBDA_1_DOT = -TRANSPOSE(F5) CAP_LAMBDA_1_HOLD;          | RK_DERIV

1679 M| 2          END;                                                              | RK_DERIV

     E|                 .                                                            |
1680 M| 2        IF CAP_LAMBDA_2_FLAG = ON THEN                                      | RK_DERIV

1681 M| 2          DO;                                                               | RK_DERIV

1682 M| 3            IF I_FD = 1 THEN                                                | RK_DERIV

     E|                     *                    *       *                           |
1683 M| 3              CAP_LAMBDA_2_DOT = -TRANSPOSE(KA) CAP_LAMBDA_1       ;         | RK_DERIV
     S|                                                        J_TIME:               |
```

```
STMT                                       SOURCE                                              CURRENT SCOPE

1684 M| 3              IF I_FD = 2 THEN                                                        | RK_DERIV
     E|                   *                               *       *                   *        |
1685 M| 3                CAP_LAMBDA_2_DOT = (-TRANSPOSE(KB) (CAP_LAMBDA_1        + CAP_LAMBDA_1 | RK_DERIV
     S|                                                          J_TIME:                       |
1685 M| 3                          )) / 2.;                                                    | RK_DERIV
     S|                  J_TIME-1:                                                             |
1686 M| 3              IF I_FD = 3 THEN                                                        | RK_DERIV
     E|                   *                             *        *                             |
1687 M| 3                CAP_LAMBDA_2_DOT = -TRANSPOSE(KC) CAP_LAMBDA_1        ;               | RK_DERIV
     S|                                                        J_TIME:                         |
1688 M| 3              IF I_FD = 4 THEN                                                        | RK_DERIV
     E|                   *                               *       *                   *        |
1689 M| 3                CAP_LAMBDA_2_DOT = (-TRANSPOSE(KD) (CAP_LAMBDA_1        + CAP_LAMBDA_1 | RK_DERIV
     S|                                                          J_TIME:                       |
1689 M| 3                          )) / 2.;                                                    | RK_DERIV
     S|                  J_TIME-1:                                                             |
1690 M| 3              IF I_FD = 5 THEN                                                        | RK_DERIV
     E|                   *                             *        *                             |
1691 M| 3                CAP_LAMBDA_2_DOT = -TRANSPOSE(KE) CAP_LAMBDA_1        ;               | RK_DERIV
     S|                                                        J_TIME:                         |
1692 M| 2            END;                                                                      | RK_DERIV
     E|                                                                                        |
1693 M| 2         IF SGV_FLAG = ON THEN                                                        | RK_DERIV
1694 M| 2            DO;                                                                       | RK_DERIV
1695 M| 3              IF I_FD = 1 THEN                                                        | RK_DERIV
     E|                  -                  *    -                 -                           |
1696 M| 3                SGV_DOT = (TRANSPOSE(KA) LAMBDA        ) + L_SUB_P_1;                 | RK_DERIV
     S|                                          J_TIME:                                       |
1697 M| 3              IF I_FD = 3 THEN                                                        | RK_DERIV
     E|                  -                  *    -                 -                           |
1698 M| 3                SGV_DOT = (TRANSPOSE(KC) LAMBDA        ) + L_SUB_P_3;                 | RK_DERIV
     S|                                          J_TIME:                                       |
1699 M| 3              IF I_FD = 5 THEN                                                        | RK_DERIV
     E|                  -                  *    -                 -                           |
1700 M| 3                SGV_DOT = (TRANSPOSE(KE) LAMBDA        ) + L_SUB_P_5;                 | RK_DERIV
     S|                                          J_TIME:                                       |
```

```
STMT                                   SOURCE                                                      CURRENT SCOPE

1701 M| 2            END;                                                                          | RK_DERIV

     E|                .            .              .                                               |
1702 M| 2         IF (I_J_J_FLAG OR I_PSI_J_FLAG CR I_PSI_PSI_FLAG) = ON THEN                      | RK_DERIV

1703 M| 2            DO;                                                                           | RK_DERIV

1704 M| 3               U_TIME = J_TIME;                                                           | RK_DERIV

1705 M| 3               CALL U_COMPUTE;                                                            | RK_DERIV

     E|                       .                                                                    |
1706 M| 3               IF I_J_J_FLAG = ON THEN                                                    | RK_DERIV

     E|                                -             *      -                                      |
1707 M| 3                  I_J_J_DOT = (H_SUB_U       U) . H_SUB_U       ;                         | RK_DERIV
     S|                                     J_TIME:              J_TIME:                           |

     E|                       .                                                                    |
1708 M| 3               IF I_PSI_J_FLAG = ON THEN                                                  | RK_DERIV

     E|                     -                        *                 *           *   -           |
1709 M| 3                  I_PSI_J_DOT = (TRANSPOSE(CAP_LAMBDA_1        ) G_MAT        ) (U H_SUB_U | RK_DERIV
     S|                                                      J_TIME:         J_TIME:               |

1709 M| 3                         );                                                               | RK_DERIV
     S|                  J_TIME:                                                                   |

     E|                        .                                                                   |
1710 M| 3               IF I_PSI_PSI_FLAG = ON THEN                                                | RK_DERIV

1711 M| 3                  DO;                                                                     | RK_DERIV

     E|                        *                 *                *                                |
1712 M| 4                     M_1 = TRANSPOSE(CAP_LAMBDA_1        ) G_MAT        ;                 | RK_DERIV
     S|                                                J_TIME:         J_TIME:                     |

     E|                             *           *  *           *                                   |
1713 M| 4                     I_PSI_PSI_DOT = (M_1 U) TRANSPOSE(M_1);                              | RK_DERIV

1714 M| 3                  END;                                                                    | RK_DERIV

1715 M| 2            END;                                                                          | RK_DERIV

1716 M| 1         END;                                                                             | RK_DERIV

     C|  COSTATE DERIVATIVE COMPUTATIONS END HERE                                                  | RK_DERIV

     E|                 .                                                                          |
1717 M| 1      IF LAMBDA_FLAG = ON THEN                                                            | RK_DERIV

1718 M| 1         RK_D_VAL = LAMBDA_DOT     ;                                                      | RK_DERIV
     S|                                I_RK                                                        |
```

```
STMT                                  SOURCE                                      CURRENT SCOPE

     E|                .                                                         |
1719 M| 1        IF CAP_LAMBDA_1_FLAG = ON THEN                                  | RK_DERIV
1720 M| 1           RK_D_VAL = CAP_LAMBDA_1_DOT          ;                       | RK_DERIV
     S|                                    I_RK,J_RK                             |

     E|                .                                                         |
1721 M| 1        IF CAP_LAMBDA_2_FLAG = ON THEN                                  | RK_DERIV
1722 M| 1           RK_D_VAL = CAP_LAMBDA_2_DOT          ;                       | RK_DERIV
     S|                                    I_RK,J_RK                             |

     E|            .                                                             |
1723 M| 1        IF SSV_FLAG = ON THEN                                           | RK_DERIV
1724 M| 1           RK_D_VAL = SSV_DOT     ;                                     | RK_DERIV
     S|                            I_RK                                          |

     E|              .                                                           |
1725 M| 1        IF I_J_J_FLAG = ON THEN                                         | RK_DERIV
1726 M| 1           RK_D_VAL = I_J_J_DOT;                                        | RK_DERIV

     E|              .                                                           |
1727 M| 1        IF I_PSI_J_FLAG = ON THEN                                       | RK_DERIV
1728 M| 1           RK_D_VAL = I_PSI_J_DOT     ;                                 | RK_DERIV
     S|                                I_RK                                      |

     E|              .                                                           |
1729 M| 1        IF I_PSI_PSI_FLAG = ON THEN                                     | RK_DERIV
1730 M| 1           RK_D_VAL = I_PSI_PSI_DOT          ;                          | RK_DERIV
     S|                                  I_RK,J_RK                               |

1731 M|        END;                                                              | RK_DERIV

1732 M| CLOSE RK_DERIV;                                                          | RK_DERIV
```

**** B L O C K   S U M M A R Y ****

OUTER PROCEDURES CALLED
MODEL_DRIVER, U_COMPUTE

COMPOOL VARIABLES USED
NUM_STATES, NUM_CONSTANT_PARAMETERS, NUM_CONSTRAINTS, NUM_CONTROLS, Q_D, CAP_Q, G_D, CAP_CA, G0, X_STORE, EARTH_RADIUS
NUM_TRANS_PTS, OMEGA_I_TIME, EARTH_OMEGA, GAMMA0, R_0, GAM1, GAM2, GAM3, U_ACTIVE, DELIVERED_PLANFORM_AREA, P
FS_FIXED_PARAMETERS, DEGREES_PER_RADIAN, ROCKET_ISP, UNIVERSAL_G_CONSTANT, EARTH_MASS, TJ_MAX_FUEL_AIR_RATIO
SCRJ_MAX_FUEL_AIR_RATIO

OUTER VARIABLES USED
STATE_INTEGRATION_FLAG, I_RK, RK_D_VAL*, DYNAMIC_P0, G_LOAD, RK_D_VAL, X_DOT, RK_STEP*, RK_STEP, FIRST_DERIV_FLAG
FIRST_DERIV_FLAG*, LAMBDA_FLAG, CAP_LAMBDA_1_FLAG, RK_VAL_N, PARTIAL_DERIV_FLAG, PARTIAL_DERIV_FLAG*, VAL_1, CAP_LAMBDA_2_FLAG
I_TIME*, I_TIME, J_TIME*, I_CL, FD_FLAG, FD_FLAG*, L_TIME*, L_TIME, OIT_FLAG*, T_FLAG*, I_TIME_STORE, T_FLAG, OIT_FLAG

STMT SOURCE CURRENT SCOPE

```
TURBOJET_THRUST, SCRAMJET_THRUST, ROCKET_THRUST, FD_COUNT*, FD_COUNT, NET_MASS, NET_MASS*, HELD_FS_MASS, MD_MASS, MO_2*, MO, TO
DTO_DR, STAGE_SEP, T2, M2, OBLIQUE_SHOCK_FLAG, BETA_NOSE_SHOCK, THETA_NOSE, MO_2, RO_0, RO_2, NORMAL_SHOCK_FLAG, EXPANSION_FLAG
SUBSONIC_FLAG, DRO_DR, U2, TURBOJET_POWER, TURBOJET_ISP, SCRAMJET_POWER, SCRAMJET_ISP, SCRJ_MASS_CAPTURE_RATIO, DMCP_DM2
DCL1_DM0, DCL2_DM0, DCD1_DM0, DCD2_DM0, LIFT, WING_AREA, SS_PLANFORM_AREA, DRAG, NET_X_FORCE, N_AERO, CAP_PHI, DCL1_DAR, CL
SS_CL, DCD1_DAR, CD, SS_CD, P_AERO, DM_DP, DCL_DALPHA, DCL2_DUA1, DCD_DCL2, DCD2_DUA1, DP_DUA*, DM_DUA*, F*, NET_R_FORCE, F
NET_THETA_FORCE, K*, DP_DUA, SIN_VEHICLE_ANGLE, DM_DUA, COS_VEHICLE_ANGLE, G, J_TIME, G_MAT*, F1*, KA*, K, L_SUB_U_1*, F2*, KB*
F3*, KC*, L_SUB_U_3*, F4*, KD*, F5*, KE*, L_SUB_U_L*, LAMBDA_DOT*, L_X_STORE, I_FD, CAP_LAMBDA_1_DOT*, F1, F2, F3, F4, F5
CAP_LAMBDA_2_DOT*, KA, CAP_LAMBDA_1, KB, KC, KD, KE, SGV_FLAG, SGV_DOT*, LAMBDA, I_J_J_FLAG, I_PSI_J_FLAG, I_PSI_PSI_FLAG
U_TIME*, I_J_J_DOT*, H_SUB_U, U, I_PSI_J_DOT*, G_MAT, M_1*, I_PSI_PSI_DOT*, M_1, LAMBDA_DOT, J_RK, CAP_LAMBDA_1_DOT
CAP_LAMBDA_2_DOT, SGV_DOT, I_J_J_DOT, I_PSI_J_DOT, I_PSI_PSI_DOT
```

```
STMT                              SOURCE                                               CURRENT SCOPE

     C|  A FOURTH ORDER RUNGE/KUTTA COMPUTATION TECHNIQUE IS USED                      | RUNGE_KUTTA
1733 M|    START_RK_COLUMNS = 1;                                                       | RUNGE_KUTTA
     E|           .                                                                    |
1734 M|    IF STATE_INTEGRATION_FLAG = ON THEN                                         | RUNGE_KUTTA
1735 M|       DO;                                                                      | RUNGE_KUTTA
1736 M| 1       CALL STATE_DERIVS;                                                     | RUNGE_KUTTA
1737 M| 1       TIME_INTERVAL = -TIME_STEP;                                            | RUNGE_KUTTA
1738 M| 1       DO FOR I_RK = 1 TO NUM_STATES;                                         | RUNGE_KUTTA
1739 M| 2          RK_VAL_N        = X_STORE           ;                              | RUNGE_KUTTA
     S|                  I_RK,1             I_TIME:I_RK                                |
1740 M| 1       END;                                                                   | RUNGE_KUTTA
1741 M| 1       RK_VAL_N                  = INTEG_L;                                   | RUNGE_KUTTA
     S|                 NUM_STATES+1,1                                                 |
1742 M|       END;                                                                     | RUNGE_KUTTA
1743 M|    ELSE                                                                        | RUNGE_KUTTA
1743 M|       DO;                                                                      | RUNGE_KUTTA
     E|               .                .                  .                            |
1744 M| 1       IF (LAMBDA_FLAG OR CAP_LAMBDA_1_FLAG OR CAP_LAMBDA_2_FLAG) = ON THEN   | RUNGE_KUTTA
1745 M| 1          TIME_INTERVAL = TIME_STEP 2.;                                       | RUNGE_KUTTA
1746 M| 1       ELSE                                                                   | RUNGE_KUTTA
1746 M| 1          TIME_INTERVAL = TIME_STEP 4.;                                       | RUNGE_KUTTA
     E|               .                                                                |
1747 M| 1       IF LAMBDA_FLAG = ON THEN                                               | RUNGE_KUTTA
1748 M| 1          DO FOR I_RK = 1 TO NUM_STATES;                                      | RUNGE_KUTTA
1749 M| 2             RK_VAL_N        = LAMBDA            ;                           | RUNGE_KUTTA
     S|                     I_RK,1             J_TIME:I_RK                             |
1750 M| 1          END;                                                                | RUNGE_KUTTA
     E|                  .                                                             |
1751 M| 1       IF CAP_LAMBDA_1_FLAG = ON THEN                                         | RUNGE_KUTTA
1752 M| 1          DO FOR I_RK = 1 TO NUM_STATES;                                      | RUNGE_KUTTA
1753 M| 2             DO FOR J_RK = 1 TO NUM_CONSTRAINTS;                              | RUNGE_KUTTA
```

```
STMT                              SOURCE                                         CURRENT SCOPE

1754 M|  3            RK_VAL_N          = CAP_LAMBDA_1                   ;      | RUNGE_KUTTA
     S|                       I_RK,J_RK                J_TIME:I_RK,J_RK         |

1755 M|  2         END;                                                          | RUNGE_KUTTA

1756 M|  1      END;                                                             | RUNGE_KUTTA

     E|                 .                                                        |
1757 M|  1   IF CAP_LAMBDA_2_FLAG = ON THEN                                      | RUNGE_KUTTA

1758 M|  1      DO FOR I_RK = 1 TO NUM_CONSTANT_PARAMETERS;                      | RUNGE_KUTTA

1759 M|  2         DO FOR J_RK = 1 TO NUM_CONSTRAINTS;                           | RUNGE_KUTTA

1760 M|  3            RK_VAL_N          = CAP_LAMBDA_2          ;                | RUNGE_KUTTA
     S|                       I_RK,J_RK                I_RK,J_RK                 |

1761 M|  2         END;                                                          | RUNGE_KUTTA

1762 M|  1      END;                                                             | RUNGE_KUTTA

     E|            .                                                             |
1763 M|  1   IF SGV_FLAG = ON THEN                                               | RUNGE_KUTTA

1764 M|  1      DO FOR I_RK = 1 TO NUM_CONSTANT_PARAMETERS;                      | RUNGE_KUTTA

1765 M|  2         RK_VAL_N        = SMALL_G_VEC     ;                           | RUNGE_KUTTA
     S|                    I_RK,1                 I_RK                           |

1766 M|  1      END;                                                             | RUNGE_KUTTA

     E|              .                                                           |
1767 M|  1   IF I_J_J_FLAG = ON THEN                                             | RUNGE_KUTTA

1768 M|  1      RK_VAL_N     = I_J_J;                                            | RUNGE_KUTTA
     S|                 1,1                                                      |

     E|               .                                                          |
1769 M|  1   IF I_PSI_J_FLAG = ON THEN                                           | RUNGE_KUTTA

1770 M|  1      DO FOR I_RK = 1 TO NUM_CONSTRAINTS;                              | RUNGE_KUTTA

1771 M|  2         RK_VAL_N        = I_PSI_J     ;                               | RUNGE_KUTTA
     S|                    I_RK,1             I_RK                               |

1772 M|  1      END;                                                             | RUNGE_KUTTA

     E|                .                                                         |
1773 M|  1   IF I_PSI_PSI_FLAG = ON THEN                                         | RUNGE_KUTTA

1774 M|  1      DO FOR I_RK = 1 TO NUM_CONSTRAINTS;                              | RUNGE_KUTTA

1775 M|  2         DO FOR J_RK = I_RK TO NUM_CONSTRAINTS;                        | RUNGE_KUTTA
```

```
STMT                                    SOURCE                                                  CURRENT SCOPE

1776 M| 3            RK_VAL_N          = I_PSI_PSI          ;                                   | RUNGE_KUTTA
     S|                      I_RK,J_RK            I_RK,J_RK                                     |

1777 M| 2          END;                                                                         | RUNGE_KUTTA

1778 M| 1        END;                                                                           | RUNGE_KUTTA

1779 M| 1      RK_STEP = 0;                                                                     | RUNGE_KUTTA

1780 M| 1      VAL_1 = RK_ROWS RK_COLUMNS;                                                      | RUNGE_KUTTA

1781 M| 1      IF I_FD = 5 THEN                                                                 | RUNGE_KUTTA

1782 M| 1        I_FD = 1;                                                                      | RUNGE_KUTTA

1783 M| 1      ELSE                                                                             | RUNGE_KUTTA

1783 M| 1        I_FD = 3;                                                                      | RUNGE_KUTTA

     E|                .                                                                        |
1784 M| 1      FIRST_DERIV_FLAG = ON;                                                           | RUNGE_KUTTA

     E|                 .                                                                       |
1785 M| 1      PARTIAL_DERIV_FLAG = ON;                                                         | RUNGE_KUTTA

1786 M|      END;                                                                               | RUNGE_KUTTA

1787 M|    DO FOR I_RK = 1 TO RK_ROWS;                                                          | RUNGE_KUTTA

     E|               .                                                                         |
1788 M| 1    IF I_PSI_PSI_FLAG = ON THEN                                                        | RUNGE_KUTTA

1789 M| 1      START_RK_COLUMNS = I_RK;                                                         | RUNGE_KUTTA

1790 M| 1    DO FOR J_RK = START_RK_COLUMNS TO RK_COLUMNS;                                      | RUNGE_KUTTA

1791 M| 2      CALL RK_DERIV;                                                                   | RUNGE_KUTTA

1792 M| 2      K0          = TIME_INTERVAL RK_D_VAL;                                            | RUNGE_KUTTA
     S|          I_RK,J_RK                                                                      |

1793 M| 2      FIRST_RK_VAL_N          = RK_VAL_N          ;                                    | RUNGE_KUTTA
     S|                   I_RK,J_RK          I_RK,J_RK                                          |

1794 M| 2      RK_VAL_N          = RK_VAL_N          + (.5 K0          );                       | RUNGE_KUTTA
     S|             I_RK,J_RK          I_RK,J_RK          I_RK,J_RK                             |

1795 M| 1    END;                                                                               | RUNGE_KUTTA

1796 M|    END;                                                                                 | RUNGE_KUTTA

1797 M|    IF I_FD = 1 THEN                                                                     | RUNGE_KUTTA

1798 M|      DO;                                                                                | RUNGE_KUTTA
```

```
STMT                                  SOURCE                                         CURRENT SCOPE

     E|                .                                                            |
1799 M| 1        IF SGV_FLAG = ON THEN                                              | RUNGE_KUTTA
1800 M| 1           I_FD = 3;                                                       | RUNGE_KUTTA
1801 M| 1        ELSE                                                               | RUNGE_KUTTA
1801 M| 1           I_FD = 2;                                                       | RUNGE_KUTTA
1802 M|        END;                                                                 | RUNGE_KUTTA
1803 M|     ELSE                                                                    | RUNGE_KUTTA
1803 M|        I_FD = 4;                                                            | RUNGE_KUTTA
     E|             .                                                               |
1804 M|     FIRST_DERIV_FLAG = ON;                                                  | RUNGE_KUTTA
     E|              .                                                              |
1805 M|     PARTIAL_DERIV_FLAG = ON;                                                | RUNGE_KUTTA
1806 M|     DO FOR I_RK = 1 TO RK_ROWS;                                             | RUNGE_KUTTA
     E|                 .                                                           |
1807 M| 1      IF I_PSI_PSI_FLAG = ON THEN                                           | RUNGE_KUTTA
1808 M| 1         START_RK_COLUMNS = I_RK;                                           | RUNGE_KUTTA
1809 M| 1      DO FOR J_RK = START_RK_COLUMNS TO RK_COLUMNS;                         | RUNGE_KUTTA
1810 M| 2         CALL RK_DERIV;                                                     | RUNGE_KUTTA
1811 M| 2         K1          = TIME_INTERVAL RK_D_VAL;                              | RUNGE_KUTTA
     S|             I_RK,J_RK                                                       |
1812 M| 2         RK_VAL_N          = RK_VAL_N          + (.5 (K1          - K0          ));  | RUNGE_KUTTA
     S|                  I_RK,J_RK            I_RK,J_RK          I_RK,J_RK    I_RK,J_RK     |
1813 M| 1      END;                                                                  | RUNGE_KUTTA
1814 M|     END;                                                                    | RUNGE_KUTTA
     E|             .                                                               |
1815 M|     FIRST_DERIV_FLAG = ON;                                                  | RUNGE_KUTTA
1816 M|     DO FOR I_RK = 1 TO RK_ROWS;                                             | RUNGE_KUTTA
     E|                 .                                                           |
1817 M| 1      IF I_PSI_PSI_FLAG = ON THEN                                           | RUNGE_KUTTA
1818 M| 1         START_RK_COLUMNS = I_RK;                                           | RUNGE_KUTTA
1819 M| 1      DO FOR J_RK = START_RK_COLUMNS TO RK_COLUMNS;                         | RUNGE_KUTTA
1820 M| 2         CALL RK_DERIV;                                                     | RUNGE_KUTTA
```

```
STMT                                   SOURCE                                                     CURRENT SCOPE

1821 M| 2        K2            = TIME_INTERVAL RK_D_VAL;                                         | RUNGE_KUTTA
     S|            I_RK,J_RK                                                                     |

1822 M| 2        RK_VAL_N            = RK_VAL_N            + K2            - (.5 K1           ); | RUNGE_KUTTA
     S|                  I_RK,J_RK           I_RK,J_RK       I_RK,J_RK         I_RK,J_RK         |

1823 M| 1     END;                                                                               | RUNGE_KUTTA

1824 M|    END;                                                                                  | RUNGE_KUTTA

     E|                            .                 .                   .                       |
1825 M|    IF ((I_FD = 2) AND ((LAMBDA_FLAG OR CAP_LAMBDA_1_FLAG OR CAP_LAMBDA_2_FLAG) = ON)) THEN | RUNGE_KUTTA

1826 M|       I_FD = 3;                                                                          | RUNGE_KUTTA

1827 M|    ELSE                                                                                  | RUNGE_KUTTA

1827 M|       I_FD = 5;                                                                          | RUNGE_KUTTA

     E|          .                                                                               |
1828 M|    FIRST_DERIV_FLAG = ON;                                                                | RUNGE_KUTTA

     E|           .                                                                              |
1829 M|    PARTIAL_DERIV_FLAG = ON;                                                              | RUNGE_KUTTA

1830 M|    DO FOR I_RK = 1 TO RK_ROWS;                                                           | RUNGE_KUTTA

     E|                .                                                                         |
1831 M| 1     IF I_PSI_PSI_FLAG = ON THEN                                                       | RUNGE_KUTTA

1832 M| 1        START_RK_COLUMNS = I_RK;                                                        | RUNGE_KUTTA

1833 M| 1     DO FOR J_RK = START_RK_COLUMNS TO RK_COLUMNS;                                       | RUNGE_KUTTA

1834 M| 2        CALL RK_DERIV;                                                                  | RUNGE_KUTTA

1835 M| 2        K3            = TIME_INTERVAL RK_D_VAL;                                         | RUNGE_KUTTA
     S|            I_RK,J_RK                                                                     |

1836 M| 2        RK_VAL_N_PLUS_1          = FIRST_RK_VAL_N          + ((K0          + (2. (K1          + K2 | RUNGE_KUTTA
     S|                       I_RK,J_RK                 I_RK,J_RK         I_RK,J_RK          I_RK,J_RK      |

1836 M| 2                 )) + K3          ) / 6.);                                              | RUNGE_KUTTA
     S|            I_RK,J_RK        I_RK,J_RK                                                    |

1837 M| 1     END;                                                                              | RUNGE_KUTTA

1838 M|    END;                                                                                  | RUNGE_KUTTA

     E|                .                                                                         |
1839 M|    IF STATE_INTEGRATION_FLAG = ON THEN                                                   | RUNGE_KUTTA

1840 M|       DO;                                                                                | RUNGE_KUTTA

1841 M| 1        I_TIME = I_TIME + 1;                                                           | RUNGE_KUTTA
```

```
STMT                                  SOURCE                                                CURRENT SCOPE

1842 M|  1      DO FOR I_RK = 1 TO NUM_STATES;                                             | RUNGE_KUTTA

1843 M|  2         X_STORE            = RK_VAL_N_PLUS_1        ;                           | RUNGE_KUTTA
     S|                   I_TIME:I_RK                      I_RK,1                          |

1844 M|  1      END;                                                                       | RUNGE_KUTTA

1845 M|  1      INTEG_L = RK_VAL_N_PLUS_1              ;                                   | RUNGE_KUTTA
     S|                                NUM_STATES+1,1                                      |

1846 M|  1      NET_MASS       = X_STORE         + X_STORE         + X_STORE        + SS_DRY_MASS;  | RUNGE_KUTTA
     S|                 I_TIME          I_TIME:5          I_TIME:6          I_TIME:7            |

     E|                                                                                    |
1847 M|  1      IF ((STAGE_SEP = OFF) OR (OMEGA_I_TIME  = I_TIME)) THEN                    | RUNGE_KUTTA
     S|                                              3                                     |

1848 M|  1         NET_MASS       = NET_MASS       + FIRST_STAGE_DRY_MASS;                 | RUNGE_KUTTA
     S|                    I_TIME           I_TIME                                         |

1849 M|      END;                                                                          | RUNGE_KUTTA

1850 M| CLOSE RUNGE_KUTTA;                                                                 | RUNGE_KUTTA
```

```
**** B L O C K   S U M M A R Y ****

OUTER PROCEDURES CALLED
    STATE_DERIVS

COMPOOL VARIABLES USED
    NUM_STATES, X_STORE, NUM_CONSTRAINTS, NUM_CONSTANT_PARAMETERS, X_STORE*, OMEGA_I_TIME

OUTER VARIABLES USED
    STATE_INTEGRATION_FLAG, TIME_INTERVAL*, TIME_STEP, RK_VAL_N*, I_TIME, INTEG_L, LAMBDA_FLAG, CAP_LAMBDA_1_FLAG, CAP_LAMBDA_2_FLAG
    J_TIME, LAMBDA, CAP_LAMBDA_1, CAP_LAMBDA_2, SGV_FLAG, SMALL_G_VEC, I_J_J_FLAG, I_J_J, I_PSI_J_FLAG. I_PSI_J, I_PSI_PSI_FLAG
    I_PSI_PSI, RK_ROWS, RK_COLUMNS, I_FD, I_FD*, FIRST_DERIV_FLAG*, PARTIAL_DERIV_FLAG*, TIME_INTERVAL, RK_VAL_N, RK_VAL_N_PLUS_1*
    I_TIME*, RK_VAL_N_PLUS_1, INTEG_L*, NET_MASS*, SS_DRY_MASS, STAGE_SEP, NET_MASS, FIRST_STAGE_DRY_MASS
```

```
STMT                                   SOURCE                                                      CURRENT SCOPE

1851 M| TIME_SET:                                                                                | TIME_SET

1851 M| PROCEDURE;                                                                               | TIME_SET

1852 M|    DECLARE TSI INTEGER AUTOMATIC;                                                        | TIME_SET

1853 M|    PRESENT_TIME_STEP = NORM_TIME_STEP;                                                   | TIME_SET

1854 M|    DO FOR TSI = 1 TO NUM_TRANS_PTS;                                                      | TIME_SET

     E|                                                                                          |
1855 M| 1     IF STATE_INTEGRATION_FLAG = ON THEN                                               | TIME_SET

1856 M| 1        DO;                                                                            | TIME_SET

1857 M| 2           IF ((OMEGA_I_TIME    > (I_TIME - 4)) AND ((OMEGA_I_TIME    < I_TIME) OR (OMEGA_I_TIME  | TIME_SET
     S|                          TSI                                    TSI                      |

1857 M| 2               = I_TIME))) THEN                                                         | TIME_SET
     S|           TSI                                                                            |

1858 M| 2              PRESENT_TIME_STEP = NEG_TIME_STEP    ;                                    | TIME_SET
     S|                                               TSI                                        |

1859 M| 2           IF ((OMEGA_I_TIME    > I_TIME) AND ((OMEGA_I_TIME    < (I_TIME + 4)) OR (OMEGA_I_TIME  | TIME_SET
     S|                          TSI                              TSI                            |

1859 M| 2               = (I_TIME + 4)))) THEN                                                   | TIME_SET
     S|           TSI                                                                            |

1860 M| 2              PRESENT_TIME_STEP = POS_TIME_STEP    ;                                    | TIME_SET
     S|                                               TSI                                        |

1861 M| 1        END;                                                                            | TIME_SET

1862 M| 1     ELSE                                                                               | TIME_SET

1862 M| 1        DO;                                                                             | TIME_SET

1863 M| 2           IF (((OMEGA_I_TIME    > (I_TIME - 4)) OR (OMEGA_I_TIME    = (I_TIME - 4))) AND (  | TIME_SET
     S|                           TSI                                 TSI                        |

1863 M| 2           OMEGA_I_TIME    < I_TIME)) THEN                                              | TIME_SET
     S|                      TSI                                                                 |

1864 M| 2              PRESENT_TIME_STEP = NEG_TIME_STEP    ;                                    | TIME_SET
     S|                                               TSI                                        |

1865 M| 2           IF (((OMEGA_I_TIME    > I_TIME) OR (OMEGA_I_TIME    = I_TIME)) AND (OMEGA_I_TIME    < (  | TIME_SET
     S|                           TSI                          TSI                         TSI   |

1865 M| 2           I_TIME + 4))) THEN                                                           | TIME_SET

1866 M| 2              PRESENT_TIME_STEP = POS_TIME_STEP    ;                                    | TIME_SET
     S|                                               TSI                                        |
```

```
STMT                          SOURCE                                        CURRENT SCOPE

1867 M| 1        END;                                                       | TIME_SET

1868 M|      END;                                                           | TIME_SET

1869 M| CLOSE TIME_SET;                                                     | TIME_SET

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
   NORM_TIME_STEP, NUM_TRANS_PTS, OMEGA_I_TIME, NEG_TIME_STEP, POS_TIME_STEP

OUTER VARIABLES USED
   PRESENT_TIME_STEP*, STATE_INTEGRATION_FLAG, I_TIME
```

```
STMT                                  SOURCE                                                    CURRENT SCOPE
1870 M| ITERATION_DRIVER:                                                                        | ITERATION_DRIVER
1870 M| PROCEDURE;                                                                               | ITERATION_DRIVER
1871 M|    DECLARE CHECK1 SCALAR DOUBLE AUTOMATIC;                                               | ITERATION_DRIVER
1872 M|    DECLARE TANK1 SCALAR DOUBLE AUTOMATIC;                                                | ITERATION_DRIVER
1873 M|    DECLARE TANK2 SCALAR DOUBLE AUTOMATIC;                                                | ITERATION_DRIVER
1874 M|    DECLARE I_INIT INTEGER AUTOMATIC;                                                     | ITERATION_DRIVER
1875 M|    DECLARE T_XDOT VECTOR(NUM_STATES) DOUBLE STATIC;                                      | ITERATION_DRIVER
1876 M|    DECLARE M_VEC VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                                  | ITERATION_DRIVER
1877 M|    DECLARE DELTA_U_NEW ARRAY((STEP_DIM + 2) / 2) VECTOR(NUM_CONTROLS) DOUBLE AUTOMATIC;  | ITERATION_DRIVER
1878 M|    DECLARE I_RCP INTEGER AUTOMATIC;                                                      | ITERATION_DRIVER
1879 M|    DECLARE J_RCP INTEGER AUTOMATIC;                                                      | ITERATION_DRIVER
1880 M|    DECLARE J_TIME_STORE INTEGER STATIC;                                                  | ITERATION_DRIVER
1881 M|    DECLARE I_U INTEGER STATIC;                                                           | ITERATION_DRIVER
1882 M|    DECLARE J_U INTEGER AUTOMATIC;                                                        | ITERATION_DRIVER
1883 M|    DECLARE J_OMEGA INTEGER STATIC;                                                       | ITERATION_DRIVER
1884 M|    DECLARE I_OMEGA INTEGER STATIC;                                                       | ITERATION_DRIVER
1885 M|    DECLARE TIME_SIGN SCALAR DOUBLE STATIC;                                               | ITERATION_DRIVER
1886 M|    DECLARE SC1 SCALAR DOUBLE STATIC;                                                     | ITERATION_DRIVER
1887 M|    DECLARE SC2 SCALAR DOUBLE STATIC;                                                     | ITERATION_DRIVER
1888 M|    DECLARE SC3 SCALAR DOUBLE STATIC;                                                     | ITERATION_DRIVER
1889 M|    DECLARE SC4 SCALAR DOUBLE STATIC;                                                     | ITERATION_DRIVER
1890 M|    DECLARE DFP3 SCALAR DOUBLE AUTOMATIC;                                                 | ITERATION_DRIVER
1891 M|    DECLARE S9 SCALAR DOUBLE STATIC INITIAL(1.0001);                                      | ITERATION_DRIVER
1892 M|    DECLARE PAST_INTEG_L SCALAR DOUBLE STATIC;                                            | ITERATION_DRIVER
1893 M|    DECLARE LI4 INTEGER AUTOMATIC;                                                        | ITERATION_DRIVER
1894 M|    DECLARE LI5 INTEGER AUTOMATIC;                                                        | ITERATION_DRIVER
1895 M|    DECLARE LI_FLAG BIT(1) AUTOMATIC;                                                     | ITERATION_DRIVER
1896 M|    DECLARE OLD_SC3 SCALAR DOUBLE AUTOMATIC;                                              | ITERATION_DRIVER
```

```
STMT                                SOURCE                                             CURRENT SCOPE
1897 M|   DECLARE T_MASS SCALAR DOUBLE STATIC;                                        | ITERATION_DRIVER
1898 M|   DECLARE OLD_FINAL_STEP INTEGER STATIC;                                      | ITERATION_DRIVER
1899 M|   DECLARE FINAL_U_STORE ARRAY(4) VECTOR(NUM_CONTROLS) DOUBLE STATIC;          | ITERATION_DRIVER
1900 M|   DECLARE STEP_I_FLAG BIT(1) STATIC;                                          | ITERATION_DRIVER
1901 M|   DECLARE U_KEEP VECTOR(NUM_CONTROLS) DOUBLE AUTOMATIC;                       | ITERATION_DRIVER
1902 M|   DECLARE HOLD_PSI VECTOR(NUM_CONSTRAINTS) DOUBLE STATIC;                     | ITERATION_DRIVER
1903 M|   DECLARE U_NEW ARRAY(1001) VECTOR(NUM_CONTROLS) DOUBLE AUTOMATIC;            | ITERATION_DRIVER
1904 M|   DECLARE UV1 SCALAR DOUBLE AUTOMATIC;                                        | ITERATION_DRIVER
```

```
STMT                          SOURCE                                   CURRENT SCOPE
1905 M| STATE_INTEGRATION:                                           | STATE_INTEGRATION
1905 M| PROCEDURE;                                                   | STATE_INTEGRATION
     E|       .                                                      |
1906 M|    LAMBDA_FLAG = OFF;                                        | STATE_INTEGRATION
     E|           .                                                  |
1907 M|    CAP_LAMBDA_1_FLAG = OFF;                                  | STATE_INTEGRATION
     E|           .                                                  |
1908 M|    CAP_LAMBDA_2_FLAG = OFF;                                  | STATE_INTEGRATION
     E|     .                                                        |
1909 M|    SGV_FLAG = OFF;                                           | STATE_INTEGRATION
     E|      .                                                       |
1910 M|    I_J_J_FLAG = OFF;                                         | STATE_INTEGRATION
     E|       .                                                      |
1911 M|    I_PSI_J_FLAG = OFF;                                       | STATE_INTEGRATION
     E|        .                                                     |
1912 M|    I_PSI_PSI_FLAG = OFF;                                     | STATE_INTEGRATION
1913 M|    RK_ROWS = NUM_STATES + 1;                                 | STATE_INTEGRATION
1914 M|    RK_COLUMNS = 1;                                           | STATE_INTEGRATION
1915 M|    I_TIME = 1;                                               | STATE_INTEGRATION
     E|       .                                                      |
1916 M|    STAGE_SEP = ON;                                           | STATE_INTEGRATION
     E|         .                                                    |
1917 M|    TURBOJET_POWER = OFF;                                     | STATE_INTEGRATION
     E|         .                                                    |
1918 M|    SCRAMJET_POWER = OFF;                                     | STATE_INTEGRATION
     E|              .                                               |
1919 M|    STATE_INTEGRATION_FLAG = ON;                              | STATE_INTEGRATION
1920 M|    X_STORE    = ALT_FINAL;                                   | STATE_INTEGRATION
     S|           1:1                                                |
1921 M|    X_STORE    = U_R_FINAL;                                   | STATE_INTEGRATION
     S|           1:2                                                |
1922 M|    X_STORE    = THETA_FINAL;                                 | STATE_INTEGRATION
     S|           1:3                                                |
1923 M|    X_STORE    = U_THETA_FINAL / (ALT_FINAL + EARTH_RADIUS);  | STATE_INTEGRATION
     S|           1:4                                                |
```

```
STMT                                    SOURCE                                                   CURRENT SCOPE

1924 M|    X_STORE     = M1_FINAL;                                                             | STATE_INTEGRATION
     S|           1:5                                                                          |

1925 M|    X_STORE     = M2_FINAL;                                                             | STATE_INTEGRATION
     S|           1:6                                                                          |

1926 M|    X_STORE     = M3_FINAL;                                                             | STATE_INTEGRATION
     S|           1:7                                                                          |

1927 M|    INTEG_L = 0.;                                                                       | STATE_INTEGRATION

1928 M|    PAST_INTEG_L = 0.;                                                                  | STATE_INTEGRATION

     E|         .                                                                              |
1929 M|    RESHAPE_FLAG = ON;                                                                  | STATE_INTEGRATION

1930 M|    CALL MODEL_DRIVER;                                                                  | STATE_INTEGRATION

1931 M|    NET_MASS  = SS_DRY_MASS;                                                            | STATE_INTEGRATION
     S|            1                                                                           |

1932 M|    HELD_FS_MASS = FIRST_STAGE_DRY_MASS;                                                | STATE_INTEGRATION

     E|         -                                                                              |
1933 M|    [L_X_STORE] = 0.;                                                                   | STATE_INTEGRATION

     E|                    .                   .                                               |
1934 M|    DO WHILE (STATE_INTEGRATION_FLAG AND NOT OVER_STEP) = ON;                           | STATE_INTEGRATION

1935 M| 1     CALL MODEL_DRIVER;                                                              | STATE_INTEGRATION

1936 M| 1     IF DYNAMIC_PO > Q_D THEN                                                        | STATE_INTEGRATION

1937 M| 1        DO;                                                                          | STATE_INTEGRATION

1938 M| 2           SC1 = 2. CAP_Q (DYNAMIC_PO - Q_D);                                        | STATE_INTEGRATION

1939 M| 2           L_X_STORE         = ((RO_0 (X_STORE         + EARTH_RADIUS) ((X_STORE         -  | STATE_INTEGRATION
     S|                    I_TIME:1                 I_TIME:1                          I_TIME:4    |

     E|                          2                                                             |
1939 M| 2           EARTH_OMEGA) )) + ((DYNAMIC_PO DRO_DR) / RO_0)) SC1;                      | STATE_INTEGRATION

1940 M| 2           L_X_STORE         = RO_0 X_STORE         SC1;                             | STATE_INTEGRATION
     S|                    I_TIME:2               I_TIME:2                                     |

     E|                                                               2                        |
1941 M| 2           L_X_STORE         = RO_0 ((X_STORE         + EARTH_RADIUS) ) (X_STORE         -  | STATE_INTEGRATION
     S|                    I_TIME:4                 I_TIME:1                          I_TIME:4    |

1941 M| 2           EARTH_OMEGA) SC1;                                                         | STATE_INTEGRATION

1942 M| 1        END;                                                                         | STATE_INTEGRATION

1943 M| 1     CALL TIME_SET;                                                                  | STATE_INTEGRATION
```

```
STMT                                    SOURCE                                         CURRENT SCOPE

1944 M| 1    TIME_STEP = PRESENT_TIME_STEP;                                          | STATE_INTEGRATION
1945 M| 1    CALL RUNGE_KUTTA;                                                       | STATE_INTEGRATION
1946 M| 1    IF MOD(I_TIME, 4) = 1 THEN                                              | STATE_INTEGRATION
1947 M| 1       DO;                                                                  | STATE_INTEGRATION
1948 M| 2          IF ((X_STORE         < THETA_DOT_INITIAL) AND (X_STORE         < R_CUTOFF_CHECK)) THEN | STATE_INTEGRATION
     S|                    I_TIME:4                                    I_TIME:1                      |
1949 M| 2             DO;                                                                            | STATE_INTEGRATION
1950 M| 3                I_TIME = I_TIME - 4;                                                        | STATE_INTEGRATION
1951 M| 3                DO FOR I_OMEGA = 1 TO 4;                                                    | STATE_INTEGRATION
     E|                       -                  -                                                   |
1952 M| 4                   FINAL_U_STORE         = U_ACTIVE                 ;                        | STATE_INTEGRATION
     S|                                I_OMEGA:            I_TIME+I_OMEGA:                           |
1953 M| 3                END;                                                                        | STATE_INTEGRATION
1954 M| 3                SC2 = 2. NORM_TIME_STEP;                                                    | STATE_INTEGRATION
1955 M| 3                CALL TIME_SET;                                                              | STATE_INTEGRATION
1956 M| 3                DO FOR I_OMEGA = 1 TO MAX_CUTOFF_ITERATIONS;                                | STATE_INTEGRATION
1957 M| 4                   INTEG_L = PAST_INTEG_L;                                                  | STATE_INTEGRATION
1958 M| 4                   IF I_OMEGA = 1 THEN                                                      | STATE_INTEGRATION
1959 M| 4                      TIME_STEP = (PRESENT_TIME_STEP / 2.);                                 | STATE_INTEGRATION
1960 M| 4                   DO FOR J_OMEGA = 1 TO 4;                                                 | STATE_INTEGRATION
1961 M| 5                      CALL MODEL_DRIVER;                                                    | STATE_INTEGRATION
1962 M| 5                      CALL RUNGE_KUTTA;                                                     | STATE_INTEGRATION
1963 M| 5                      IF I_OMEGA = MAX_CUTOFF_ITERATIONS THEN                               | STATE_INTEGRATION
1964 M| 5                         DO;                                                                | STATE_INTEGRATION
1965 M| 6                            IF DYNAMIC_P0 -> Q_D THEN                                       | STATE_INTEGRATION
     E|                                    -                                                         |
1966 M| 6                               L_X_STORE        = 0.;                                        | STATE_INTEGRATION
     S|                                         I_TIME:                                              |
1967 M| 6                            ELSE                                                            | STATE_INTEGRATION
1967 M| 6                               DO;                                                          | STATE_INTEGRATION
```

```
STMT                                        SOURCE                                             CURRENT SCOPE
1968 M| 7                   SC1 = 2. CAP_Q (DYNAMIC_P0 - Q_D);                                | STATE_INTEGRATION
1969 M| 7                   L_X_STORE          = ((RO_0 (X_STORE          + EARTH_RADIUS) ((  | STATE_INTEGRATION
     S|                              I_TIME:1                  I_TIME:1                       |
     E|                                                  2                                    |
1969 M| 7                   X_STORE          - EARTH_OMEGA) )) + ((DYNAMIC_P0 DRO_DR) / RO_0  | STATE_INTEGRATION
     S|                            I_TIME:4                                                   |
1969 M| 7                   )) SC1;                                                           | STATE_INTEGRATION
1970 M| 7                   L_X_STORE          = RO_0 X_STORE          SC1;                   | STATE_INTEGRATION
     S|                              I_TIME:2               I_TIME:2                          |
     E|                                                                       2               |
1971 M| 7                   L_X_STORE          = RO_0 ((X_STORE          + EARTH_RADIUS) ) (  | STATE_INTEGRATION
     S|                              I_TIME:4                 I_TIME:1                        |
1971 M| 7                   X_STORE          - EARTH_OMEGA) SC1;                              | STATE_INTEGRATION
     S|                            I_TIME:4                                                   |
1972 M| 6                END;                                                                 | STATE_INTEGRATION
1973 M| 5          END;                                                                       | STATE_INTEGRATION
1974 M| 4    END;                                                                             | STATE_INTEGRATION
1975 M| 4    IF X_STORE          < THETA_DOT_INITIAL THEN                                      | STATE_INTEGRATION
     S|                I_TIME:4                                                               |
1976 M| 4       TIME_SIGN = -1.;                                                              | STATE_INTEGRATION
1977 M| 4    ELSE                                                                             | STATE_INTEGRATION
1977 M| 4       TIME_SIGN = 1.;                                                               | STATE_INTEGRATION
1978 M| 4    IF I_OMEGA ¬= MAX_CUTOFF_ITERATIONS THEN                                         | STATE_INTEGRATION
1979 M| 4       DO;                                                                           | STATE_INTEGRATION
     E|                                                                    1+I_OMEGA          |
1980 M| 5          TIME_STEP = TIME_STEP + ((TIME_SIGN PRESENT_TIME_STEP) / (2.          ));  | STATE_INTEGRATION
1981 M| 5          I_TIME = I_TIME - 4;                                                       | STATE_INTEGRATION
1982 M| 5          DO FOR J_OMEGA = 1 TO 4;                                                   | STATE_INTEGRATION
1983 M| 6             SC1 = TIME_STEP J_OMEGA;                                                | STATE_INTEGRATION
1984 M| 6             IF SC1 < SC2 THEN                                                       | STATE_INTEGRATION
     E|                   -                         -                   -                     |
1985 M| 6                U_ACTIVE                = U_ACTIVE          + ((FINAL_U_STORE   -    | STATE_INTEGRATION
     S|                         I_TIME+J_OMEGA:             I_TIME:                   2:      |
```

```
STMT                                   SOURCE                                                          CURRENT SCOPE

     E|                     -                                                                         |
1985 M| 6               U_ACTIVE        ) (SC1 / SC2));                                               | STATE_INTEGRATION
     S|                         I_TIME:                                                               |

1985 M| 6             ELSE                                                                            | STATE_INTEGRATION

     E|                     -                          -                -                             |
1986 M| 6               U_ACTIVE                  = FINAL_U_STORE  + ((FINAL_U_STORE   -               | STATE_INTEGRATION
     S|                         I_TIME+J_OMEGA:                 2:                  4:                |

     E|                       -                                                                       |
1985 M| 6               FINAL_U_STORE  ) ((SC1 - SC2) / SC2));                                        | STATE_INTEGRATION
     S|                            2:                                                                 |

1987 M| 5           END;                                                                              | STATE_INTEGRATION

1988 M| 4         END;                                                                                | STATE_INTEGRATION

1989 M| 3     END;                                                                                    | STATE_INTEGRATION

1990 M| 3     FINAL_TIME_STEP = TIME_STEP;                                                            | STATE_INTEGRATION

1991 M| 3     DO FOR I_OMEGA = 1 TO 4;                                                                | STATE_INTEGRATION

1992 M| 4       U_OLD_TIME                 = U_OLD_TIME         + (I_OMEGA FINAL_TIME_STEP);          | STATE_INTEGRATION
     S|                I_TIME-4+I_OMEGA             I_TIME-4                                          |

1993 M| 3     END;                                                                                    | STATE_INTEGRATION

1994 M| 3     DO FOR I_OMEGA = 1 TO I_TIME;                                                           | STATE_INTEGRATION

1995 M| 4       IF MOD(I_TIME, 2) = 1 THEN                                                            | STATE_INTEGRATION

1996 M| 4         U_J_OLD_TIME                 = U_OLD_TIME       ;                                   | STATE_INTEGRATION
     S|                   CEILING(I_OMEGA/2)             I_OMEGA                                      |

1997 M| 3     END;                                                                                    | STATE_INTEGRATION

1998 M| 3     L_FINAL = 0.;                                                                           | STATE_INTEGRATION

1999 M| 3     IF DYNAMIC_P0 > Q_D THEN                                                                | STATE_INTEGRATION

     E|                                         2                                                     |
2000 M| 3       L_FINAL = CAP_Q ((DYNAMIC_P0 - Q_D) );                                                | STATE_INTEGRATION

2001 M| 3     IF G_LOAD > G_D THEN                                                                    | STATE_INTEGRATION

     E|                                                     2                                         |
2002 M| 3       L_FINAL = L_FINAL + (CAP_CA (((G_LOAD - G_D) G0) ));                                  | STATE_INTEGRATION

     E|            .                                                                                  |
2003 M| 3     STATE_INTEGRATION_FLAG = OFF;                                                           | STATE_INTEGRATION

2004 M| 3     FINAL_STEP = I_TIME;                                                                    | STATE_INTEGRATION
```

```
STMT                                   SOURCE                                                        CURRENT SCOPE

2005 M| 2             END;                                                                           | STATE_INTEGRATION

2006 M| 1         END;                                                                               | STATE_INTEGRATION

     E|                 .                                                                            |
2007 M| 1     IF STATE_INTEGRATION_FLAG = ON THEN                                                    | STATE_INTEGRATION

2008 M| 1         DO;                                                                                | STATE_INTEGRATION

     C|  IT IS ASSUMED THAT EITHER THE SCRAMJET ONLY OR THE SCRAMJET AND                             | STATE_INTEGRATION
     C|  TURBOJET BOTH WILL BE ON AT STAGING BUT NOT JUST THE TURBOJET                               | STATE_INTEGRATION

2009 M| 2           IF MOD(I_TIME, 4) = 1 THEN                                                       | STATE_INTEGRATION

2010 M| 2             DO;                                                                            | STATE_INTEGRATION

     E|                       .                                                                      |
2011 M| 3               IF STAGE_SEP = ON THEN                                                       | STATE_INTEGRATION

2012 M| 3                 DO;                                                                        | STATE_INTEGRATION

2013 M| 4                   IF I_TIME = OMEGA_I_TIME  THEN                                           | STATE_INTEGRATION
     S|                                             3                                                |

2014 M| 4                     DO;                                                                    | STATE_INTEGRATION

     E|                             .                                                                |
2015 M| 5                       STAGE_SEP = OFF;                                                     | STATE_INTEGRATION

     E|                                .                                                             |
2016 M| 5                       SCRAMJET_POWER = ON;                                                 | STATE_INTEGRATION

2017 M| 4                     END;                                                                   | STATE_INTEGRATION

2018 M| 4                   IF I_TIME = OMEGA_I_TIME  THEN                                           | STATE_INTEGRATION
     S|                                             2                                                |

2019 M| 4                     DO;                                                                    | STATE_INTEGRATION

     E|                                .                                                             |
2020 M| 5                       TURBOJET_POWER = ON;                                                 | STATE_INTEGRATION

2021 M| 5                       IF I_TIME = OMEGA_I_TIME  THEN                                       | STATE_INTEGRATION
     S|                                                 1                                            |

     E|                                  .                                                           |
2022 M| 5                         SCRAMJET_POWER = OFF;                                              | STATE_INTEGRATION

2023 M| 4                     END;                                                                   | STATE_INTEGRATION

2024 M| 3                 END;                                                                       | STATE_INTEGRATION

2025 M| 3               ELSE                                                                         | STATE_INTEGRATION

2025 M| 3                 DO;                                                                        | STATE_INTEGRATION
```

```
STMT                                   SOURCE                                          CURRENT SCOPE

2026 M| 4                  IF I_TIME = OMEGA_I_TIME  THEN                             | STATE_INTEGRATION
     S|                                          2                                     |

     E|                        .                                                       |
2027 M| 4                    TURBOJET_POWER = ON;                                      | STATE_INTEGRATION

2028 M| 4                  IF I_TIME = OMEGA_I_TIME  THEN                             | STATE_INTEGRATION
     S|                                          1                                     |

     E|                        .                                                       |
2029 M| 4                    SCRAMJET_POWER = OFF;                                     | STATE_INTEGRATION

2030 M| 3                END;                                                          | STATE_INTEGRATION

2031 M| 3              PAST_INTEG_L = INTEG_L;                                         | STATE_INTEGRATION

2032 M| 2            END;                                                              | STATE_INTEGRATION

2033 M| 2          IF I_TIME > STEP_DIM THEN                                           | STATE_INTEGRATION

     E|              .                                                                 |
2034 M| 2            OVER_STEP = ON;                                                   | STATE_INTEGRATION

2035 M| 1        END;                                                                  | STATE_INTEGRATION

2036 M|    END;                                                                        | STATE_INTEGRATION

     E|          .                                                                     |
2037 M|    IF OVER_STEP = OFF THEN                                                     | STATE_INTEGRATION

2038 M|      DO;                                                                       | STATE_INTEGRATION

2039 M| 1        PSI  = X_STORE         ;                                              | STATE_INTEGRATION
     S|            1         I_TIME:1                                                  |

2040 M| 1        PSI  = X_STORE         ;                                              | STATE_INTEGRATION
     S|            2         I_TIME:2                                                  |

2041 M| 1        PSI  = X_STORE          - (HC_TANK_VOL HYDROCARBON_DENSITY);          | STATE_INTEGRATION
     S|            3         I_TIME:5                                                  |

2042 M| 1        PSI  = X_STORE          - (H2_TANK_VOL H2_DENSITY);                   | STATE_INTEGRATION
     S|            4         I_TIME:6                                                  |

2043 M| 1        PSI  = X_STORE          - P ;                                         | STATE_INTEGRATION
     S|            5         I_TIME:7    9                                             |

     E|                 -         *    -                                               |
2044 M| 1        PSI_MAG = PSI . (PSI_WEIGHT PSI);                                     | STATE_INTEGRATION

2045 M|        END;                                                                   | STATE_INTEGRATION

2046 M| CLOSE STATE_INTEGRATION;                                                      | STATE_INTEGRATION
```

```
STMT                                SOURCE                                CURRENT SCOPE

**** B L O C K   S U M M A R Y ****

OUTER PROCEDURES CALLED
    MODEL_DRIVER, TIME_SET, RUNGE_KUTTA

COMPOOL VARIABLES USED
    NUM_STATES, X_STORE*, ALT_FINAL, U_R_FINAL, THETA_FINAL, U_THETA_FINAL, EARTH_RADIUS, M1_FINAL, M2_FINAL, M3_FINAL, Q_D, CAP_Q
    X_STORE, EARTH_OMEGA, THETA_DOT_INITIAL, R_CUTOFF_CHECK, U_ACTIVE, NORM_TIME_STEP, MAX_CUTOFF_ITERATIONS, U_ACTIVE*, U_OLD_TIME*
    U_OLD_TIME, G_D, CAP_CA, G0, FINAL_STEP*, OMEGA_I_TIME, STEP_DIM, HYDROCARBON_DENSITY, H2_DENSITY, P, PSI_WEIGHT

OUTER VARIABLES USED
    LAMBDA_FLAG*, CAP_LAMBDA_1_FLAG*, CAP_LAMBDA_2_FLAG*, SGV_FLAG*, I_J_J_FLAG*, I_PSI_J_FLAG*, I_PSI_PSI_FLAG*, RK_ROWS*
    RK_COLUMNS*, I_TIME*, STAGE_SEP*, TURBOJET_POWER*, SCRAMJET_POWER*, STATE_INTEGRATION_FLAG*, INTEG_L*, PAST_INTEG_L*
    RESHAPE_FLAG*, NET_MASS*, SS_DRY_MASS, HELD_FS_MASS*, FIRST_STAGE_DRY_MASS, L_X_STORE*, STATE_INTEGRATION_FLAG, OVER_STEP
    DYNAMIC_P0, SC1*, I_TIME, RO_0, DRO_DR, SC1, TIME_STEP*, PRESENT_TIME_STEP, I_OMEGA*, I_OMEGA, FINAL_U_STORE*, SC2*
    PAST_INTEG_L, J_OMEGA*, TIME_SIGN*, TIME_STEP, TIME_SIGN, J_OMEGA, SC2, FINAL_U_STORE, FINAL_TIME_STEP*, FINAL_TIME_STEP
    U_J_OLD_TIME*, L_FINAL*, G_LOAD, L_FINAL, STAGE_SEP, INTEG_L, OVER_STEP*, PSI*, HC_TANK_VOL, H2_TANK_VOL, PSI_MAG*, PSI
```

```
STMT                                   SOURCE                                          CURRENT SCOPE

     E|            .                                                                  |
2047 M|   STEP_I_FLAG = ON;                                                           | ITERATION_DRIVER
2048 M|   OLD_FINAL_STEP = FINAL_STEP;                                                | ITERATION_DRIVER
2049 M|   SC4 = 0.;                                                                   | ITERATION_DRIVER
2050 M|   DO FOR I_U = 2 TO (STEP_DIM + 1);                                           | ITERATION_DRIVER
2051 M| 1    IF I_U < (FINAL_STEP - 3) THEN                                           | ITERATION_DRIVER
2052 M| 1       DO;                                                                   | ITERATION_DRIVER
2053 M| 2          I_TIME = I_U;                                                      | ITERATION_DRIVER
2054 M| 2          CALL TIME_SET;                                                     | ITERATION_DRIVER
2055 M| 2          SC4 = SC4 + PRESENT_TIME_STEP;                                     | ITERATION_DRIVER
2056 M| 1       END;                                                                  | ITERATION_DRIVER
2057 M| 1    ELSE                                                                     | ITERATION_DRIVER
2057 M| 1       DO;                                                                   | ITERATION_DRIVER
2058 M| 2          IF I_U < (FINAL_STEP + 1) THEN                                     | ITERATION_DRIVER
2059 M| 2             SC4 = SC4 + FINAL_TIME_STEP;                                    | ITERATION_DRIVER
2060 M| 2          ELSE                                                               | ITERATION_DRIVER
2060 M| 2             SC4 = SC4 + NORM_TIME_STEP;                                     | ITERATION_DRIVER
2061 M| 1       END;                                                                  | ITERATION_DRIVER
2062 M| 1    U_OLD_TIME    = SC4;                                                     | ITERATION_DRIVER
     S|                I_U                                                            |
2063 M| 1    IF ((ITERATION = 1) AND (MOD(I_U, 2) = 1)) THEN                          | ITERATION_DRIVER
2064 M| 1       U_TIME_KEEP                = U_OLD_TIME    ;                          | ITERATION_DRIVER
     S|                   CEILING(I_U/2)              I_U                             |
2065 M|   END;                                                                        | ITERATION_DRIVER
2066 M|   [HOLD_U_T] = [U_OLD_TIME];                                                  | ITERATION_DRIVER
     E|               .                                                               |
2067 M|   IF FIRST_ITERATION_FLAG = ON THEN                                           | ITERATION_DRIVER
2068 M|      DO;                                                                      | ITERATION_DRIVER
2069 M| 1       CALL STATE_INTEGRATION;                                               | ITERATION_DRIVER
```

```
STMT                              SOURCE                                          CURRENT SCOPE

     E|                 .                                                          |
2070 M| 1       FIRST_ITERATION_FLAG = OFF;                                        | ITERATION_DRIVER
2071 M|       END;                                                                 | ITERATION_DRIVER
2072 M|    ELSE                                                                    | ITERATION_DRIVER
2072 M|       DO;                                                                  | ITERATION_DRIVER
     E|                .                                                           |
2073 M| 1       IF OVER_STEP = OFF THEN                                           | ITERATION_DRIVER
     E|                                                        .                   |
2074 M| 1          DO FOR STEP_I = 1 TO MAX_STEP_I WHILE STEP_I_FLAG = ON;        | ITERATION_DRIVER
     E|                                           1-STEP_I                         |
2075 M| 2             STEP_SCALE_J = J_SCALE_FACTOR        ;                       | ITERATION_DRIVER
     E|                                               1-STEP_I                     |
2076 M| 2             STEP_SCALE_PSI = PSI_SCALE_FACTOR        ;                   | ITERATION_DRIVER
2077 M| 2             IF STEP_I = 1 THEN                                           | ITERATION_DRIVER
2078 M| 2                DO;                                                       | ITERATION_DRIVER
     E|                      -          -                                          |
2079 M| 3                   HOLD_PSI = PSI;                                        | ITERATION_DRIVER
2080 M| 3                   J_TIME = CEILING(FINAL_STEP / 2);                      | ITERATION_DRIVER
2081 M| 3                   M1_FLOW_FINAL_STEP = (-X_STORE             + X_STORE              ) /   | ITERATION_DRIVER
     S|                                         FINAL_STEP:5         FINAL_STEP-1:5         |
2081 M| 3                   FINAL_TIME_STEP;                                       | ITERATION_DRIVER
2082 M| 3                   M2_FLOW_FINAL_STEP = (-X_STORE             + X_STORE              ) /   | ITERATION_DRIVER
     S|                                         FINAL_STEP:6         FINAL_STEP-1:6         |
2082 M| 3                   FINAL_TIME_STEP;                                       | ITERATION_DRIVER
2083 M| 3                   M3_FLOW_FINAL_STEP = (-X_STORE             + X_STORE              ) /   | ITERATION_DRIVER
     S|                                         FINAL_STEP:7         FINAL_STEP-1:7         |
2083 M| 3                   FINAL_TIME_STEP;                                       | ITERATION_DRIVER
2084 M| 3                   MASS_FLOW_FINAL_STEP = (-NET_MASS           + NET_MASS            ) /   | ITERATION_DRIVER
     S|                                           FINAL_STEP          FINAL_STEP-1          |
2084 M| 3                   FINAL_TIME_STEP;                                       | ITERATION_DRIVER
2085 M| 3                   L_TIME = FINAL_STEP - 1;                               | ITERATION_DRIVER
2086 M| 3                   CALL MODEL_DRIVER;                                     | ITERATION_DRIVER
```

```
STMT                                          SOURCE                                                 CURRENT SCOPE

     E|                      .                                                                    |
2087 M| 3              STAGE_SEP = OFF;                                                           | ITERATION_DRIVER

     E|                        .                                                                  |
2088 M| 3              TURBOJET_POWER = ON;                                                       | ITERATION_DRIVER

     E|                        .                                                                  |
2089 M| 3              SCRAMJET_POWER = OFF;                                                      | ITERATION_DRIVER

2090 M| 3              DO FOR I_INIT = 1 TO NUM_CONSTANT_PARAMETERS;                              | ITERATION_DRIVER

2091 M| 4                 SMALL_G_VEC       = 0.;                                                 | ITERATION_DRIVER
     S|                            I_INIT                                                         |

2092 M| 3              END;                                                                       | ITERATION_DRIVER

2093 M| 3              I_J_J = 0.;                                                                | ITERATION_DRIVER

     E|                   -                                                                       |
2094 M| 3              I_PSI_J = 0.;                                                              | ITERATION_DRIVER

     E|                    *                                                                      |
2095 M| 3              I_PSI_PSI = 0.;                                                            | ITERATION_DRIVER

2096 M| 3              CHECK1 = (X_STORE             + EARTH_RADIUS) / ((NET_THETA_FORCE / NET_MASS | ITERATION_DRIVER
     S|                                FINAL_STEP:1                                               |

2096 M| 3                        ) - (2. X_STORE              X_STORE             ));             | ITERATION_DRIVER
     S|                FINAL_STEP                FINAL_STEP:2          FINAL_STEP:4               |

2097 M| 3              LAMBDA          = 0.;                                                      | ITERATION_DRIVER
     S|                      J_TIME:1                                                             |

2098 M| 3              LAMBDA          = 0.;                                                      | ITERATION_DRIVER
     S|                      J_TIME:2                                                             |

2099 M| 3              LAMBDA          = 0.;                                                      | ITERATION_DRIVER
     S|                      J_TIME:3                                                             |

2100 M| 3              LAMBDA          = CHECK1 (MASS_FLOW_FINAL_STEP - L_FINAL);                 | ITERATION_DRIVER
     S|                      J_TIME:4                                                             |

2101 M| 3              LAMBDA          = -1.;                                                     | ITERATION_DRIVER
     S|                      J_TIME:5                                                             |

2102 M| 3              LAMBDA          = -1.;                                                     | ITERATION_DRIVER
     S|                      J_TIME:6                                                             |

2103 M| 3              LAMBDA          = -1.;                                                     | ITERATION_DRIVER
     S|                      J_TIME:7                                                             |

     E|                     *                                                                     |
2104 M| 3              CAP_LAMBDA_1        = 0.;                                                  | ITERATION_DRIVER
     S|                            J_TIME:                                                        |
```

```
STMT                              SOURCE                                                 CURRENT SCOPE

2105 M| 3        CAP_LAMBDA_1             = -1.;                                        | ITERATION_DRIVER
     S|                      J_TIME:1,1                                                 |

2106 M| 3        CAP_LAMBDA_1             = -1.;                                        | ITERATION_DRIVER
     S|                      J_TIME:2,2                                                 |

2107 M| 3        CAP_LAMBDA_1             = -1.;                                        | ITERATION_DRIVER
     S|                      J_TIME:5,3                                                 |

2108 M| 3        CAP_LAMBDA_1             = -1.;                                        | ITERATION_DRIVER
     S|                      J_TIME:6,4                                                 |

2109 M| 3        CAP_LAMBDA_1             = -1.;                                        | ITERATION_DRIVER
     S|                      J_TIME:7,5                                                 |

2110 M| 3        CAP_LAMBDA_1             = CHECK1 X_STORE            ;                 | ITERATION_DRIVER
     S|                      J_TIME:4,1                 FINAL_STEP:2                    |

2111 M| 3        CAP_LAMBDA_1             = CHECK1 (((X_STORE             + EARTH_RADIUS) X_STORE | ITERATION_DRIVER
     S|                      J_TIME:4,2                    FINAL_STEP:1                 |

2111 M| 3                     X_STORE              ) + (NET_R_FORCE / NET_MASS          )); | ITERATION_DRIVER
     S|          FINAL_STEP:4         FINAL_STEP:4                        FINAL_STEP    |

2112 M| 3        CAP_LAMBDA_1             = CHECK1 M1_FLOW_FINAL_STEP;                  | ITERATION_DRIVER
     S|                      J_TIME:4,3                                                 |

2113 M| 3        CAP_LAMBDA_1             = CHECK1 M2_FLOW_FINAL_STEP;                  | ITERATION_DRIVER
     S|                      J_TIME:4,4                                                 |

2114 M| 3        CAP_LAMBDA_1             = CHECK1 M3_FLOW_FINAL_STEP;                  | ITERATION_DRIVER
     S|                      J_TIME:4,5                                                 |

     E|               *                                                                 |
2115 M| 3        CAP_LAMBDA_2 = 0.;                                                     | ITERATION_DRIVER

2116 M| 3        TANK2 = (P  P  P  SIN(P ) SIN(NOZZLE_ANGLE)) / SIN(P  + NOZZLE_ANGLE); | ITERATION_DRIVER
     S|                   8  8  2      1                        1                       |

2117 M| 3        TANK1 = .5452 P  TANK2;                                                | ITERATION_DRIVER
     S|                        7                                                        |

2118 M| 3        TANK2 = .0544 (1. - P ) TANK2;                                         | ITERATION_DRIVER
     S|                              7                                                  |

2119 M| 3        CHECK1 = SIN(NOZZLE_ANGLE) / (SIN(P  + NOZZLE_ANGLE) SIN(P ));         | ITERATION_DRIVER
     S|                                        1                     1                  |

2120 M| 3        CAP_LAMBDA_2     = CHECK1 TANK1;                                       | ITERATION_DRIVER
     S|                      1,3                                                        |

2121 M| 3        CAP_LAMBDA_2     = CHECK1 TANK2;                                       | ITERATION_DRIVER
     S|                      1,4                                                        |
```

```
STMT                                   SOURCE                                                  CURRENT SCOPE

2122 M| 3      CAP_LAMBDA_2     = TANK1 / P ;                                                  | ITERATION_DRIVER
     S|                    2,3             2                                                   |

2123 M| 3      CAP_LAMBDA_2     = TANK2 / P ;                                                  | ITERATION_DRIVER
     S|                    2,4             2                                                   |

2124 M| 3      CAP_LAMBDA_2     = TANK1 / P ;                                                  | ITERATION_DRIVER
     S|                    7,3             7                                                   |

2125 M| 3      CAP_LAMBDA_2     = TANK2 / (P  - 1.);                                           | ITERATION_DRIVER
     S|                    7,4              7                                                  |

2126 M| 3      CAP_LAMBDA_2     = (2. TANK1) / P ;                                             | ITERATION_DRIVER
     S|                    8,3                  8                                              |

2127 M| 3      CAP_LAMBDA_2     = (2. TANK2) / P ;                                             | ITERATION_DRIVER
     S|                    8,4                  8                                              |

2128 M| 3      CAP_LAMBDA_2     = 1.;                                                          | ITERATION_DRIVER
     S|                    9,5                                                                 |

2129 M| 3      SC1 = SIN(P  + NOZZLE_ANGLE);                                                   | ITERATION_DRIVER
     S|                 1                                                                      |

2130 M| 3      SC2 = SIN(NOZZLE_ANGLE);                                                        | ITERATION_DRIVER

2131 M| 3      SC3 = P  P  SC2 (.0136 - (.008148 P ));                                         | ITERATION_DRIVER
     S|              8  8                        7                                             |

2132 M| 3      SC4 = SIN(P ) / SC1;                                                            | ITERATION_DRIVER
     S|                 1                                                                      |

2133 M| 3      DM_DP  = (((((P  P ) / 4.) (1. - COS(P  + NOZZLE_ANGLE))) + (P  SC3)) (SC2 / ( | ITERATION_DRIVER
     S|             1       2  8                    1                       2                  |

     E|                                           2                                            |
2133 M| 3      SC1 SC1))) + (.25 (((P  SC2) / SC1)  ));                                        | ITERATION_DRIVER
     S|                           8                                                            |

2134 M| 3      IF SCRAMJET_MASS > (MIN_SCRJ_MASS_PER_FT P  S9) THEN                            | ITERATION_DRIVER
     S|                                                2                                       |

2135 M| 3         DFP3 = 15.2 + (4.6 / (P  P ));                                               | ITERATION_DRIVER
     S|                                3  3                                                    |

2136 M| 3      ELSE                                                                            | ITERATION_DRIVER

2136 M| 3         DFP3 = 0.;                                                                   | ITERATION_DRIVER

2137 M| 3      DM_DP  = (SCRAMJET_MASS / P ) + (5. P ) + ((P  / 4.) (1. + SC4 + (SC2 / SC1))) | ITERATION_DRIVER
     S|             2                    2         4       8                                   |

2137 M| 3      + (SC3 SC4);                                                                    | ITERATION_DRIVER
```

```
STMT                    SOURCE                                                              CURRENT SCOPE

2138 M| 3    DM_DP  = DFP3 P ;                                                             | ITERATION_DRIVER
     S|         3          2                                                               |

2139 M| 3    DM_DP  = 5. P ;                                                               | ITERATION_DRIVER
     S|         4        2                                                                 |

2140 M| 3    DM_DP  = P  / (8. TAN(P ));                                                   | ITERATION_DRIVER
     S|         5    5            6                                                        |

     E|                                  2                                                 |
2141 M| 3    DM_DP  = -(P  P ) / (16. (SIN(P ) ));                                         | ITERATION_DRIVER
     S|         6       5  5             6                                                 |

2142 M| 3    DM_DP  = P  P  P  SC4 SC2 (-.008148);                                         | ITERATION_DRIVER
     S|         7    2  8  8                                                               |

2143 M| 3    DM_DP  = ((P  / 4.) (1. + SC4 + (SC2 / SC1))) + ((2. P  SC3 SC4) / P ) + (P   | ITERATION_DRIVER
     S|         8      2                                      2             8      8       |

2143 M| 3    SC2 SC4 .5);                                                                  | ITERATION_DRIVER

2144 M| 3    DM_DP  = .04;                                                                 | ITERATION_DRIVER
     S|         9                                                                          |

2145 M| 3    DM_DP   = 1. / 3220.;                                                         | ITERATION_DRIVER
     S|         10                                                                         |

2146 M| 3    I_TIME_STORE = FINAL_STEP;                                                    | ITERATION_DRIVER

2147 M| 3    J_TIME_STORE = CEILING(FINAL_STEP / 2);                                       | ITERATION_DRIVER

2148 M| 3    TIME_STEP = FINAL_TIME_STEP;                                                  | ITERATION_DRIVER

2149 M| 3    RK_COLUMNS = 1;                                                               | ITERATION_DRIVER

2150 M| 3    DO WHILE I_TIME > 1;                                                          | ITERATION_DRIVER

2151 M| 4       I_TIME = I_TIME_STORE;                                                     | ITERATION_DRIVER

2152 M| 4       J_TIME = J_TIME_STORE;                                                     | ITERATION_DRIVER

2153 M| 4       IF I_TIME = OMEGA_I_TIME  THEN                                             | ITERATION_DRIVER
     S|                               1                                                    |

     E|               .                                                                    |
2154 M| 4          SCRAMJET_POWER = ON;                                                    | ITERATION_DRIVER

2155 M| 4       IF I_TIME = OMEGA_I_TIME  THEN                                             | ITERATION_DRIVER
     S|                               2                                                    |

     E|               .                                                                    |
2156 M| 4          TURBOJET_POWER = OFF;                                                   | ITERATION_DRIVER

2157 M| 4       IF I_TIME = OMEGA_I_TIME  THEN                                             | ITERATION_DRIVER
     S|                               3                                                    |
```

```
 STMT                           SOURCE                                                   CURRENT SCOPE
 2153 M| 4            DO;                                                              | ITERATION_DRIVER
      E|                 .                                                             |
 2159 M| 5               SCRAMJET_POWER = OFF;                                         | ITERATION_DRIVER
      E|                  .                                                            |
 2160 M| 5               STAGE_SEP = ON;                                               | ITERATION_DRIVER
 2161 M| 4            END;                                                             | ITERATION_DRIVER
      E|                .                                                              |
 2162 M| 4         LAMBDA_FLAG = ON;                                                   | ITERATION_DRIVER
 2163 M| 4         RK_ROWS = NUM_STATES;                                               | ITERATION_DRIVER
      E|             .                                                                 |
 2164 M| 4         FD_FLAG = ON;                                                       | ITERATION_DRIVER
 2165 M| 4         DO FOR I_CL = 1 TO 2;                                               | ITERATION_DRIVER
 2166 M| 5            CALL RUNGE_KUTTA;                                                | ITERATION_DRIVER
 2167 M| 5            DO FOR I_STORE = 1 TO NUM_STATES;                                | ITERATION_DRIVER
 2168 M| 6               LAMBDA              = RK_VAL_N_PLUS_1          ;             | ITERATION_DRIVER
      S|                     J_TIME:I_STORE                   I_STORE,1                |
 2169 M| 5            END;                                                             | ITERATION_DRIVER
 2170 M| 4         END;                                                                | ITERATION_DRIVER
      E|                .                                                              |
 2171 M| 4         LAMBDA_FLAG = OFF;                                                  | ITERATION_DRIVER
 2172 M| 4         I_TIME = I_TIME_STORE;                                              | ITERATION_DRIVER
 2173 M| 4         J_TIME = J_TIME_STORE;                                              | ITERATION_DRIVER
      E|                   .                                                           |
 2174 M| 4         CAP_LAMBDA_1_FLAG = ON;                                             | ITERATION_DRIVER
 2175 M| 4         RK_COLUMNS = NUM_CONSTRAINTS;                                       | ITERATION_DRIVER
 2176 M| 4         DO FOR I_CL = 1 TO 2;                                               | ITERATION_DRIVER
 2177 M| 5            CALL RUNGE_KUTTA;                                                | ITERATION_DRIVER
 2178 M| 5            DO FOR I_STORE = 1 TO NUM_STATES;                                | ITERATION_DRIVER
 2179 M| 6               DO FOR J_STORE = 1 TO NUM_CONSTRAINTS;                        | ITERATION_DRIVER
 2180 M| 7                  CAP_LAMBDA_1                      = RK_VAL_N_PLUS_1         | ITERATION_DRIVER
      S|                               J_TIME:I_STORE,J_STORE                  I_STORE, |
```

```
STMT                          SOURCE                                          CURRENT SCOPE
2180 M| 7                                 ;                                   | ITERATION_DRIVER
     S|                          J_STORE                                      |
2181 M| 6                      END;                                           | ITERATION_DRIVER
2182 M| 5                   END;                                              | ITERATION_DRIVER
2183 M| 4                END;                                                 | ITERATION_DRIVER
     E|                                                                       |
2184 M| 4                CAP_LAMBDA_1_FLAG = OFF;                             | ITERATION_DRIVER
2185 M| 4                I_TIME = I_TIME_STORE;                               | ITERATION_DRIVER
2186 M| 4                J_TIME = J_TIME_STORE;                               | ITERATION_DRIVER
     E|                                                                       |
2187 M| 4                CAP_LAMBDA_2_FLAG = ON;                              | ITERATION_DRIVER
2188 M| 4                RK_ROWS = NUM_CONSTANT_PARAMETERS;                   | ITERATION_DRIVER
2189 M| 4                DO FOR I_CL = 1 TO 2;                                | ITERATION_DRIVER
2190 M| 5                   CALL RUNGE_KUTTA;                                 | ITERATION_DRIVER
2191 M| 5                   DO FOR I_STORE = 1 TO NUM_CONSTANT_PARAMETERS;    | ITERATION_DRIVER
2192 M| 6                      DO FOR J_STORE = 1 TO NUM_CONSTRAINTS;         | ITERATION_DRIVER
2193 M| 7                         CAP_LAMBDA_2                  = RK_VAL_N_PLUS_1                 ;   | ITERATION_DRIVER
     S|                                     I_STORE,J_STORE                   I_STORE,J_STORE     |
2194 M| 6                      END;                                           | ITERATION_DRIVER
2195 M| 5                   END;                                              | ITERATION_DRIVER
2196 M| 4                END;                                                 | ITERATION_DRIVER
     E|                                                                       |
2197 M| 4                CAP_LAMBDA_2_FLAG = OFF;                             | ITERATION_DRIVER
2198 M| 4                I_TIME = I_TIME_STORE;                               | ITERATION_DRIVER
2199 M| 4                J_TIME = J_TIME_STORE;                               | ITERATION_DRIVER
2200 M| 4                DO FOR I_CL = 1 TO 2;                                | ITERATION_DRIVER
2201 M| 5                   CALL HAM_SUB_U;                                   | ITERATION_DRIVER
2202 M| 5                   I_TIME = I_TIME - 2;                              | ITERATION_DRIVER
2203 M| 5                   J_TIME = CEILING(I_TIME / 2);                     | ITERATION_DRIVER
2204 M| 4                END;                                                 | ITERATION_DRIVER
```

```
STMT                          SOURCE                                              CURRENT SCOPE
2205 M| 4          IF I_TIME = 1 THEN                                             | ITERATION_DRIVER
2206 M| 4             CALL HAM_SUB_U;                                             | ITERATION_DRIVER
2207 M| 4          I_TIME = I_TIME_STORE;                                         | ITERATION_DRIVER
2208 M| 4          J_TIME = J_TIME_STORE;                                         | ITERATION_DRIVER
     E|               .                                                           |
2209 M| 4          SGV_FLAG = ON;                                                 | ITERATION_DRIVER
2210 M| 4          RK_COLUMNS = 1;                                                | ITERATION_DRIVER
2211 M| 4          CALL RUNGE_KUTTA;                                              | ITERATION_DRIVER
2212 M| 4          DO FOR I_STORE = 1 TO NUM_CONSTANT_PARAMETERS;                 | ITERATION_DRIVER
2213 M| 5             SMALL_G_VEC        = RK_VAL_N_PLUS_1          ;            | ITERATION_DRIVER
     S|                        I_STORE                    I_STORE,1               |
2214 M| 4          END;                                                           | ITERATION_DRIVER
     E|               .                                                           |
2215 M| 4          SGV_FLAG = OFF;                                                | ITERATION_DRIVER
2216 M| 4          I_TIME_STORE = I_TIME;                                         | ITERATION_DRIVER
2217 M| 4          J_TIME_STORE = CEILING(I_TIME_STORE / 2);                      | ITERATION_DRIVER
2218 M| 4          CALL TIME_SET;                                                 | ITERATION_DRIVER
2219 M| 4          TIME_STEP = PRESENT_TIME_STEP;                                 | ITERATION_DRIVER
2220 M| 3       END;                                                              | ITERATION_DRIVER
2221 M| 3       I_TIME = FINAL_STEP;                                              | ITERATION_DRIVER
2222 M| 3       I_TIME_STORE = FINAL_STEP;                                        | ITERATION_DRIVER
2223 M| 3       J_TIME_STORE = CEILING(FINAL_STEP / 2);                           | ITERATION_DRIVER
2224 M| 3       TIME_STEP = FINAL_TIME_STEP;                                      | ITERATION_DRIVER
     E|           .                                                               |
2225 M| 3       STAGE_SEP = OFF;                                                  | ITERATION_DRIVER
     E|               .                                                           |
2226 M| 3       TURBOJET_POWER = ON;                                              | ITERATION_DRIVER
     E|               .                                                           |
2227 M| 3       SCRAMJET_POWER = OFF;                                             | ITERATION_DRIVER
2228 M| 3       DO WHILE I_TIME > 1;                                              | ITERATION_DRIVER
2229 M| 4          I_TIME = I_TIME_STORE;                                         | ITERATION_DRIVER
```

```
STMT                              SOURCE                                              CURRENT SCOPE
2230 M| 4           J_TIME = J_TIME_STORE;                                          | ITERATION_DRIVER
2231 M| 4           IF I_TIME = OMEGA_I_TIME  THEN                                  | ITERATION_DRIVER
     S|                                  1                                          |
     E|                  .                                                          |
2232 M| 4              SCRAMJET_POWER = ON;                                         | ITERATION_DRIVER
2233 M| 4           IF I_TIME = OMEGA_I_TIME  THEN                                  | ITERATION_DRIVER
     S|                                  2                                          |
     E|                  .                                                          |
2234 M| 4              TURBOJET_POWER = OFF;                                        | ITERATION_DRIVER
2235 M| 4           IF I_TIME = OMEGA_I_TIME  THEN                                  | ITERATION_DRIVER
     S|                                  3                                          |
2236 M| 4              DO;                                                          | ITERATION_DRIVER
     E|                      .                                                      |
2237 M| 5                 SCRAMJET_POWER = OFF;                                     | ITERATION_DRIVER
     E|                      .                                                      |
2238 M| 5                 STAGE_SEP = ON;                                           | ITERATION_DRIVER
2239 M| 4              END;                                                         | ITERATION_DRIVER
     E|               .                                                             |
2240 M| 4           I_J_J_FLAG = ON;                                                | ITERATION_DRIVER
2241 M| 4           RK_ROWS = 1;                                                    | ITERATION_DRIVER
2242 M| 4           RK_COLUMNS = 1;                                                 | ITERATION_DRIVER
2243 M| 4           CALL RUNGE_KUTTA;                                               | ITERATION_DRIVER
2244 M| 4           I_J_J = RK_VAL_N_PLUS_1   ;                                     | ITERATION_DRIVER
     S|                                 1,1                                         |
     E|               .                                                             |
2245 M| 4           I_J_J_FLAG = OFF;                                               | ITERATION_DRIVER
2246 M| 4           I_TIME = I_TIME_STORE;                                          | ITERATION_DRIVER
2247 M| 4           J_TIME = J_TIME_STORE;                                          | ITERATION_DRIVER
     E|                .                                                            |
2248 M| 4           I_PSI_J_FLAG = ON;                                              | ITERATION_DRIVER
2249 M| 4           RK_ROWS = NUM_CONSTRAINTS;                                      | ITERATION_DRIVER
2250 M| 4           CALL RUNGE_KUTTA;                                               | ITERATION_DRIVER
2251 M| 4           DO FOR I_STORE = 1 TO NUM_CONSTRAINTS;                          | ITERATION_DRIVER
```

```
STMT                                  SOURCE                                              CURRENT SCOPE

2252 M| 5          I_PSI_J          = RK_VAL_N_PLUS_1          ;                          | ITERATION_DRIVER
     S|                 I_STORE                      I_STORE,1                            |
2253 M| 4       END;                                                                      | ITERATION_DRIVER
     E|              .                                                                    |
2254 M| 4       I_PSI_J_FLAG = OFF;                                                       | ITERATION_DRIVER
2255 M| 4       I_TIME = I_TIME_STORE;                                                    | ITERATION_DRIVER
2256 M| 4       J_TIME = J_TIME_STORE;                                                    | ITERATION_DRIVER
     E|               .                                                                   |
2257 M| 4       I_PSI_PSI_FLAG = ON;                                                      | ITERATION_DRIVER
2258 M| 4       RK_COLUMNS = NUM_CONSTRAINTS;                                             | ITERATION_DRIVER
2259 M| 4       CALL RUNGE_KUTTA;                                                         | ITERATION_DRIVER
2260 M| 4       DO FOR I_STORE = 1 TO NUM_CONSTRAINTS;                                    | ITERATION_DRIVER
2261 M| 5          DO FOR J_STORE = I_STORE TO NUM_CONSTRAINTS;                           | ITERATION_DRIVER
2262 M| 6             I_PSI_PSI                = RK_VAL_N_PLUS_1                  ;       | ITERATION_DRIVER
     S|                      I_STORE,J_STORE                    I_STORE,J_STORE           |
2263 M| 5          END;                                                                   | ITERATION_DRIVER
2264 M| 4       END;                                                                      | ITERATION_DRIVER
     E|               .                                                                   |
2265 M| 4       I_PSI_PSI_FLAG = OFF;                                                     | ITERATION_DRIVER
2266 M| 4       I_TIME_STORE = I_TIME;                                                    | ITERATION_DRIVER
2267 M| 4       J_TIME_STORE = CEILING(I_TIME_STORE / 2);                                 | ITERATION_DRIVER
2268 M| 4       CALL TIME_SET;                                                            | ITERATION_DRIVER
2269 M| 4       TIME_STEP = PRESENT_TIME_STEP;                                            | ITERATION_DRIV
2270 M| 3    END;                                                                         | ITERATION_DRIVER
2271 M| 3    DO FOR I_STORE = 2 TO NUM_CONSTRAINTS;                                       | ITERATION_DRIVER
2272 M| 4       DO FOR J_STORE = 1 TO (I_STORE - 1);                                      | ITERATION_DRIVER
2273 M| 5          I_PSI_PSI                = I_PSI_PSI                  ;                | ITERATION_DRIVER
     S|                   I_STORE,J_STORE              J_STORE,I_STORE                    |
2274 M| 4       END;                                                                      | ITERATION_DRIVER
2275 M| 3    END;                                                                         | ITERATION_DRIVER
2276 M| 3    DO FOR CONFIG_INDEX = 1 TO NUM_TRANS_PTS;                                    | ITERATION_DRIVER
```

```
STMT                                   SOURCE                                          CURRENT SCOPE

2277 M| 4        J_TIME = CEILING(OMEGA_I_TIME              / 2);                       | ITERATION_DRIVER
     S|                                  CONFIG_INDEX                                   |

2278 M| 4        IF ((OMEGA_I_TIME              > 1) AND (OMEGA_I_TIME            < (    | ITERATION_DRIVER
     S|                       CONFIG_INDEX                          CONFIG_INDEX        |

2278 M| 4        FINAL_STEP - 1))) THEN                                                 | ITERATION_DRIVER

2279 M| 4           DO;                                                                 | ITERATION_DRIVER

2280 M| 5              I_TIME = OMEGA_I_TIME              ;                             | ITERATION_DRIVER
     S|                                 CONFIG_INDEX                                    |

     E|                -                                                                |
2281 M| 5              T_XDOT = 0.;                                                     | ITERATION_DRIVER

     E|                -                                                                |
2282 M| 5              X_DOT = 0.;                                                      | ITERATION_DRIVER

     E|                  .                                                              |
2283 M| 5              STAGE_SEP = OFF;                                                 | ITERATION_DRIVER

2284 M| 5              IF I_TIME = OMEGA_I_TIME  THEN                                   | ITERATION_DRIVER
     S|                                       1                                         |

     E|                       .                                                         |
2285 M| 5                 SCRAMJET_POWER = OFF;                                         | ITERATION_DRIVER

2286 M| 5              ELSE                                                             | ITERATION_DRIVER

     E|                       .                                                         |
2286 M| 5                 SCRAMJET_POWER = ON;                                          | ITERATION_DRIVER

2287 M| 5              IF I_TIME < OMEGA_I_TIME  THEN                                   | ITERATION_DRIVER
     S|                                       2                                         |

     E|                       .                                                         |
2288 M| 5                 TURBOJET_POWER = OFF;                                         | ITERATION_DRIVER

2289 M| 5              ELSE                                                             | ITERATION_DRIVER

     E|                       .                                                         |
2289 M| 5                 TURBOJET_POWER = ON;                                          | ITERATION_DRIVER

     E|                         .                                                       |
2290 M| 5              STATE_INTEGRATION_FLAG = ON;                                     | ITERATION_DRIVER

2291 M| 5              CALL MODEL_DRIVER;                                               | ITERATION_DRIVER

     E|                         .                                                       |
2292 M| 5              STATE_INTEGRATION_FLAG = OFF;                                    | ITERATION_DRIVER

2293 M| 5              T_XDOT  = NET_R_FORCE / NET_MASS       ;                         | ITERATION_DRIVER
     S|                    2                          I_TIME                            |
```

```
STMT                           SOURCE                                                     CURRENT SCOPE

2294 M| 5     T_XDOT  = NET_THETA_FORCE / (NET_MASS       (X_STORE          +           | ITERATION_DRIVER
     S|             4                          I_TIME          I_TIME:1                  |

2294 M| 5     EARTH_RADIUS));                                                          | ITERATION_DRIVER

     E|                .                                                               |
2295 M| 5     IF TURBOJET_POWER = ON THEN                                              | ITERATION_DRIVER

2296 M| 5        T_XDOT  = -TJ_FUEL_FLOW;                                              | ITERATION_DRIVER
     S|                5                                                               |

     E|                .                                                               |
2297 M| 5     IF SCRAMJET_POWER = ON THEN                                              | ITERATION_DRIVER

2298 M| 5        T_XDOT  = -SCRJ_FUEL_FLOW;                                            | ITERATION_DRIVER
     S|                6                                                               |

2299 M| 5     IF I_TIME = OMEGA_I_TIME  THEN                                           | ITERATION_DRIVER
     S|                             3                                                  |

2300 M| 5        DO;                                                                   | ITERATION_DRIVER

     E|                .                                                               |
2301 M| 6           STAGE_SEP = ON;                                                    | ITERATION_DRIVER

     E|                  .                                                             |
2302 M| 6           SCRAMJET_POWER = OFF;                                              | ITERATION_DRIVER

2303 M| 5        END;                                                                  | ITERATION_DRIVER

2304 M| 5     ELSE                                                                     | ITERATION_DRIVER

     E|                .                                                               |
2304 M| 5        SCRAMJET_POWER = ON;                                                  | ITERATION_DRIVER

2305 M| 5     IF I_TIME > OMEGA_I_TIME  THEN                                           | ITERATION_DRIVER
     S|                             2                                                  |

     E|                .                                                               |
2306 M| 5        TURBOJET_POWER = ON;                                                  | ITERATION_DRIVER

2307 M| 5     ELSE                                                                     | ITERATION_DRIVER

     E|                .                                                               |
2307 M| 5        TURBOJET_POWER = OFF;                                                 | ITERATION_DRIVER

2308 M| 5     L_TIME = I_TIME - 1;                                                     | ITERATION_DRIVER

2309 M| 5     T_MASS = NET_MASS       ;                                                | ITERATION_DRIVER
     S|                    I_TIME                                                      |

2310 M| 5     IF I_TIME = OMEGA_I_TIME  THEN                                           | ITERATION_DRIVER
     S|                             3                                                  |
```

```
STMT                                        SOURCE                                             CURRENT SCOPE

2311 M|  5           NET_MASS        = T_MASS - HELD_FS_MASS;                                  | ITERATION_DRIVER
     S|                      I_TIME                                                            |

2312 M|  5        CALL MODEL_DRIVER;                                                           | ITERATION_DRIVER

2313 M|  5        X_DOT  = NET_R_FORCE / NET_MASS       ;                                      | ITERATION_DRIVER
     S|                2                         I_TIME                                        |

2314 M|  5        X_DOT  = NET_THETA_FORCE / (NET_MASS      (X_STORE           +               | ITERATION_DRIVER
     S|                4                          I_TIME         I_TIME:1                      |

2314 M|  5        EARTH_RADIUS));                                                              | ITERATION_DRIVER

     E|                     .                                                                  |
2315 M|  5        IF TURBOJET_POWER = ON THEN                                                  | ITERATION_DRIVER

2316 M|  5           X_DOT  = -TJ_FUEL_FLOW;                                                   | ITERATION_DRIVER
     S|                  5                                                                     |

     E|                     .                                                                  |
2317 M|  5        IF SCRAMJET_POWER = ON THEN                                                  | ITERATION_DRIVER

2318 M|  5           X_DOT  = -SCRJ_FUEL_FLOW;                                                 | ITERATION_DRIVER
     S|                  6                                                                     |

     E|                  .                                                                     |
2319 M|  5        IF STAGE_SEP = ON THEN                                                       | ITERATION_DRIVER

2320 M|  5           X_DOT  = -SS_FUEL_FLOW;                                                   | ITERATION_DRIVER
     S|                  7                                                                     |

     E|             -         -        -                                                       |
2321 M|  5        T_XDOT = T_XDOT - X_DOT;                                                     | ITERATION_DRIVER

2322 M|  5        NET_MASS        = T_MASS;                                                    | ITERATION_DRIVER
     S|                I_TIME                                                                  |

2323 M|  4     END;                                                                            | ITERATION_DRIVER

2324 M|  4  ELSE                                                                               | ITERATION_DRIVER

     E|        -                                                                               |
2324 M|  4     T_XDOT = 0.;                                                                    | ITERATION_DRIVER

     E|                                                        -              -                |
2325 M|  4  SMALL_G_VEC                                 = LAMBDA          . T_XDOT;            | ITERATION_DRIVER
     S|                CONFIG_INDEX+NUM_CONSTANT_PARAMETERS          J_TIME:                   |

     E|     -                  *              -                                                |
2326 M|  4  M_VEC = TRANSPOSE(CAP_LAMBDA_1        ) T_XDOT;                                    | ITERATION_DRIVER
     S|                                   J_TIME:                                              |

2327 M|  4  DO FOR I_LOOP = 1 TO NUM_CONSTRAINTS;                                              | ITERATION_DRIVER
```

```
STMT                                  SOURCE                                              CURRENT SCOPE
2328 M|  5                M                                         = M_VEC     ;        | ITERATION_DRIVER
     S|                    I_LOOP,CONFIG_INDEX+NUM_CONSTANT_PARAMETERS      I_LOOP       |
2329 M|  4             END;                                                              | ITERATION_DRIVER
2330 M|  3          END;                                                                 | ITERATION_DRIVER
2331 M|  3          DO FOR CONFIG_INDEX = 1 TO NUM_CONSTRAINTS;                          | ITERATION_DRIVER
2332 M|  4             DO FOR I_LOOP = 1 TO NUM_CONSTANT_PARAMETERS;                     | ITERATION_DRIVER
2333 M|  5                M                    = -CAP_LAMBDA_2                   ;       | ITERATION_DRIVER
     S|                    CONFIG_INDEX,I_LOOP                I_LOOP,CONFIG_INDEX        |
2334 M|  4             END;                                                              | ITERATION_DRIVER
2335 M|  3          END;                                                                 | ITERATION_DRIVER
2336 M|  2       END;                                                                    | ITERATION_DRIVER
2337 M|  2    ELSE                                                                       | ITERATION_DRIVER
     E|          -       -                                                               |
2337 M|  2       PSI = HOLD_PSI;                                                         | ITERATION_DRIVER
2338 M|  2    CALL DELTA_U_V_CALC;                                                       | ITERATION_DRIVER
2339 M|  2    DO FOR I_STORE = 1 TO NUM_CONSTANT_PARAMETERS;                             | ITERATION_DRIVER
2340 M|  3       P          = P          + DELTA_V          ;                            | ITERATION_DRIVER
     S|           I_STORE      I_STORE            I_STORE                                |
2341 M|  2    END;                                                                       | ITERATION_DRIVER
2342 M|  2    DO FOR I_RCP = 1 TO NUM_TRANS_PTS;                                         | ITERATION_DRIVER
2343 M|  3       SC1 = DELTA_V                              / 4.;                        | ITERATION_DRIVER
     S|                  I_RCP+NUM_CONSTANT_PARAMETERS                                   |
2344 M|  3       IF (((SC1 < POS_TIME_STEP      ) OR (SC1 = POS_TIME_STEP      )) AND ((SC1 > - | ITERATION_DRIVER
     S|                                I_RCP                          I_RCP              |
2344 M|  3       NEG_TIME_STEP      ) OR (SC1 = -NEG_TIME_STEP      ))) THEN              | ITERATION_DRIVER
     S|                    I_RCP                          I_RCP                          |
2345 M|  3          DO;                                                                  | ITERATION_DRIVER
2346 M|  4             POS_TIME_STEP      = POS_TIME_STEP      - SC1;                     | ITERATION_DRIVER
     S|                          I_RCP                I_RCP                              |
2347 M|  4             NEG_TIME_STEP      = NEG_TIME_STEP      + SC1;                     | ITERATION_DRIVER
     S|                          I_RCP                I_RCP                              |
2348 M|  4             DELTA_OMEGA_I_TIME      = 0;                                      | ITERATION_DRIVER
     S|                               I_RCP                                              |
```

```
STMT                                    SOURCE                                                         CURRENT SCOPE

2349 M| 3            END;                                                                             | ITERATION_DRIVER

2350 M| 3         ELSE                                                                                | ITERATION_DRIVER

2350 M| 3            DO;                                                                              | ITERATION_DRIVER

2351 M| 4               IF SC1 < 0. THEN                                                              | ITERATION_DRIVER

2352 M| 4                  SC2 = DELTA_V                               + (4. NEG_TIME_STEP      );    | ITERATION_DRIVER
     S|                             I_RCP+NUM_CONSTANT_PARAMETERS                       I_RCP         |

2353 M| 4               ELSE                                                                          | ITERATION_DRIVER

2353 M| 4                  SC2 = DELTA_V                               - (4. POS_TIME_STEP      );    | ITERATION_DRIVER
     S|                             I_RCP+NUM_CONSTANT_PARAMETERS                       I_RCP         |

2354 M| 4               SC3 = ABS(SC2 / (4. NORM_TIME_STEP));                                         | ITERATION_DRIVER

2355 M| 4               DELTA_OMEGA_I_TIME      = 4 CEILING(SC3) SIGN(-SC1);                          | ITERATION_DRIVER
     S|                                   I_RCP                                                       |

2356 M| 4               SC2 = SC3 - TRUNCATE(SC3);                                                    | ITERATION_DRIVER

2357 M| 4               IF SC1 < 0. THEN                                                              | ITERATION_DRIVER

2358 M| 4                  DO;                                                                        | ITERATION_DRIVER

2359 M| 5                     NEG_TIME_STEP      = (1. - SC2) NORM_TIME_STEP;                         | ITERATION_DRIVER
     S|                                   I_RCP                                                       |

2360 M| 5                     POS_TIME_STEP      = SC2 NORM_TIME_STEP;                                | ITERATION_DRIVER
     S|                                   I_RCP                                                       |

2361 M| 4                  END;                                                                       | ITERATION_DRIVER

2362 M| 4               ELSE                                                                          | ITERATION_DRIVER

2362 M| 4                  DO;                                                                        | ITERATION_DRIVER

2363 M| 5                     NEG_TIME_STEP      = SC2 NORM_TIME_STEP;                                | ITERATION_DRIVER
     S|                                   I_RCP                                                       |

2364 M| 5                     POS_TIME_STEP      = (1. - SC2) NORM_TIME_STEP;                         | ITERATION_DRIVER
     S|                                   I_RCP                                                       |

2365 M| 4                  END;                                                                       | ITERATION_DRIVER

2366 M| 3            END;                                                                             | ITERATION_DRIVER

2367 M| 2         END;                                                                                | ITERATION_DRIVER

2368 M| 2         DO FOR I_STORE = 1 TO NUM_TRANS_PTS;                                                | ITERATION_DRIVER

2369 M| 3            OMEGA_I_TIME        = OMEGA_I_TIME        + DELTA_OMEGA_I_TIME        ;          | ITERATION_DRIVER
     S|                         I_STORE               I_STORE                        I_STORE          |
```

```
STMT                              SOURCE                                        CURRENT SCOPE

2370 M| 3        IF OMEGA_I_TIME        > (FINAL_STEP - 12) THEN               | ITERATION_DRIVER
     S|                     I_STORE                                            |

2371 M| 3           DO;                                                         | ITERATION_DRIVER

2372 M| 4              IF FINAL_STEP > 12 THEN                                  | ITERATION_DRIVER

2373 M| 4                 OMEGA_I_TIME        = FINAL_STEP - 8;                 | ITERATION_DRIVER
     S|                            I_STORE                                      |

2374 M| 4              ELSE                                                     | ITERATION_DRIVER

2374 M| 4                 OMEGA_I_TIME        = 8;                              | ITERATION_DRIVER
     S|                            I_STORE                                      |

2375 M| 4              NEG_TIME_STEP         = NORM_TIME_STEP / 2.;             | ITERATION_DRIVER
     S|                         I_STORE                                         |

2376 M| 4              POS_TIME_STEP         = NORM_TIME_STEP / 2.;             | ITERATION_DRIVER
     S|                         I_STORE                                         |

2377 M| 3           END;                                                        | ITERATION_DRIVER

2378 M| 3        IF OMEGA_I_TIME        < 8 THEN                                | ITERATION_DRIVER
     S|                     I_STORE                                             |

2379 M| 3           DO;                                                         | ITERATION_DRIVER

2380 M| 4              OMEGA_I_TIME        = 8;                                 | ITERATION_DRIVER
     S|                        I_STORE                                          |

2381 M| 4              NEG_TIME_STEP         = NORM_TIME_STEP / 2.;             | ITERATION_DRIVER
     S|                         I_STORE                                         |

2382 M| 4              POS_TIME_STEP         = NORM_TIME_STEP / 2.;             | ITERATION_DRIVER
     S|                         I_STORE                                         |

2383 M| 3           END;                                                        | ITERATION_DRIVER

2384 M| 2        END;                                                           | ITERATION_DRIVER

2385 M| 2        IF OMEGA_I_TIME  < OMEGA_I_TIME  THEN                          | ITERATION_DRIVER
     S|                        2                3                               |

2386 M| 2           DO;                                                         | ITERATION_DRIVER

2387 M| 3              OMEGA_I_TIME  = OMEGA_I_TIME ;                           | ITERATION_DRIVER
     S|                           2                3                            |

2388 M| 3              NEG_TIME_STEP  = NEG_TIME_STEP ;                         | ITERATION_DRIVER
     S|                            2                3                           |

2389 M| 3              POS_TIME_STEP  = POS_TIME_STEP ;                         | ITERATION_DRIVER
     S|                            2                3                           |
```

```
STMT                                    SOURCE                                              CURRENT SCOPE
2390 M| 2           END;                                                                   | ITERATION_DRIVER
2391 M| 2        IF OMEGA_I_TIME  < OMEGA_I_TIME  THEN                                     | ITERATION_DRIVER
     S|                         1               2                                          |
2392 M| 2           DO;                                                                    | ITERATION_DRIVER
2393 M| 3              IF OMEGA_I_TIME  < (FINAL_STEP - 12) THEN                           | ITERATION_DRIVER
     S|                              2                                                     |
2394 M| 3                 DO;                                                              | ITERATION_DRIVER
2395 M| 4                    OMEGA_I_TIME  = OMEGA_I_TIME  + 8;                            | ITERATION_DRIVER
     S|                                 1              2                                   |
2396 M| 4                    NEG_TIME_STEP  = NORM_TIME_STEP / 2.;                         | ITERATION_DRIVER
     S|                                  1                                                 |
2397 M| 4                    POS_TIME_STEP  = NORM_TIME_STEP / 2.;                         | ITERATION_DRIVER
     S|                                  1                                                 |
2398 M| 3                 END;                                                             | ITERATION_DRIVER
2399 M| 3              ELSE                                                                | ITERATION_DRIVER
2399 M| 3                 DO;                                                              | ITERATION_DRIVER
2400 M| 4                    OMEGA_I_TIME  = OMEGA_I_TIME ;                                | ITERATION_DRIVER
     S|                                 1              2                                   |
2401 M| 4                    NEG_TIME_STEP  = NEG_TIME_STEP ;                              | ITERATION_DRIVER
     S|                                  1               2                                 |
2402 M| 4                    POS_TIME_STEP  = POS_TIME_STEP ;                              | ITERATION_DRIVER
     S|                                  1               2                                 |
2403 M| 3                 END;                                                             | ITERATION_DRIVER
2404 M| 2           END;                                                                   | ITERATION_DRIVER
2405 M| 2        IF (OMEGA_I_TIME  - OMEGA_I_TIME ) < 8 THEN                               | ITERATION_DRIVER
     S|                         2               3                                          |
2406 M| 2           DO;                                                                    | ITERATION_DRIVER
2407 M| 3              OMEGA_I_TIME  = OMEGA_I_TIME ;                                      | ITERATION_DRIVER
     S|                           2              3                                         |
2408 M| 3              NEG_TIME_STEP  = NEG_TIME_STEP ;                                    | ITERATION_DRIVER
     S|                            2               3                                       |
2409 M| 3              POS_TIME_STEP  = POS_TIME_STEP ;                                    | ITERATION_DRIVER
     S|                            2               3                                       |
```

```
STMT                                    SOURCE                                        CURRENT SCOPE

2410 M|  3         IF ((ITERATION < 6) AND (FINAL_STEP > (OMEGA_I_TIME  + 12))) THEN    | ITERATION_DRIVER
     S|                                                         2                      |
2411 M|  3            DO;                                                              | ITERATION_DRIVER
2412 M|  4               OMEGA_I_TIME  = OMEGA_I_TIME  + 8;                            | ITERATION_DRIVER
     S|                              2              2                                  |
2413 M|  4               NEG_TIME_STEP  = NORM_TIME_STEP / 2.;                         | ITERATION_DRIVER
     S|                               2                                                |
2414 M|  4               POS_TIME_STEP  = NORM_TIME_STEP / 2.;                         | ITERATION_DRIVER
     S|                               2                                                |
2415 M|  3            END;                                                             | ITERATION_DRIVER
2416 M|  2      END;                                                                   | ITERATION_DRIVER
2417 M|  2   IF (OMEGA_I_TIME  - OMEGA_I_TIME ) < 8 THEN                               | ITERATION_DRIVER
     S|                     1              2                                           |
2418 M|  2      DO;                                                                    | ITERATION_DRIVER
2419 M|  3         OMEGA_I_TIME  = OMEGA_I_TIME ;                                      | ITERATION_DRIVER
     S|                         1              2                                       |
2420 M|  3         NEG_TIME_STEP  = NEG_TIME_STEP ;                                    | ITERATION_DRIVER
     S|                          1               2                                     |
2421 M|  3         POS_TIME_STEP  = POS_TIME_STEP ;                                    | ITERATION_DRIVER
     S|                          1               2                                     |
2422 M|  3         IF ((ITERATION < 9) AND (FINAL_STEP > (OMEGA_I_TIME  + 12))) THEN    | ITERATION_DRIVER
     S|                                                         1                      |
2423 M|  3            DO;                                                              | ITERATION_DRIVER
2424 M|  4               OMEGA_I_TIME  = OMEGA_I_TIME  + 8;                            | ITERATION_DRIVER
     S|                              1              1                                  |
2425 M|  4               NEG_TIME_STEP  = NORM_TIME_STEP / 2.;                         | ITERATION_DRIVER
     S|                               1                                                |
2426 M|  4               POS_TIME_STEP  = NORM_TIME_STEP / 2.;                         | ITERATION_DRIVER
     S|                               1                                                |
2427 M|  3            END;                                                             | ITERATION_DRIVER
2428 M|  2      END;                                                                   | ITERATION_DRIVER
2429 M|  2   SC4 = 0.;                                                                 | ITERATION_DRIVER
2430 M|  2   DO FOR I_U = 2 TO FINAL_STEP;                                             | ITERATION_DRIVER
2431 M|  3      IF I_U < (FINAL_STEP - 3) THEN                                         | ITERATION_DRIVER
```

```
STMT                              SOURCE                                              CURRENT SCOPE
2432 M| 3          DO;                                                                | ITERATION_DRIVER
2433 M| 4             I_TIME = I_U;                                                   | ITERATION_DRIVER
2434 M| 4             CALL TIME_SET;                                                  | ITERATION_DRIVER
2435 M| 4             SC4 = SC4 + PRESENT_TIME_STEP;                                  | ITERATION_DRIVER
2436 M| 3          END;                                                               | ITERATION_DRIVER
2437 M| 3       ELSE                                                                  | ITERATION_DRIVER
2437 M| 3          SC4 = SC4 + FINAL_TIME_STEP;                                       | ITERATION_DRIVER
2438 M| 3       U_NEW_TIME    = SC4;                                                  | ITERATION_DRIVER
     S|                 I_U                                                           |
2439 M| 2    END;                                                                     | ITERATION_DRIVER
2440 M| 2    IF OLD_FINAL_STEP > FINAL_STEP THEN                                      | ITERATION_DRIVER
2441 M| 2       OLD_FINAL_STEP = FINAL_STEP;                                          | ITERATION_DRIVER
2442 M| 2    IF OLD_FINAL_STEP > 9 THEN                                               | ITERATION_DRIVER
2443 M| 2       DO FOR I_U = 5 TO (OLD_FINAL_STEP - 4);                               | ITERATION_DRIVER
2444 M| 3          IF MOD(I_U, 2) = 1 THEN                                            | ITERATION_DRIVER
2445 M| 3             DO;                                                             | ITERATION_DRIVER
2446 M| 4                J_U = CEILING(I_U / 2);                                      | ITERATION_DRIVER
2447 M| 4                LI4 = 4 NUM_TRANS_PTS;                                       | ITERATION_DRIVER
2448 M| 4                SC4 = (LI4 + .1) NORM_TIME_STEP;                             | ITERATION_DRIVER
2449 M| 4                IF (I_U + LI4) < OLD_FINAL_STEP THEN                         | ITERATION_DRIVER
2450 M| 4                   LI5 = LI4;                                                | ITERATION_DRIVER
2451 M| 4                ELSE                                                         | ITERATION_DRIVER
2451 M| 4                   LI5 = OLD_FINAL_STEP - 1 - I_U;                           | ITERATION_DRIVER
2452 M| 4                IF (I_U - LI4) < 2 THEN                                      | ITERATION_DRIVER
2453 M| 4                   LI4 = I_U - 2;                                            | ITERATION_DRIVER
2454 M| 4                OLD_SC3 = SC4;                                               | ITERATION_DRIVER
     E|                   .                                                           |
2455 M| 4                LI_FLAG = ON;                                                | ITERATION_DRIVER
```

```
STMT                          SOURCE                                             CURRENT SCOPE
     E|                                            .                            |
2456 M| 4       DO FOR I_RCP = (I_U - LI4) TO (I_U + LI5) WHILE LI_FLAG = ON;     | ITERATION_DRIVER
2457 M| 5          IF MOD(I_RCP, 2) = 1 THEN                                      | ITERATION_DRIVER
2458 M| 5             DO;                                                         | ITERATION_DRIVER
2459 M| 6                J_RCP = CEILING(I_RCP / 2);                              | ITERATION_DRIVER
2460 M| 6                SC3 = ABS(U_NEW_TIME    - U_OLD_TIME      );             | ITERATION_DRIVER
     S|                                    I_U              I_RCP                |
2461 M| 6                IF SC3 > OLD_SC3 THEN                                    | ITERATION_DRIVER
     E|                   .                                                      |
2462 M| 6                   LI_FLAG = OFF;                                        | ITERATION_DRIVER
2463 M| 6                IF SC3 < SC4 THEN                                        | ITERATION_DRIVER
2464 M| 6                   DO;                                                   | ITERATION_DRIVER
2465 M| 7                      UV1 = U_NEW_TIME    - U_OLD_TIME      ;           | ITERATION_DRIVER
     S|                                      I_U              I_RCP              |
2466 M| 7                      IF U_NEW_TIME    > U_OLD_TIME      THEN           | ITERATION_DRIVER
     S|                                   I_U              I_RCP                 |
2467 M| 7                         DO;                                            | ITERATION_DRIVER
2468 M| 8                            UV1 = UV1 / (U_OLD_TIME        - U_OLD_TIME      );  | ITERATION_DRIVER
     S|                                                 I_RCP+1          I_RCP   |
     E|                                 -                  -               -      |
2469 M| 8                            DELTA_U_NEW      = DELTA_U        + ((DELTA_U        | ITERATION_DRIVER
     S|                                         J_U:             J_RCP:          J_RCP+1: |
     E|                                 -                                         |
2469 M| 8                            - DELTA_U        ) UV1);                     | ITERATION_DRIVER
     S|                                       J_RCP:                              |
     E|                            -              -                   -           |
2470 M| 8                            U_NEW     = U_ACTIVE               + ((U_ACTIVE       | ITERATION_DRIVER
     S|                                  J_U:             (2 J_RCP)-1:         (2  |
     E|                                           -                               |
2470 M| 8                                      - U_ACTIVE              ) UV1);    | ITERATION_DRIVER
     S|                            J_RCP)+1:               (2 J_RCP)-1:          |
2471 M| 7                         END;                                           | ITERATION_DRIVER
2472 M| 7                      ELSE                                              | ITERATION_DRIVER
2472 M| 7                         DO;                                            | ITERATION_DRIVER
```

```
STMT                                  SOURCE                                              CURRENT SCOPE

2473 M|  8                            UV1 = UV1 / (U_OLD_TIME       - U_OLD_TIME         );  | ITERATION_DRIVER
     S|                                                      I_RCP                I_RCP-1      |

     E|                                 -                   -                   -              |
2474 M|  8                            DELTA_U_NEW     = DELTA_U       + ((DELTA_U        -  | ITERATION_DRIVER
     S|                                          J_U:              J_RCP:              J_RCP:  |

     E|                                 -                                                      |
2474 M|  8                            DELTA_U          ) UV1);                                 | ITERATION_DRIVER
     S|                                      J_RCP-1:                                          |

     E|                                 -            -                     -                   |
2475 M|  8                            U_NEW      = U_ACTIVE             + ((U_ACTIVE           | ITERATION_DRIVER
     S|                                     J_U:             (2 J_RCP)-1:              (2      |

     E|                                              -                                         |
2475 M|  8                                      - U_ACTIVE             ) UV1);                 | ITERATION_DRIVER
     S|                               J_RCP)-1:              (2 J_RCP)-3:                      |

2476 M|  7                         END;                                                        | ITERATION_DRIVER
2477 M|  7                      SC4 = SC3;                                                     | ITERATION_DRIVER
2478 M|  6                   END;                                                              | ITERATION_DRIVER
2479 M|  6                 OLD_SC3 = SC3;                                                      | ITERATION_DRIVER
2480 M|  5              END;                                                                   | ITERATION_DRIVER
2481 M|  4           END;                                                                      | ITERATION_DRIVER
2482 M|  3         END;                                                                        | ITERATION_DRIVER
2483 M|  2       END;                                                                          | ITERATION_DRIVER
2484 M|  2     DO FOR J_U = 4 TO FLOOR((OLD_FINAL_STEP - 5) / 2);                              | ITERATION_DRIVER

     E|          -               -                                                             |
2485 M|  3       DELTA_U     = DELTA_U_NEW    ;                                                | ITERATION_DRIVER
     S|                 J_U:             J_U:                                                  |

     E|          -                     -                                                       |
2486 M|  3       U_ACTIVE           = U_NEW    ;                                               | ITERATION_DRIVER
     S|                  (2 J_U)-1:        J_U:                                                |

2487 M|  2     END;                                                                            | ITERATION_DRIVER
2488 M|  2     DO FOR I_U = 1 TO (FINAL_STEP - 2);                                             | ITERATION_DRIVER
2489 M|  3       IF MOD(I_U, 2) = 1 THEN                                                       | ITERATION_DRIVER

     E|            -            -            -                                                 |
2490 M|  3         U_ACTIVE     = U_ACTIVE     + DELTA_U                 ;                     | ITERATION_DRIVER
     S|                    I_U:           I_U:          CEILING(I_U/2):                        |
```

```
STMT                                   SOURCE                                                    CURRENT SCOPE
2491 M| 2          END;                                                                       | ITERATION_DRIVER
2492 M| 2          DO FOR I_U = 1 TO (FINAL_STEP - 3);                                        | ITERATION_DRIVER
2493 M| 3             IF MOD(I_U, 2) ¬= 1 THEN                                                | ITERATION_DRIVER
2494 M| 3                DO;                                                                  | ITERATION_DRIVER
2495 M| 4                   IF (((I_U + 1) = OMEGA_I_TIME ) OR ((I_U + 1) = OMEGA_I_TIME ) OR ((I_U + 1 | ITERATION_DRIVER
     S|                                                 1                              2      |
2495 M| 4                   ) = OMEGA_I_TIME )) THEN                                          | ITERATION_DRIVER
     S|                                   3                                                   |
     E|                        -              -                                               |
2496 M| 4                      U_ACTIVE     = U_ACTIVE      ;                                 | ITERATION_DRIVER
     S|                               I_U:           I_U-1:                                   |
2497 M| 4                   ELSE                                                              | ITERATION_DRIVER
     E|                        -               -                 -                            |
2497 M| 4                      U_ACTIVE     = (U_ACTIVE       + U_ACTIVE       ) / 2.;        | ITERATION_DRIVER
     S|                               I_U:             I_U+1:           I_U-1:                |
2498 M| 3                END;                                                                 | ITERATION_DRIVER
2499 M| 2          END;                                                                       | ITERATION_DRIVER
     E|            -        -                                                                 |
2500 M| 2          U_KEEP = U_ACTIVE             ;                                            | ITERATION_DRIVER
     S|                           FINAL_STEP-2:                                               |
2501 M| 2          DO FOR I_U = (FINAL_STEP - 3) TO (STEP_DIM + 1);                           | ITERATION_DRIVER
2502 M| 3             IF I_U > FINAL_STEP THEN                                                | ITERATION_DRIVER
     E|                -             -                    -                   -               |
2503 M| 3                U_ACTIVE     = U_ACTIVE            - (((U_ACTIVE            - U_ACTIVE | ITERATION_DRIVER
     S|                         I_U:           FINAL_STEP:              FINAL_STEP-4:         |
2503 M| 3                            ) (I_U - FINAL_STEP)) / 2.);                             | ITERATION_DRIVER
     S|                FINAL_STEP-2:                                                          |
2504 M| 3             ELSE                                                                    | ITERATION_DRIVER
     E|                -             -                      -                  -              |
2504 M| 3                U_ACTIVE     = U_ACTIVE              - (((U_ACTIVE             - U_KEEP) (I_U | ITERATION_DRIVER
     S|                         I_U:           FINAL_STEP-4:              FINAL_STEP-4:       |
2504 M| 3                + 4 - FINAL_STEP)) (NORM_TIME_STEP / (2. FINAL_TIME_STEP)));         | ITERATION_DRIVER
2505 M| 2          END;                                                                       | ITERATION_DRIVER
2506 M| 2          CALL STATE_INTEGRATION;                                                    | ITERATION_DRIVER
```

```
STMT                              SOURCE                                                       CURRENT SCOPE

     E|                  -          *       -        -            *        -                   |
2507 M| 2        IF (((PSI . (PSI_WEIGHT PSI)) < (HOLD_PSI . (PSI_WEIGHT HOLD_PSI))) OR ((PSI_MAG < (  | ITERATION_DRIVER

2507 M| 2        MIN_PSI_THRESHOLD FIRST_PSI_MAG)) AND ((X_STORE             + X_STORE             +   | ITERATION_DRIVER
     S|                                                  FINAL_STEP:5          FINAL_STEP:6        |

2507 M| 2        X_STORE            - INTEG_L) < COST))) THEN                                      | ITERATION_DRIVER
     S|                 FINAL_STEP:7                                                               |

     E|                .                                                                           |
2508 M| 2           STEP_I_FLAG = OFF;                                                             | ITERATION_DRIVER

     E|                     .                                                                      |
2509 M| 2        ELSE IF ((STEP_I_FLAG = OFF) OR (MAX_STEP_I ¬= STEP_I)) THEN                      | ITERATION_DRIVER

2510 M| 2           DO;                                                                            | ITERATION_DRIVER

     E|                   -          -                                                             |
2511 M| 3              [X_STORE] = [HOLD_X];                                                       | ITERATION_DRIVER

2512 M| 3              [NET_MASS] = [HOLD_M];                                                      | ITERATION_DRIVER

     E|                    -           -                                                           |
2513 M| 3              [L_X_STORE] = [HOLD_L_X];                                                   | ITERATION_DRIVER

     E|               -     -                                                                      |
2514 M| 3              P = HOLD_P;                                                                 | ITERATION_DRIVER

     E|                   -           -                                                            |
2515 M| 3              [U_ACTIVE] = [HOLD_U];                                                      | ITERATION_DRIVER

2516 M| 3              [U_OLD_TIME] = [HOLD_U_T];                                                  | ITERATION_DRIVER

2517 M| 3              [OMEGA_I_TIME] = [HOLD_T];                                                  | ITERATION_DRIVER

2518 M| 3              [POS_TIME_STEP] = [HOLD_POS_TIME];                                          | ITERATION_DRIVER

2519 M| 3              [NEG_TIME_STEP] = [HOLD_NEG_TIME];                                          | ITERATION_DRIVER

2520 M| 2           END;                                                                           | ITERATION_DRIVER

2521 M| 1        END;                                                                              | ITERATION_DRIVER

2522 M|     END;                                                                                  | ITERATION_DRIVER

2523 M| CLOSE ITERATION_DRIVER;                                                                    | ITERATION_DRIVER

**** B L O C K   S U M M A R Y ****

OUTER PROCEDURES CALLED
    TIME_SET, MODEL_DRIVER, RUNGE_KUTTA, HAM_SUB_U, DELTA_U_V_CALC

COMPOOL VARIABLES USED
    NUM_STATES, NUM_CONSTRAINTS, STEP_DIM, NUM_CONTROLS, FINAL_STEP, NORM_TIME_STEP, U_OLD_TIME*, ITERATION, U_TIME_KEEP*
```

```
STMT                                   SOURCE                                   CURRENT SCOPE

   U_OLD_TIME, FIRST_ITERATION_FLAG, FIRST_ITERATION_FLAG*, MAX_STEP_I, STEP_SCALE_J*, J_SCALE_FACTOR, STEP_SCALE_PSI*
   PSI_SCALE_FACTOR, X_STORE, NUM_CONSTANT_PARAMETERS, EARTH_RADIUS, P, NOZZLE_ANGLE, MIN_SCRJ_MASS_PER_FT, CHECK_I_TIME
   NUM_TRANS_PTS, P*, POS_TIME_STEP, NEG_TIME_STEP, POS_TIME_STEP*, NEG_TIME_STEP*, OMEGA_I_TIME*, U_ACTIVE, U_ACTIVE*, PSI_WEIGHT
   MIN_PSI_THRESHOLD, FIRST_PSI_MAG, X_STORE*

OUTER VARIABLES USED
   I_TIME*, PRESENT_TIME_STEP, FINAL_TIME_STEP, HOLD_U_T*, OVER_STEP, STEP_I*, STEP_I, PSI, J_TIME*, M1_FLOW_FINAL_STEP*
   M2_FLOW_FINAL_STEP*, M3_FLOW_FINAL_STEP*, MASS_FLOW_FINAL_STEP*, NET_MASS, L_TIME*, STAGE_SEP*, TURBOJET_POWER*, SCRAMJET_POWER*
   SMALL_G_VEC*, I_J_J*, I_PSI_J*, I_PSI_PSI*, NET_THETA_FORCE, J_TIME, LAMBDA*, MASS_FLOW_FINAL_STEP, L_FINAL, CAP_LAMBDA_1*
   NET_R_FORCE, M1_FLOW_FINAL_STEP, M2_FLOW_FINAL_STEP, M3_FLOW_FINAL_STEP, CAP_LAMBDA_2*, DM_DP*, SCRAMJET_MASS, I_TIME_STORE*
   TIME_STEP*, RK_COLUMNS*, I_TIME, I_TIME_STORE, LAMBDA_FLAG*, RK_ROWS*, FD_FLAG*, I_CL*, I_STORE*, I_STORE, RK_VAL_N_PLUS_1
   CAP_LAMBDA_1_FLAG*, J_STORE*, J_STORE, CAP_LAMBDA_2_FLAG*, SGV_FLAG*, I_J_J_FLAG*, I_PSI_J_FLAG*, I_PSI_PSI_FLAG*, I_PSI_PSI
   CONFIG_INDEX*, CONFIG_INDEX, X_DOT*, STATE_INTEGRATION_FLAG*, TURBOJET_POWER, TJ_FUEL_FLOW, SCRAMJET_POWER, SCRJ_FUEL_FLOW
   NET_MASS*, HELD_FS_MASS, STAGE_SEP, SS_FUEL_FLOW, X_DOT, LAMBDA, CAP_LAMBDA_1, I_LOOP*, I_LOOP, M*, CAP_LAMBDA_2, PSI*, DELTA_V
   DELTA_OMEGA_I_TIME*, DELTA_OMEGA_I_TIME, U_NEW_TIME*, U_NEW_TIME, DELTA_U, DELTA_U*, PSI_MAG, INTEG_L, COST, HOLD_X, HOLD_M
   L_X_STORE*, HOLD_L_X, HOLD_P, HOLD_U, HOLD_U_T, HOLD_T, HOLD_POS_TIME, HOLD_NEG_TIME
```

```
STMT                                   SOURCE                                                               CURRENT SCOPE
2524 M| OPTIMIZATION:                                                                                      | OPTIMIZATION
2524 M| PROCEDURE;                                                                                         | OPTIMIZATION
2525 M|    DECLARE ANGLE_OF_ATTACK_INITIAL ARRAY(STEP_DIM + 1) SCALAR DOUBLE AUTOMATIC;                    | OPTIMIZATION
2526 M|    DECLARE CAP_PHI_INITIAL ARRAY(STEP_DIM + 1) SCALAR DOUBLE AUTOMATIC;                            | OPTIMIZATION
2527 M|    DECLARE CP_INDEX ARRAY(23) INTEGER STATIC INITIAL(1, 250, 266, 352, 394, 538, 802, 913, 917, 921 | OPTIMIZATION
2527 M|    , 985, 989, 993, 1006, 1026, 1177, 1181, 1185, 1218, 1230, 1254, 1278, 2001);                   | OPTIMIZATION
2528 M|    DECLARE CP_INIT ARRAY(23) SCALAR DOUBLE STATIC INITIAL(.035, .035, .170, .250, .669, .9, .97,   | OPTIMIZATION
2528 M|    .97, .97, .97, .97, .97, .97, .97, .9, .9, .9, .9, .9, .9, .81, .9, .9);                         | OPTIMIZATION
2529 M|    DECLARE MAX_CP_INDEX INTEGER STATIC INITIAL(23);                                                | OPTIMIZATION
2530 M|    DECLARE W_INDEX ARRAY(5) INTEGER STATIC INITIAL(1, 301, 351, 401, 1001);                        | OPTIMIZATION
2531 M|    DECLARE W_INIT ARRAY(5, NUM_CONTROLS) SCALAR DOUBLE STATIC INITIAL(5., 1000., 5., 1000., 5.,    | OPTIMIZATION
2531 M|    1000., 5., 1000., 5., 1000.);                                                                   | OPTIMIZATION
2532 M|    DECLARE MAX_W_INDEX INTEGER STATIC INITIAL(5);                                                  | OPTIMIZATION
2533 M|    DECLARE AA_INDEX ARRAY(31) INTEGER STATIC INITIAL(1, 106, 210, 326, 458, 638, 678, 874, 894, 913 | OPTIMIZATION
2533 M|    , 914, 917, 921, 934, 954, 985, 989, 993, 1006, 1036, 1106, 1150, 1170, 1177, 1181, 1185, 1250,  | OPTIMIZATION
2533 M|    1258, 1270, 1278, 2001);                                                                        | OPTIMIZATION
2534 M|    DECLARE AA_INIT ARRAY(31) SCALAR DOUBLE STATIC INITIAL(0., 0., -.120, .020, .090, .210, .215,   | OPTIMIZATION
2534 M|    .210, .180, .1496, .148, .14425, .14425, .128, .022, .022, .022, .022, .022, .029, .029, .002,  | OPTIMIZATION
2534 M|    .002, .00421, .00453, .00547, .026, .035, .070, .097, .097);                                    | OPTIMIZATION
2535 M|    DECLARE MAX_AA_INDEX INTEGER STATIC INITIAL(31);                                                | OPTIMIZATION
2536 M|    DECLARE P_INITIAL VECTOR(NUM_CONSTANT_PARAMETERS) DOUBLE STATIC INITIAL(.021, 32.157, 9.4, 6.49, | OPTIMIZATION
2536 M|    268.19, 1.222, .416, 161., 33450., 1.8708E06);                                                  | OPTIMIZATION
2537 M|    DECLARE OIT_INIT ARRAY(NUM_TRANS_PTS) INTEGER STATIC INITIAL(1181, 989, 917);                   | OPTIMIZATION
2538 M|    DECLARE FUEL_COST SCALAR DOUBLE AUTOMATIC;                                                      | OPTIMIZATION
     E|     *                                                                                              |
2539 M|    [W] = 0.;                                                                                       | OPTIMIZATION
2540 M|    ITERATION = 1;                                                                                  | OPTIMIZATION
2541 M|    DO FOR I_STORE = 1 TO FLOOR((STEP_DIM + 3) / 2);                                                | OPTIMIZATION
```

```
STMT                                       SOURCE                                              CURRENT SCOPE

2542 M| 1    IF ((I_STORE ¬< W_INDEX          ) AND (ITERATION < MAX_W_INDEX)) THEN          | OPTIMIZATION
     S|                         ITERATION                                                    |

2543 M| 1       ITERATION = ITERATION + 1;                                                   | OPTIMIZATION

2544 M| 1    DO FOR IIT = 1 TO NUM_CONTROLS;                                                 | OPTIMIZATION

2545 M| 2       W              = W_INIT              + ((W_INIT           - W_INIT          ) | OPTIMIZATION
     S|          I_STORE:IIT,IIT      ITERATION-1,IIT       ITERATION,IIT     ITERATION-1,IIT  |

2545 M| 2       ((I_STORE - W_INDEX           ) / (W_INDEX          - W_INDEX          )));  | OPTIMIZATION
     S|                        ITERATION-1           ITERATION          ITERATION-1            |

2546 M| 1    END;                                                                            | OPTIMIZATION

2547 M|   END;                                                                               | OPTIMIZATION

2548 M|   ITERATION = 1;                                                                     | OPTIMIZATION

2549 M|   DO FOR I_STORE = 1 TO (STEP_DIM + 1);                                              | OPTIMIZATION

2550 M| 1    IF ((I_STORE ¬< AA_INDEX          ) AND (ITERATION < MAX_AA_INDEX)) THEN        | OPTIMIZATION
     S|                         ITERATION                                                    |

2551 M| 1       ITERATION = ITERATION + 1;                                                   | OPTIMIZATION

2552 M| 1    ANGLE_OF_ATTACK_INITIAL        = AA_INIT             + ((AA_INIT          - AA_INIT     | OPTIMIZATION
     S|                         I_STORE        ITERATION-1            ITERATION         ITERATION-1 |

2552 M| 1    ) ((I_STORE - AA_INDEX           ) / (AA_INDEX          - AA_INDEX          )));  | OPTIMIZATION
     S|                        ITERATION-1           ITERATION          ITERATION-1            |

2553 M|   END;                                                                               | OPTIMIZATION

2554 M|   ITERATION = 1;                                                                     | OPTIMIZATION

2555 M|   DO FOR I_STORE = 1 TO (STEP_DIM + 1);                                              | OPTIMIZATION

2556 M| 1    IF ((I_STORE ¬< CP_INDEX          ) AND (ITERATION < MAX_CP_INDEX)) THEN        | OPTIMIZATION
     S|                         ITERATION                                                    |

2557 M| 1       ITERATION = ITERATION + 1;                                                   | OPTIMIZATION

2558 M| 1    CAP_PHI_INITIAL        = CP_INIT             + ((CP_INIT          - CP_INIT          ) (( | OPTIMIZATION
     S|                  I_STORE        ITERATION-1            ITERATION         ITERATION-1          |

2558 M| 1    I_STORE - CP_INDEX           ) / (CP_INDEX          - CP_INDEX          )));      | OPTIMIZATION
     S|                 ITERATION-1           ITERATION          ITERATION-1                 |

2559 M|   END;                                                                               | OPTIMIZATION

     E|   -    -                                                                             |
2560 M|   P = P_INITIAL;                                                                     | OPTIMIZATION

2561 M|   ITERATION = 1;                                                                     | OPTIMIZATION
```

```
STMT                                    SOURCE                                              CURRENT SCOPE

2562 M|     DO FOR I_STORE = 1 TO (STEP_DIM + 1);                                           | OPTIMIZATION

2563 M| 1      U_ACTIVE          = ANGLE_OF_ATTACK_INITIAL        ;                         | OPTIMIZATION
     S|                I_STORE:1                         I_STORE                             |

2564 M| 1      IF CAP_PHI_INITIAL        < 1. THEN                                          | OPTIMIZATION
     S|                          I_STORE                                                     |

2565 M| 1         U_ACTIVE          = SQRT((1. / CAP_PHI_INITIAL        ) - 1.);            | OPTIMIZATION
     S|                  I_STORE:2                             I_STORE                       |

2566 M| 1      ELSE                                                                         | OPTIMIZATION

2566 M| 1         U_ACTIVE          = 0.;                                                   | OPTIMIZATION
     S|                  I_STORE:2                                                           |

2567 M|     END;                                                                            | OPTIMIZATION

2568 M|     [OMEGA_I_TIME] = [OIT_INIT];                                                    | OPTIMIZATION

     E|        -                                                                             |
2569 M|     [X_STORE] = 0.;                                                                 | OPTIMIZATION

2570 M|     [NET_MASS] = 0.;                                                                | OPTIMIZATION

2571 M|     DJS = DJS_INIT;                                                                 | OPTIMIZATION

     E|       -      -                                                                       |
2572 M|     HOLD_P = P;                                                                     | OPTIMIZATION

2573 M|     CALL ITERATION_DRIVER;                                                          | OPTIMIZATION

2574 M|     FIRST_PASS_PSI_MAG = PSI_MAG;                                                   | OPTIMIZATION

     E|             .                                                                        |
2575 M|     IF OVER_STEP = OFF THEN                                                         | OPTIMIZATION

2576 M|        DO;                                                                          | OPTIMIZATION

2577 M| 1         FUEL_COST = X_STORE             + X_STORE             + X_STORE             ;  | OPTIMIZATION
     S|                          FINAL_STEP:5           FINAL_STEP:6           FINAL_STEP:7  |

2578 M| 1         COST = FUEL_COST - INTEG_L;                                               | OPTIMIZATION

     E|                .                                                                     |
2579 M| 1         ITER_FLAG = ON;                                                           | OPTIMIZATION

     E|                .                                                                     |
2580 M| 1         OVER_ITER_FLAG = OFF;                                                     | OPTIMIZATION

2581 M| 1         L_FILE = ITERATION;                                                       | OPTIMIZATION

     E|                                                       .                 .            |
2582 M| 1         DO FOR IIT = L_FILE TO MAX_ITERATIONS WHILE (ITER_FLAG AND NOT OVER_ITER_FLAG) = ON;  | OPTIMIZATION
```

```
STMT                                  SOURCE                                                  CURRENT SCOPE
2583 M| 2        IF IIT = MAX_ITERATIONS THEN                                               | OPTIMIZATION
     E|                                                                                     |
2584 M| 2           OVER_ITER_FLAG = ON;                                                    | OPTIMIZATION
2585 M| 2        ELSE                                                                       | OPTIMIZATION
2585 M| 2           DO;                                                                     | OPTIMIZATION
     E|                                                                                     |
2586 M| 3              [HOLD_X] = [X_STORE];                                                | OPTIMIZATION
2587 M| 3              [HOLD_M] = [NET_MASS];                                               | OPTIMIZATION
     E|                                                                                     |
2588 M| 3              [HOLD_L_X] = [L_X_STORE];                                            | OPTIMIZATION
     E|                                                                                     |
2589 M| 3              HOLD_P = P;                                                          | OPTIMIZATION
2590 M| 3              ITERATION = ITERATION + 1;                                           | OPTIMIZATION
2591 M| 3              [HOLD_T] = [OMEGA_I_TIME];                                           | OPTIMIZATION
     E|                                                                                     |
2592 M| 3              [HOLD_U] = [U_ACTIVE];                                               | OPTIMIZATION
2593 M| 3              [HOLD_POS_TIME] = [POS_TIME_STEP];                                   | OPTIMIZATION
2594 M| 3              [HOLD_NEG_TIME] = [NEG_TIME_STEP];                                   | OPTIMIZATION
2595 M| 3              CALL ITERATION_DRIVER;                                               | OPTIMIZATION
2596 M| 3              FIRST_PASS_PSI_MAG = PSI_MAG;                                        | OPTIMIZATION
     E|                                                                                     |
2597 M| 3              IF OVER_STEP = OFF THEN                                              | OPTIMIZATION
2598 M| 3                 DO;                                                               | OPTIMIZATION
2599 M| 4                    FUEL_COST = X_STORE             + X_STORE             + X_STORE | OPTIMIZATION
     S|                               FINAL_STEP:5          FINAL_STEP:6          FINAL_STEP:7 |
2599 M| 4                    ;                                                              | OPTIMIZATION
2600 M| 4                    DELTA_COST = COST - FUEL_COST + INTEG_L;                       | OPTIMIZATION
2601 M| 4                    COST = FUEL_COST - INTEG_L;                                    | OPTIMIZATION
2602 M| 4                    IF ((ABS(DELTA_COST) < MIN_COST_RQMT) AND (PSI_MAG < MIN_PSI_RQMT)) THEN | OPTIMIZATION
     E|                                                                                     |
2603 M| 4                       ITER_FLAG = OFF;                                            | OPTIMIZATION
2604 M| 4                    ELSE IF ABS(PSI_MAG / FIRST_PSI_MAG) < PSI_CHECK THEN          | OPTIMIZATION
```

```
STMT                    SOURCE                                              CURRENT SCOPE
2605 M| 4              DO;                                                  | OPTIMIZATION
2606 M| 5                 IF DELTA_COST > 0. THEN                           | OPTIMIZATION
2607 M| 5                    DO;                                            | OPTIMIZATION
2608 M| 6                       IF DJS < (DELTA_COST_CHECK DELTA_COST) THEN | OPTIMIZATION
2609 M| 6                          DJS = DELTA_COST_SHRINK DJS;             | OPTIMIZATION
2610 M| 6                       IF DJS > -DELTA_COST THEN                   | OPTIMIZATION
2611 M| 6                          DJS = -DELTA_COST;                       | OPTIMIZATION
2612 M| 6                       IF DJS < (2. DJS_INIT) THEN                 | OPTIMIZATION
2613 M| 6                          DJS = 2. DJS_INIT;                       | OPTIMIZATION
2614 M| 5                    END;                                           | OPTIMIZATION
2615 M| 5                 ELSE                                              | OPTIMIZATION
     E|                          .                                          |
2615 M| 5                    ITER_FLAG = OFF;                               | OPTIMIZATION
2616 M| 4              END;                                                 | OPTIMIZATION
2617 M| 3           END;                                                    | OPTIMIZATION
2618 M| 2        END;                                                       | OPTIMIZATION
2619 M| 1     END;                                                          | OPTIMIZATION
2620 M|    END;                                                             | OPTIMIZATION
2621 M| CLOSE OPTIMIZATION;                                                  | OPTIMIZATION

**** B L O C K   S U M M A R Y ****

OUTER PROCEDURES CALLED
    ITERATION_DRIVER

COMPOOL VARIABLES USED
    STEP_DIM, NUM_CONTROLS, NUM_CONSTANT_PARAMETERS, NUM_TRANS_PTS, W*, ITERATION*, ITERATION, P*, U_ACTIVE*, OMEGA_I_TIME*
    X_STORE*, DJS_INIT, P, FINAL_STEP, X_STORE, L_FILE*, L_FILE, MAX_ITERATIONS, OMEGA_I_TIME, U_ACTIVE, POS_TIME_STEP
    NEG_TIME_STEP, MIN_COST_RQMT, MIN_PSI_RQMT, FIRST_FSI_MAG, PSI_CHECK, DELTA_COST_CHECK, DELTA_COST_SHRINK

OUTER VARIABLES USED
    I_STORE*, I_STORE, IIT*, IIT, NET_MASS*, DJS*, HOLD_P*, FIRST_PASS_PSI_MAG*, FSI_MAG, OVER_STEP, COST*, INTEG_L, ITER_FLAG*
    OVER_ITER_FLAG*, ITER_FLAG, OVER_ITER_FLAG, HOLD_X*, HOLD_M*, NET_MASS, HOLD_L_X*, L_X_STORE, HOLD_T*, HOLD_U*, HOLD_POS_TIME*
    HOLD_NEG_TIME*, DELTA_COST*, COST, DELTA_COST, DJS
```

```
STMT                              SOURCE                                      CURRENT SCOPE

2622 M|    FINAL_STEP = STEP_DIM + 1;                                         | THESIS_ALGORITHM
2623 M|    STEP_I = 1;                                                        | THESIS_ALGORITHM
2624 M|    [NET_MASS] = 0.;                                                   | THESIS_ALGORITHM
     E|       -                                                               |
2625 M|    DELTA_V = 0.;                                                      | THESIS_ALGORITHM
     E|       *                                                               |
2626 M|    [G_MAT] = 0.;                                                      | THESIS_ALGORITHM
2627 M|    [U_J_OLD_TIME] = 0.;                                               | THESIS_ALGORITHM
2628 M|    [U_OLD_TIME] = 0.;                                                 | THESIS_ALGORITHM
2629 M|    [U_NEW_TIME] = 0.;                                                 | THESIS_ALGORITHM
2630 M|    FINAL_TIME_STEP = NORM_TIME_STEP;                                  | THESIS_ALGORITHM
     E|        .                                                              |
2631 M|    OVER_STEP = OFF;                                                   | THESIS_ALGORITHM
2632 M|    FD_COUNT = 0;                                                      | THESIS_ALGORITHM
2633 M|    PRESENT_TIME_STEP = NORM_TIME_STEP;                                | THESIS_ALGORITHM
     E|    *                                                                  |
2634 M|    U = 0.;                                                            | THESIS_ALGORITHM
2635 M|    [U_TIME_KEEP] = 0.;                                                | THESIS_ALGORITHM
2636 M|    I_FD = 5;                                                          | THESIS_ALGORITHM
     E|              .                                                        |
2637 M|    FIRST_DERIV_FLAG = OFF;                                            | THESIS_ALGORITHM
     E|                .                                                      |
2638 M|    PARTIAL_DERIV_FLAG = OFF;                                          | THESIS_ALGORITHM
     E|      -                                                                |
2639 M|    DP_DUA = 0.;                                                       | THESIS_ALGORITHM
     E|      -                                                                |
2640 M|    DN_DUA = 0.;                                                       | THESIS_ALGORITHM
2641 M|    CALL OPTIMIZATION;                                                 | THESIS_ALGORITHM
2642 M| CLOSE THESIS_ALGORITHM;                                               | THESIS_ALGORITHM

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
   NUM_CONTROLS, NUM_CONSTRAINTS, STEP_DIM, NUM_STATES, NUM_CONSTANT_PARAMETERS, NUM_TRANS_PTS, FINAL_STEP*, U_OLD_TIME*
```

STMT SOURCE CURRENT SCOPE

```
NORM_TIME_STEP, U_TIME_KEEP*
```

```
STMT                              SOURCE                                      CURRENT SCOPE

     E|             .                                                        |
2643 M|     FIRST_ITERATION_FLAG = ON;                                       | AIR_BREATHER_OPTIMIZAT
2644 M|     FIRST_PSI_MAG = 0.;                                              | AIR_BREATHER_OPTIMIZAT
     E|        -                                                             |
2645 M|     [X_STORE] = 0.;                                                  | AIR_BREATHER_OPTIMIZAT
     E|               .                                                      |
2646 M|     STATE_INTEGRATION_FLAG = OFF;                                    | AIR_BREATHER_OPTIMIZAT
     E|             .                                                        |
2647 M|     OBLIQUE_SHOCK_FLAG = OFF;                                        | AIR_BREATHER_OPTIMIZAT
     E|             .                                                        |
2648 M|     NORMAL_SHOCK_FLAG = OFF;                                         | AIR_BREATHER_OPTIMIZAT
     E|          .                                                           |
2649 M|     EXPANSION_FLAG = OFF;                                            | AIR_BREATHER_OPTIMIZAT
     E|          .                                                           |
2650 M|     SUBSONIC_FLAG = OFF;                                             | AIR_BREATHER_OPTIMIZAT
2651 M|     CALL THESIS_ALGORITHM;                                           | AIR_BREATHER_OPTIMIZAT
2652 M| CLOSE AIR_BREATHER_OPTIMIZATION;                                     | AIR_BREATHER_OPTIMIZAT

**** B L O C K   S U M M A R Y ****

COMPOOL VARIABLES USED
    FIRST_ITERATION_FLAG*, FIRST_PSI_MAG*, X_STORE*
```

```
**** C O M P I L A T I O N   L A Y O U T ****
RUN_POOL: EXTERNAL COMPOOL;
AIR_BREATHER_OPTIMIZATION: PROCEDURE;
   MODEL_DRIVER: PROCEDURE;
      VEHICLE: PROCEDURE;
         FIRST_STAGE: PROCEDURE;
         SCND_STAGE: PROCEDURE;
      ENVIRONMENT: PROCEDURE;
   THESIS_ALGORITHM: PROCEDURE;
      U_COMPUTE: PROCEDURE;
      STATE_DERIVS: PROCEDURE;
      HAM_SUB_U: PROCEDURE;
      DELTA_U_V_CALC: PROCEDURE;
      RUNGE_KUTTA: PROCEDURE;
         RK_DERIV: PROCEDURE;
      TIME_SET: PROCEDURE;
      ITERATION_DRIVER: PROCEDURE;
         STATE_INTEGRATION: PROCEDURE;
      OPTIMIZATION: PROCEDURE;
```

```
S Y M B O L  &  C R O S S   R E F E R E N C E   T A B L E   L I S T I N G :

       (CROSS REFERENCE FLAG KEY:  4 = ASSIGNMENT, 2 = REFERENCE, 1 = SUBSCRIPT USE, 0 = DEFINITION)

 DCL  NAME                        TYPE                        ATTRIBUTES & CROSS REFERENCE

 250  A_FIND                      BIT(1)                      ALIGNED, AUTOMATIC        XREF: 0 0250  4 0313  2 0314  4 0318
2533  AA_INDEX                    INTEGER ARRAY               ARRAY(31), SINGLE, ALIGNED, STATIC, INITIAL      XREF: 0 2533
                                                              2 2550  2 2552
2534  AA_INIT                     SCALAR ARRAY                ARRAY(31), DOUBLE, ALIGNED, STATIC, INITIAL      XREF: 0 2534
                                                              2 2552
 102  AIR_BREATHER_OPTIMIZATION   PROCEDURE                     XREF: 0 0102  NOT REFERENCED
  53  ALT_FINAL                   SCALAR                      DOUBLE, ALIGNED, CONSTANT         XREF: 0 0053  2 1920  2 1923
  46  ALT_METER_INTERVAL          SCALAR                      DOUBLE, ALIGNED, CONSTANT         XREF: 0 0046  2 0513  2 0530
                                                              2 0534
 511  ALTITUDE                    SCALAR                      DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0511  4 0513  2 0514
                                                              2 0529  2 0533
 404  ANG_FIND                    BIT(1)                      ALIGNED, AUTOMATIC        XREF: 0 0404  4 0417  2 0418  4 0422
 216  ANGLE_OF_ATTACK             SCALAR                      DOUBLE, ALIGNED, STATIC   XREF: 0 0216  2 0346  2 0347  2 0356
                                                              2 0414  2 0419  2 0460  4 0577  4 0581  2 0589  2 0606  2 0694
                                                              2 0695
2525  ANGLE_OF_ATTACK_INITIAL     SCALAR ARRAY                ARRAY(2001), DOUBLE, ALIGNED, AUTOMATIC   XREF: 0 2525  4 2552
                                                              2 2563
 236  ASPECT_RATIO                SCALAR                      DOUBLE, ALIGNED, STATIC   XREF: 0 0236  4 0309  2 0310  2 0315
                                                              2 0341  2 0342  2 0348
  44  ATM_DENS                    SCALAR ARRAY                ARRAY(51), DOUBLE, ALIGNED, CONSTANT      XREF: 0 0044  2 0518
                                                              2 0525  2 0531  2 0532
  43  ATM_TEMP                    SCALAR ARRAY                ARRAY(51), DOUBLE, ALIGNED, CONSTANT      XREF: 0 0043  2 0519
                                                              2 0524  2 0529  2 0530
 861  B                           5 X 5 MATRIX                DOUBLE, ALIGNED, STATIC   XREF: 0 0861  4 0880  2 0894  2 0887
                                                              2 0889  2 0398  6 0899  4 0900  2 0905  6 0906  4 0907  6 0911
                                                              6 0917  6 0922  6 0924  2 0931  6 0932  4 0933  2 0938  6 0939
                                                              4 0940  2 0953  2 0955
 877  B_HOLD                      SCALAR                      DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0877  4 0898  2 0900
                                                              4 0905  2 0907  4 0931  2 0933  4 0938  2 0940
 874  B_INT_1                     SCALAR ARRAY                ARRAY(5), DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0874  4 0882
                                                              4 0890  2 0895  2 0896  2 0928
 875  B_INT_2                     SCALAR ARRAY                ARRAY(5), DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0875  4 0883
                                                              4 0891  2 0902  2 0903  2 0935
 199  BETA_LOWER_BOUND            SCALAR                      DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0199  4 0618  2 0622
                                                              4 0626  2 0628
 201  BETA_NEW_BOUND              SCALAR                      DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0201  4 0622  2 0623
                                                              2 0625  2 0626
 138  BETA_NOSE_SHOCK             SCALAR                      DOUBLE, ALIGNED, STATIC   XREF: 0 0138  4 0628  2 0629  2 0630
                                                              2 1249  2 1250  2 1252  2 1253
 213  BETA_THETA_MAX              SCALAR                      DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0213  4 0610  2 0611
                                                              2 0619
 198  BETA_UPPER_BOUND            SCALAR                      DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0198  4 0619  2 0622
                                                              4 0625  2 0628
 239  BODY_WING_MASS              SCALAR                      DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0239  4 0268  2 0274
 867  C_SUB_J                     SCALAR                      DOUBLE, ALIGNED, STATIC   XREF: 0 0867  4 0961  4 0962  2 0967
                                                              2 0969
 756  C_SUB_PSI                   SCALAR                      DOUBLE, ALIGNED, STATIC   XREF: 0 0756  4 0958  2 0961  2 0967
                                                              2 0969
 873  C_SUB_PSI_KEEP              SCALAR                      DOUBLE, ALIGNED, STATIC   XREF: 0 0873  4 0944  4 0950  4 0951
                                                              2 0958
```

```
DCL  NAME                     TYPE                 ATTRIBUTES & CROSS REFERENCE

  85 CAP_CA                   SCALAR               DOUBLE, ALIGNED, INITIAL  XREF: 0 0085  2 1163  2 1216  2 2002
 733 CAP_LAMBDA_1             7 X 5 MATRIX ARRAY   ARRAY(1001), DOUBLE, ALIGNED, STATIC      XREF: 0 0733  2 0954
                                                   2 1603  2 1685  2 1687  2 1609  2 1691  2 1709  2 1712  2 1724
                                                   4 2104  4 2105  4 2106  4 2107  4 2108  4 2109  4 2110  4 2111
                                                   4 2112  4 2113  4 2114  4 2180  2 2326
 743 CAP_LAMBDA_1_DOT         7 X 5 MATRIX         DOUBLE, ALIGNED, STATIC   XREF: 0 0743  4 1670  4 1672  4 1674
                                                   4 1676  4 1678  2 1720
 762 CAP_LAMBDA_1_FLAG        BIT(1)               ALIGNED, STATIC   XREF: 0 0762  2 1172  2 1185  2 1667  2 1719
                                                   2 1744  2 1751  2 1625  4 1907  4 2174  4 2184
1050 CAP_LAMBDA_1_HOLD        7 X 5 MATRIX         DOUBLE  ALIGNED, STATIC   XREF: 0 1050  4 1177  2 1670  2 1672
                                                   2 1674  2 1676  2 1678
 734 CAP_LAMBDA_2             10 X 5 MATRIX        DOUBLE, ALIGNED, STATIC   XREF: 0 0734  2 1760  4 2115  4 2120
                                                   4 2121  4 2122  4 2123  4 2124  4 2125  4 2126  4 2127  4 2128
                                                   4 2193  2 2333
 744 CAP_LAMBDA_2_DOT         10 X 5 MATRIX        DOUBLE, ALIGNED, STATIC   XREF: 0 0744  4 1683  4 1685  4 1687
                                                   4 1689  4 1691  2 1722
 763 CAP_LAMBDA_2_FLAG        BIT(1)               ALIGNED, STATIC   XREF: 0 0763  2 1185  2 1600  2 1721  2 1744
                                                   2 1757  2 1825  4 1908  4 2107  4 2197
 127 CAP_PHI                  SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0127  2 0305  2 0308  4 0578
                                                   4 0582  2 0701  2 1461  2 1476  2 1477  2 1437  2 1492  2 1495
                                                   2 1521  2 1533  2 1588  2 1591  2 1599
2526 CAP_PHI_INITIAL          SCALAR ARRAY         ARRAY(2001), DOUBLE, ALIGNED, AUTOMATIC   XREF: 0 2526  4 2558
                                                   2 2564  2 2565
  83 CAP_Q                    SCALAR               DOUBLE, ALIGNED, INITIAL  XREF: 0 0083  2 1161  2 1938  2 1960
                                                   2 2000
 162 CD                       SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0162  4 0380  6 0383  2 0392
                                                   4 0396  4 0562  2 1464
 235 CD0                      SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0235  4 0360  4 0363  4 0376
                                                   2 0380
1871 CHECK1                   SCALAR               DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1871  4 2096  2 2109
                                                   2 2110  2 2111  2 2112  2 2113  2 2114  4 2119  2 2120  2 2121
 161 CL                       SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0161  4 0346  2 0351  2 0380
                                                   2 0381  6 0384  2 0391  4 0551  2 1463  2 1479
 770 CONFIG_INDEX             INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0770  4 2276  1 2277  1 2278
                                                   1 2280  1 2325  1 2328  4 2331  1 2333
1119 COS_A                    SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 1119  4 1363  2 1364  2 1365
                                                   2 1366  2 1367  2 1368  2 1369  2 1455  2 1456  2 1430  2 1451
1070 COS_B_T                  SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 1070  4 1253  2 1259  2 1415
1067 COS_BETA                 SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 1067  4 1250  2 1255  2 1258
                                                   2 1262  2 1263  2 1264  2 1266  2 1267  2 1268  2 1413  2 1415
                                                   2 1417
1045 COS_DELTA                SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 1045  4 1501  2 1507  2 1508
                                                   2 1511  2 1512  2 1515  2 1516  2 1578  2 1579
 166 COS_VEHICLE_ANGLE        SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0166  4 0680  2 0713  2 0714
                                                   2 1611  2 1612  2 1614
 781 CCST                     SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0781  2 2507  4 2578  2 2600
                                                   4 2601
2527 CP_INDEX                 INTEGER ARRAY        ARRAY(23), SINGLE, ALIGNED, STATIC, INITIAL      XREF: 0 2527
                                                   2 2556  2 2558
2528 CP_INIT                  SCALAR ARRAY         ARRAY(23), DOUBLE, ALIGNED, STATIC, INITIAL      XREF: 0 2528
                                                   2 2558
1131 DAE1_DPI                 10 - VECTOR          DOUBLE, ALIGNED, STATIC   XREF: 0 1131  4 1396  4 1439  4 1445
                                                   2 1454  2 1457
1132 DAE2_DPI                 10 - VECTOR          DOUBLE, ALIGNED, STATIC   XREF: 0 1132  4 1397  4 1440  4 1443
                                                   2 1456  2 1458
```

```
 DCL  NAME                TYPE                ATTRIBUTES & CROSS REFERENCE

1125  DAR_DPI             10 - VECTOR         DOUBLE, ALIGNED, STATIC   XREF: 0 1125  4 1390  4 1450  2 1463
                                              2 1464
1126  DAW1_DPI            10 - VECTOR         DOUBLE, ALIGNED, STATIC   XREF: 0 1126  4 1391  4 1447  4 1451
                                              2 1463  2 1464
1127  DAW2_DPI            10 - VECTOR         DOUBLE, ALIGNED, STATIC   XREF: 0 1127  4 1392  4 1402  2 1463
                                              2 1464
1122  DBETA_DM0           SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1122  4 1240  4 1255  2 1298
                                              2 1299  2 1300
1135  DBETA_DP1           SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1135  4 1412  2 1413  2 1415
                                              2 1417
1076  DBETA_DR            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1076  4 1298  2 1301  2 1304
                                              2 1310
1078  DBETA_DTDOT         SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1078  4 1300  2 1303  2 1306
                                              2 1312
1077  DBETA_DUR           SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1077  4 1299  2 1302  2 1305
                                              2 1311
 171  DCD_DCL2            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0171  4 0343  4 0344  2 0351
                                              2 0330  2 0381  6 0399  4 0399  4 0560  2 1479
 257  DCD0_DM0            SCALAR              DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0257  4 0358  4 0377
                                              2 0381
 160  DCD1_DAR            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0160  4 0351  6 0387  4 0398
                                              4 0568  2 1464
 157  DCD1_DM0            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0157  4 0381  6 0385  4 0397
                                              4 0565  2 1345  2 1349  2 1353
1101  DCD1_DR             SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1101  4 1345  2 1359
1109  DCD1_DTDOT          SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1109  4 1353  2 1361
1105  DCD1_DUR            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1105  4 1349  2 1360
 158  DCD2_DM0            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0158  4 0472  4 0476  6 0482
                                              4 0491  4 0566  2 1346  2 1350  2 1354
1102  DCD2_DR             SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1102  4 1346  2 1359
1110  DCD2_DTDOT          SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1110  4 1354  2 1361
 169  DCD2_DUA1           SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0169  4 0464  6 0484  4 0492
                                              2 1479
1106  DCD2_DUR            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1106  4 1350  2 1360
 170  DCL_DALPHA          SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0170  4 0345  2 0346  6 0390
                                              4 0559  2 1478  2 1479
 159  DCL1_DAR            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0159  4 0347  2 0351  6 0388
                                              4 0567  2 1463
 155  DCL1_DM0            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0155  4 0356  2 0381  6 0386
                                              4 0563  2 1343  2 1347  2 1351
1099  DCL1_DR             SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1099  4 1343  2 1356
1107  DCL1_DTDOT          SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1107  4 1351  2 1358
1103  DCL1_DUR            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1103  4 1347  2 1357
 156  DCL2_DM0            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0156  4 0471  4 0475  6 0481
                                              4 0564  2 1344  2 1348  2 1352
1100  DCL2_DR             SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1100  4 1344  2 1356
1108  DCL2_DTDOT          SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1108  4 1352  2 1358
 168  DCL2_DUA1           SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0168  4 0463  6 0483  2 1478
1104  DCL2_DUR            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1104  4 1348  2 1357
1143  DD_DPI              SCALAR              DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1143  4 1464  2 1465
                                              2 1466
1115  DD_DR               SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1115  4 1359  2 1364  2 1367
1117  DD_DTDOT            SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1117  4 1361  2 1366  2 1369
1152  DD_DUA1             SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1152  4 1479  2 1480  2 1481
1116  DD_DUR              SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 1116  4 1360  2 1365  2 1368
```

```
DCL  NAME                        TYPE                  ATTRIBUTES & CROSS REFERENCE

 255 DDCD_DCL2_DAR               SCALAR                DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0255  4 0349  4 0350
                                                       2 0351
 256 DDCD_DCL2_DMO               SCALAR                DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0256  4 0357  2 0381
1038 DDELTA_DR                   SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1038  4 1497  2 1505  2 1506
1040 DDELTA_DTDOT                SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1040  4 1499  2 1513  2 1514
1039 DDELTA_DUR                  SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1039  4 1498  2 1509  2 1510
  20 DEGREES_PER_RADIAN          SCALAR                DOUBLE, ALIGNED, CONSTANT         XREF: 0 0020  2 0346  2 0347
                                                       2 0356  2 1478  2 1479
  34 DELIVERED_MASS              SCALAR                DOUBLE, ALIGNED, CONSTANT         XREF: 0 0034  2 0413
  35 DELIVERED_PLANFORM_AREA     SCALAR                DOUBLE, ALIGNED, CONSTANT         XREF: 0 0035  2 0411  2 1402
 179 DELTA_ANGLE                 SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 0179  2 0265  4 0547
 757 DELTA_COST                  SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 0757  4 2600  2 2602  2 2606
                                                       2 2608  2 2610  2 2611
  80 DELTA_COST_CHECK            SCALAR                DOUBLE, ALIGNED, INITIAL  XREF: 0 0080  2 2608
  81 DELTA_COST_SHRINK           SCALAR                DOUBLE, ALIGNED, INITIAL  XREF: 0 0081  2 2609
 737 DELTA_OMEGA_I_TIME          INTEGER ARRAY         ARRAY(3), SINGLE, ALIGNED, STATIC         XREF: 0 0737  4 2348
                                                       4 2355  2 2369
 730 DELTA_U                     2 - VECTOR ARRAY      ARRAY(1001), DOUBLE, ALIGNED, STATIC      XREF: 0 0730  4 0967
                                                       2 2469  2 2474  4 2485  2 2490
1877 DELTA_U_NEW                 2 - VECTOR ARRAY      ARRAY(1001), DOUBLE, ALIGNED, AUTOMATIC   XREF: 0 1877  4 2469
                                                       4 2474  2 2485
 860 DELTA_U_V_CALC              PROCEDURE               XREF: 0 0860  2 2338
 731 DELTA_V                     13 - VECTOR           DOUBLE, ALIGNED, STATIC   XREF: 0 0731  4 0969  2 2340  2 2343
                                                       2 2352  2 2353  4 2625
1890 DFP3                        SCALAR                DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1890  4 2135  4 2136
                                                       2 2138
1024 DFR_DMI                     SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1024  4 1504  2 1558
1000 DFR_DP                      SCALAR ARRAY          ARRAY(10), DOUBLE, ALIGNED, STATIC        XREF: 0 1000  4 1578
                                                       4 1582  2 1604
1010 DFR_DR                      SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1010  4 1507  2 1555
1029 DFR_DTDOT                   SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1029  4 1515  2 1557
1147 DFR_DUA                     2 - VECTOR            DOUBLE, ALIGNED, STATIC   XREF: 0 1147  4 1611  4 1613  2 1618
1025 DFR_DUR                     SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1025  4 1511  2 1556
1001 DFTHETA_DP                  SCALAR ARRAY          ARRAY(10), DOUBLE, ALIGNED, STATIC        XREF: 0 1001  4 1579
                                                       4 1583  2 1605
1011 DFTHETA_DR                  SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1011  4 1508  2 1561
1030 DFTHETA_DTDOT               SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1030  4 1516  2 1563
1148 DFTHETA_DUA                 2 - VECTOR            DOUBLE, ALIGNED, STATIC   XREF: 0 1148  4 1612  4 1614  2 1619
1026 DFTHETA_DUR                 SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1026  4 1512  2 1562
1133 DG_DPI                      10 - VECTOR           DOUBLE, ALIGNED, STATIC   XREF: 0 1133  4 1399  4 1468  2 1470
1091 DISP1_DM2                   SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1091  4 1313  2 1321  2 1322
                                                       2 1323  2 1426  2 1485
1138 DISP1_DP1                   SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1138  4 1426  2 1431
1092 DISP2_DM2                   SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1092  4 1315  2 1333  2 1334
                                                       2 1335  2 1427  2 1491
1139 DISP2_DP1                   SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1139  4 1427  2 1433
 758 DJS                         SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 0758  2 0959  4 2571  2 2608
                                                       6 2609  2 2610  4 2611  2 2612  4 2613
  94 DJS_INIT                    SCALAR                DOUBLE, ALIGNED, INITIAL  XREF: 0 0094  2 2571  2 2612  2 2613
 797 DJS_SCALE                   SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 0797  4 0959  2 0961
1142 DL_DPI                      SCALAR                DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1142  4 1463  2 1465
                                                       2 1466
1112 DL_DR                       SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1112  4 1356  2 1364  2 1367
1114 DL_DTDOT                    SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1114  4 1358  2 1366  2 1369
1151 DL_DUA1                     SCALAR                DOUBLE, ALIGNED, STATIC   XREF: 0 1151  4 1478  2 1480  2 1481
```

DCL	NAME	TYPE	ATTRIBUTES & CROSS REFERENCE
1113	DL_DUR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1113 4 1357 2 1365 2 1368
728	DM_DP	SCALAR ARRAY	ARRAY(10), DOUBLE, ALIGNED, STATIC XREF: 0 0728 2 1468 2 1578 2 1604 2 1605 4 2133 4 2137 4 2138 4 2139 4 2140 4 2141 4 2142 4 2143 4 2144 4 2145
1008	DMASSFLUX_DP1	SCALAR ARRAY	ARRAY(10), DOUBLE, ALIGNED, STATIC XREF: 0 1008 4 1398 4 1434 6 1457 2 1588
1009	DMASSFLUX_DP2	SCALAR ARRAY	ARRAY(10), DOUBLE, ALIGNED, STATIC XREF: 0 1009 4 1399 4 1435 6 1458 2 1591
1017	DMASSFLUX_DR1	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1017 4 1372 2 1522
1018	DMASSFLUX_DR2	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1018 4 1375 2 1534
1036	DMASSFLUX_DTDOT1	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1036 4 1374 2 1524
1037	DMASSFLUX_DTDOT2	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1037 4 1377 2 1536
1022	DMASSFLUX_DUR1	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1022 4 1373 2 1523
1023	DMASSFLUX_DUR2	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1023 4 1376 2 1535
154	DMCR_DM2	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0154 4 0292 4 0297 4 0569 2 1332 2 1375 2 1376 2 1377 2 1428
1140	DMCR_DP1	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1140 4 1428 2 1433 2 1435
1058	DM0_DR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1058 4 1232 2 1298 2 1301 2 1304 2 1310 2 1343 2 1344 2 1345 2 1346
1060	DM0_DTDOT	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1060 4 1234 2 1300 2 1303 2 1306 2 1312 2 1351 2 1352 2 1353 2 1354
1059	DM0_DUR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1059 4 1233 2 1299 2 1302 2 1305 2 1311 2 1347 2 1348 2 1349 2 1350
1002	DM1DOT_DP	SCALAR ARRAY	ARRAY(10), DOUBLE, ALIGNED, STATIC XREF: 0 1002 4 1508 4 1509 4 1596 2 1606
1012	DM1DOT_DR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1012 4 1522 4 1527 4 1545 2 1567
1031	DM1DOT_DTDOT	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1031 4 1524 4 1529 4 1547 2 1569
1144	DM1DOT_DUA	2 - VECTOR	DOUBLE, ALIGNED, STATIC XREF: 0 1144 4 1472 4 1486 4 1487 2 1620
1027	DM1DOT_DUR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1027 4 1523 4 1528 4 1546 2 1568
1061	DM2_DBETA	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1061 4 1235 4 1258 2 1301 2 1302 2 1303
1123	DM2_DM0	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1123 4 1241 4 1259 4 1273 4 1282 4 1292 2 1301 2 1302 2 1303
1137	DM2_DP1	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1137 4 1409 4 1415 4 1422 2 1423 2 1424 2 1426 2 1427 2 1428 2 1429 2 1486 2 1491
1079	DM2_DR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1079 4 1301 2 1304 2 1307 2 1310 2 1321 2 1333 2 1375
1081	DM2_DTDOT	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1081 4 1303 2 1306 2 1309 2 1312 2 1323 2 1335 2 1377
1080	DM2_DUR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1080 4 1302 2 1305 2 1308 2 1311 2 1322 2 1334 2 1376
1003	DM2DOT_DP	SCALAR ARRAY	ARRAY(10), DOUBLE, ALIGNED, STATIC XREF: 0 1003 4 1591 4 1592 4 1597 2 1607
1013	DM2DOT_DR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1013 4 1534 4 1539 4 1548 2 1570
1032	DM2DOT_DTDOT	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1032 4 1536 4 1541 4 1550 2 1572
1145	DM2DOT_DUA	2 - VECTOR	DOUBLE, ALIGNED, STATIC XREF: 0 1145 4 1473 4 1491 4 1492 2 1621
1028	DM2DOT_DUR	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 1028 4 1535 4 1540 4 1549 2 1571

```
 DCL  NAME               TYPE             ATTRIBUTES & CROSS REFERENCE

1004  DM3DOT_DP          SCALAR ARRAY     ARRAY(10), DOUBLE, ALIGNED, STATIC       XREF: 0 1004  4 1593
                                          4 1599  4 1600  2 1603
1146  DM3DOT_DUA         2 - VECTOR       DOUBLE, ALIGNED, STATIC   XREF: 0 1146  4 1474  4 1495  2 1622
1006  DN_DP              SCALAR ARRAY     ARRAY(10), DOUBLE, ALIGNED, STATIC       XREF: 0 1006  4 1466
                                          2 1468  2 1578  2 1579
1015  DN_DR              SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1015  4 1367  2 1385  2 1506
1034  DN_DTDOT           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1034  4 1369  2 1387  2 1514
 723  DN_DUA             2 - VECTOR       DOUBLE, ALIGNED, STATIC   XREF: 0 0723  4 1481  2 1611  2 1612
                                          2 1624  4 2640
1020  DN_DUR             SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1020  4 1368  2 1386  2 1510
1005  DP_DP              SCALAR ARRAY     ARRAY(10), DOUBLE, ALIGNED, STATIC       XREF: 0 1005  4 1465
                                          2 1468  2 1577
1014  DP_DR              SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1014  4 1364  2 1385  2 1505
1033  DP_DTDOT           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1033  4 1366  2 1387  2 1513
 722  DP_DUA             2 - VECTOR       DOUBLE, ALIGNED, STATIC   XREF: 0 0722  4 1480  2 1611  2 1612
                                          2 1624  4 2639
1019  DP_DUR             SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1019  4 1365  2 1386  2 1509
1153  DFHI_DUA2          SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1153  4 1476  2 1477  2 1487
                                          2 1492  2 1495
 145  DRAG               SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 0145  4 0693  2 0696  2 0697
                                          2 1359  2 1360  2 1361
 126  DRO_DR             SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 0126  4 0534  4 0571  2 1310
                                          2 1356  2 1359  2 1939  2 1969
1063  DRO2_DBETA         SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1063  4 1237  4 1266  2 1268
                                          2 1310  2 1311  2 1312
1074  DRO2_DM0           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1074  4 1268  4 1276  4 1285
                                          4 1295  2 1310  2 1311  2 1312
1121  DRO2_DM2           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1121  4 1239  4 1288  2 1310
                                          2 1311  2 1312
1134  DRO2_DP1           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1134  4 1407  4 1413  4 1424
                                          2 1431  2 1433  2 1434  2 1435
1088  DRO2_DR            SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1088  4 1310  2 1321  2 1333
                                          2 1372  2 1375
1075  DRO2_DRO_0         SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1075  4 1269  4 1277  4 1286
                                          4 1296  2 1310
1090  DRO2_DTDOT         SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1090  4 1312  2 1323  2 1335
                                          2 1374  2 1377
1089  DRO2_DUR           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1089  4 1311  2 1322  2 1334
                                          2 1373  2 1376
 253  DRY_TANK_VOLUME    SCALAR           DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0253  4 0269  2 0270
                                          2 0272
1007  DT_DP              SCALAR ARRAY     ARRAY(10), DOUBLE, ALIGNED, STATIC       XREF: 0 1007  4 1462
                                          2 1468  2 1475  2 1577
1016  DT_DR              SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1016  4 1380  2 1385  2 1505
1035  DT_DTDOT           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1035  4 1382  2 1387  2 1513
1149  DT_DUA             2 - VECTOR       DOUBLE, ALIGNED, STATIC   XREF: 0 1149  4 1475  4 1477  2 1611
                                          2 1612  2 1613  2 1614  2 1624
1021  DT_DUR             SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1021  4 1381  2 1386  2 1509
1128  DTH1_DPI           10 - VECTOR      DOUBLE, ALIGNED, STATIC   XREF: 0 1128  4 1393  4 1431  6 1454
                                          2 1462  2 1486
1129  DTH2_DPI           10 - VECTOR      DOUBLE, ALIGNED, STATIC   XREF: 0 1129  4 1394  4 1433  6 1456
                                          2 1462  2 1491
1096  DTH2_DR            SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1096  4 1333  4 1338  2 1380
1098  DTH2_DTDOT         SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1098  4 1335  4 1340  2 1382
1097  DTH2_DUR           SCALAR           DOUBLE, ALIGNED, STATIC   XREF: 0 1097  4 1334  4 1339  2 1381
```

```
 DCL  NAME                 TYPE               ATTRIBUTES & CROSS REFERENCE

1130  DTH3_DPI             10 - VECTOR        DOUBLE, ALIGNED, STATIC   XREF: 0 1130  4 1395  4 1461  2 1462
 153  DT0_DR               SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 0153  4 0530  4 0570  2 1232
                                              2 1304
1093  DT1_DR               SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1093  4 1321  4 1326  2 1300
1095  DT1_DTDOT            SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1095  4 1323  4 1328  2 1302
1094  DT1_DUR              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1094  4 1322  4 1327  2 1301
1062  DT2_DBETA            SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1062  4 1236  4 1262  2 1264
                                              2 1304  2 1305  2 1306
1072  DT2_DM0              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1072  4 1264  4 1274  4 1283
                                              4 1293  2 1304  2 1305  2 1306
1120  DT2_DM2              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1120  4 1238  4 1287  2 1304
                                              2 1305  2 1306
1136  DT2_DP1              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1136  4 1408  4 1417  4 1423
                                              2 1429
1082  DT2_DR               SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1082  4 1304  2 1307
1084  DT2_DTDOT            SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1084  4 1306  2 1309
1073  DT2_DT0              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1073  4 1265  4 1275  4 1284
                                              4 1294  2 1304
1083  DT2_DUR              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1083  4 1305  2 1308
1064  DU2_DM2              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1064  4 1244  2 1246  2 1307
                                              2 1308  2 1309  2 1429
1141  DU2_DP1              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1141  4 1429  2 1431  2 1433
                                              2 1434  2 1435
1085  DU2_DR               SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1085  4 1307  2 1321  2 1333
                                              2 1372  2 1375
1087  DU2_DTDOT            SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1087  4 1309  2 1323  2 1335
                                              2 1374  2 1377
1065  DU2_DT2              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1065  4 1246  2 1307  2 1308
                                              2 1309  2 1429
1086  DU2_DUR              SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 1086  4 1303  2 1322  2 1334
                                              2 1373  2 1376
 103  DYNAMIC_P0           SCALAR             DOUBLE, ALIGNED, STATIC   XREF: 0 0103  2 0391  2 0392  2 0485
                                              2 0486  4 0592  2 1160  2 1161  2 1356  2 1357  2 1358  2 1359
                                              2 1360  2 1361  2 1463  2 1464  2 1478  2 1479  2 1936  2 1938
                                              2 1939  2 1965  2 1968  2 1969  2 1999  2 2000
   5  EARTH_MASS           SCALAR             DOUBLE, ALIGNED, CONSTANT        XREF: 0 0005  2 0540  2 1503
  11  EARTH_OMEGA          SCALAR             DOUBLE, ALIGNED, CONSTANT        XREF: 0 0011  2 0538  2 1230
                                              2 1939  2 1941  2 1969  2 1971
   3  EARTH_RADIUS         SCALAR             DOUBLE, ALIGNED, CONSTANT        XREF: 0 0003  2 0572  2 0342
                                              2 1195  2 1923  2 1939  2 1941  2 1959  2 1971  2 2096  2 2111
                                              2 2294  2 2314
 508  ENVIRONMENT          PROCEDURE            XREF: 0 0508  2 0591
 151  EXPANSION_FLAG       BIT(1)             ALIGNED, STATIC   XREF: 0 0151  4 0557  4 0642  2 1279  2 1419
                                              4 2649
 211  EXTREMAL_ARRAY       SCALAR ARRAY       ARRAY(2), DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0211  4 0616
                                              4 0617  2 0618
 740  F                    7 X 7 MATRIX       DOUBLE, ALIGNED, STATIC   XREF: 0 0740  4 1552  4 1553  4 1554
                                              4 1555  4 1556  4 1557  4 1558  6 1559  6 1560  4 1561  4 1562
                                              4 1563  4 1564  6 1565  6 1566  4 1567  4 1568  4 1569  4 1570
                                              4 1571  4 1572  2 1630  2 1637  2 1642  2 1649  2 1654  2 1666
 786  FD_COUNT             INTEGER            SINGLE, ALIGNED, STATIC   XREF: 0 0786  6 1217  2 1218  4 1219
                                              2 1615  2 1628  2 1635  2 1640  2 1647  2 1652  4 2632
 790  FD_FLAG              BIT(1)             ALIGNED, STATIC   XREF: 0 0790  2 1192  4 1193  4 2164
  45  FEET_PER_METER       SCALAR             DOUBLE, ALIGNED, CONSTANT        XREF: 0 0045  2 0513  2 0530
                                              2 0534
```

```
DCL  NAME                      TYPE                   ATTRIBUTES & CROSS REFERENCE

 100 FINAL_STEP                INTEGER                SINGLE, ALIGNED   XREF: 0 0100  2 0823  2 0824  2 0963  4 2004
                                                      2 2048  2 2051  2 2058  2 2080  1 2081  1 2082  1 2083  1 2084
                                                      2 2085  1 2096  1 2110  1 2111  2 2146  2 2147  2 2221  2 2222
                                                      2 2223  2 2278  2 2370  2 2372  2 2373  2 2393  2 2410  2 2422
                                                      2 2430  2 2431  2 2440  2 2441  2 2408  2 2492  1 2500  2 2501
                                                      2 2502  3 2503  3 2504  1 2507  1 2577  1 2599  4 2622
 754 FINAL_TIME_STEP           SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0754  4 1990  2 1992  2 2059
                                                      2 2081  2 2082  2 2083  2 2084  2 2148  2 2224  2 2437  2 2504
                                                      4 2630
1899 FINAL_U_STORE             2 - VECTOR ARRAY       ARRAY(4), DOUBLE, ALIGNED, STATIC         XREF: 0 1899  4 1952
                                                      2 1985  2 1986
 242 FIND_FLAG                 BIT(1)                 ALIGNED, AUTOMATIC        XREF: 0 0242  4 0365  2 0366  4 0373
 720 FIRST_DERIV_FLAG          BIT(1)                 ALIGNED, STATIC   XREF: 0 0720  2 1169  4 1171  4 1784  4 1804
                                                      4 1815  4 1828  4 2637
  74 FIRST_ITERATION_FLAG      BIT(1)                 ALIGNED   XREF: 0 0074  2 2067  4 2070  4 2643
 808 FIRST_PASS_PSI_MAG        SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0808  2 0960  4 2574  4 2596
  72 FIRST_PSI_MAG             SCALAR                 DOUBLE, ALIGNED   XREF: 0 0072  2 0946  2 0960  2 2507  2 2604
                                                      4 2644
 976 FIRST_RK_VAL_N            10 X 5 MATRIX          DOUBLE, ALIGNED, STATIC   XREF: 0 0976  4 1793  2 1836
 234 FIRST_STAGE               PROCEDURE               XREF: 0 0234  2 0499  2 0503
 117 FIRST_STAGE_DRY_MASS      SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0117  4 0274  2 0700  2 0941
                                                      2 1648  2 1932
 181 FIRST_STAGE_LENGTH        SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0181  2 0266  2 0267  4 0549
  33 FS_A_VAL                  SCALAR ARRAY           ARRAY(5), DOUBLE, ALIGNED, CONSTANT       XREF: 0 0033  2 0310
                                                      2 0315  2 0341  2 0342  2 0347  2 0348  2 0350
  30 FS_CD_MAT                 5 X 10 MATRIX          DOUBLE, ALIGNED, CONSTANT         XREF: 0 0030  2 0338  2 0340
                                                      2 0353  2 0355
  31 FS_CL_MAT                 5 X 10 MATRIX          DOUBLE, ALIGNED, CONSTANT         XREF: 0 0031  2 0337  2 0339
                                                      2 0352  2 0354
 222 FS_DRAG                   SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0222  4 0392  2 0393  4 0395
                                                      4 0689  2 0693
  52 FS_FIXED_PARAMETERS       INTEGER                SINGLE, ALIGNED, CONSTANT         XREF: 0 0052  2 1467  2 1575
                                                      2 1602
 221 FS_LIFT                   SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0221  4 0391  4 0688  2 0692
  32 FS_M_VAL                  SCALAR ARRAY           ARRAY(10), DOUBLE, ALIGNED, CONSTANT      XREF: 0 0032  2 0323
                                                      2 0328  2 0336  2 0356  2 0357
2538 FUEL_COST                 SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 2538  4 2577  2 2578
                                                      4 2599  2 2600  2 2601
 238 FUSELAGE_SURFACE_AREA     SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0238  4 0267  2 0268
 254 FUSELAGE_VOLUME           SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0254  4 0266  2 0267
                                                      2 0269
 191 F1                        SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0191  4 0336  2 0337
                                                      2 0338  2 0339  2 0340  4 0341  2 0344  2 0345  2 0352  2 0353
                                                      2 0354  2 0355  4 0382  2 0383  2 0384  2 0385  2 0386  2 0387
                                                      2 0388  2 0389  2 0390  4 0453  2 0454  2 0455  2 0456  2 0457
                                                      4 0460  2 0461  2 0462  2 0467  2 0468  2 0469  2 0470  4 0478
                                                      2 0479  2 0480  2 0481  2 0482  2 0483  2 0484  4 0531  2 0532
                                                      2 0533  2 0534  4 0694  2 0696  2 0697
 977 F1                        7 X 7 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0977  4 1630  2 1670
 978 F2                        7 X 7 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0978  4 1637  2 1672
 192 F2                        SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0192  4 0532  2 0533
                                                      4 0695  2 0696  2 0697
 979 F3                        7 X 7 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0979  4 1642  2 1674
 193 F3                        SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0193  4 0651  2 0652
                                                      4 0659  2 0660
```

DCL	NAME	TYPE	ATTRIBUTES & CROSS REFERENCE
980	F4	7 X 7 MATRIX	DOUBLE, ALIGNED, STATIC XREF: 0 0980 4 1649 2 1676
194	F4	SCALAR	DOUBLE, ALIGNED, AUTOMATIC XREF: 0 0194 4 0629 2 0630
			2 0631 2 0632 4 0643 2 0645 2 0652 2 0660
981	F5	7 X 7 MATRIX	DOUBLE, ALIGNED, STATIC XREF: 0 0981 4 1654 2 1678
195	F5	SCALAR	DOUBLE, ALIGNED, AUTOMATIC XREF: 0 0195 4 0644 2 0645
167	G	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0167 4 0540 2 0713 2 1612
86	G_D	SCALAR	DOUBLE, ALIGNED, INITIAL XREF: 0 0086 2 1162 2 1163 2 1216
			2 1225 2 1383 2 1469 2 1623 2 2001 2 2002
132	G_LOAD	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0132 4 0715 2 1162 2 1163
			2 1216 2 1225 2 1227 2 1379 2 1383 2 1468 2 1469 2 1623
			2 1624 2 2001 2 2002
732	G_MAT	7 X 2 MATRIX ARRAY	ARRAY(1001), DOUBLE, ALIGNED, STATIC XREF: 0 0732 2 0855
			2 0856 2 0953 2 0964 4 1618 4 1619 4 1620 4 1621 4 1622
			2 1709 2 1712 4 2626
15	GAMMA0	SCALAR	DOUBLE, ALIGNED, CONSTANT XREF: 0 0015 2 0016 2 0018
			2 0539 2 0609 2 0611 2 0623 2 0630 2 0631 2 0632 2 0636
			2 0637 2 0638 2 0667 2 0672 2 1231 2 1244 2 1254 2 1255
			2 1258 2 1259 2 1262 2 1273 2 1274 2 1412 2 1415 2 1417
17	GAM1	SCALAR	DOUBLE, ALIGNED, CONSTANT XREF: 0 0017 2 0630 2 0636
			2 0666 2 1254 2 1258 2 1259 2 1273 2 1283 2 1285 2 1287
			2 1280 2 1415 2 1423 2 1424
18	GAM2	SCALAR	DOUBLE, ALIGNED, CONSTANT XREF: 0 0018 2 0609 2 0631
			2 0632 2 0637 2 0638 2 0643 2 1225 2 1258 2 1259 2 1262
			2 1266 2 1273 2 1274 2 1276 2 1412 2 1413 2 1415 2 1417
			2 1422
16	GAM3	SCALAR	DOUBLE, ALIGNED, CONSTANT XREF: 0 0016 2 0017 2 0609
			2 0632 2 0638 2 0643 2 0667 2 1262 2 1266 2 1274 2 1276
			2 1282 2 1283 2 1287 2 1412 2 1413 2 1417 2 1422 2 1423
47	GROUND_RO	SCALAR	DOUBLE, ALIGNED, CONSTANT XREF: 0 0047 2 0534 2 0537
12	G0	SCALAR	DOUBLE, ALIGNED, CONSTANT XREF: 0 0012 2 0395 2 0303
			2 0704 2 0705 2 0711 2 0715 2 1163 2 1216 2 1379 2 1468
			2 1486 2 1487 2 1491 2 1492 2 1495 2 1599 2 1624 2 2002
176	H_C	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0176 2 0260 2 0393 4 0544
177	H_C_TJ	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0177 2 0264 2 0395 4 0545
729	H_SUB_U	2 - VECTOR ARRAY	ARRAY(1001), DOUBLE, ALIGNED, STATIC XREF: 0 0729 4 0855
			4 0856 4 0858 2 0967 2 1707 2 1709
851	HAM_SUB_U	PROCEDURE	XREF: 0 0851 2 2201 2 2206
240	HC_TANK_MASS	SCALAR	DOUBLE, ALIGNED, AUTOMATIC XREF: 0 0240 4 0271 2 0274
106	HC_TANK_VOL	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0106 4 0270 2 0271 2 2041
180	HC_TANK_VOL_FRACTION	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0180 2 0270 2 0272 4 0548
794	HELD_FS_MASS	SCALAR	DOUBLE, ALIGNED, STATIC XREF: 0 0794 2 1223 4 1932 2 2311
252	HIGH_A	INTEGER	SINGLE, ALIGNED, AUTOMATIC XREF: 0 0252 4 0311 4 0317
			2 0322 1 0339 1 0340 1 0341 1 0347 1 0350 1 0352 1 0353
			1 0354 1 0355
510	HIGH_ALT	INTEGER	SINGLE, ALIGNED, AUTOMATIC XREF: 0 0510 4 0528 1 0529
			1 0530 1 0531
406	HIGH_ANGLE	INTEGER	SINGLE, ALIGNED, AUTOMATIC XREF: 0 0406 4 0415 4 0421
			2 0426 1 0431 1 0432 1 0438 1 0439 1 0456 1 0457 1 0460
			1 0463 1 0464 1 0467 1 0468 1 0469 1 0470
229	HIGH_CD	SCALAR	DOUBLE, ALIGNED, AUTOMATIC XREF: 0 0229 4 0340 2 0343
			2 0344 2 0350 4 0355 2 0357 4 0432 4 0439 4 0457 2 0462
			2 0464 4 0470 2 0472
246	HIGH_CD0	SCALAR	DOUBLE, ALIGNED, AUTOMATIC XREF: 0 0246 4 0371 2 0376
			2 0377
231	HIGH_CL	SCALAR	DOUBLE, ALIGNED, AUTOMATIC XREF: 0 0231 4 0339 2 0345

```
DCL  NAME                  TYPE                 ATTRIBUTES & CROSS REFERENCE

                                                2 0347  4 0354  2 0356  4 0431  4 0438  4 0456  2 0461  2 0463
                                                4 0469  2 0471
 226 HIGH_M                INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0226  4 0287  2 0290
                                                1 0291  1 0292  4 0324  4 0330  2 0335  1 0336  1 0337  1 0338
                                                1 0339  1 0340  1 0354  1 0355  1 0356  1 0357  4 0448  2 0452
                                                1 0453  1 0454  1 0455  1 0456  1 0457  1 0469  1 0470  1 0471
                                                1 0472
 247 HIGH_M_S              SCALAR               DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0247  4 0372  2 0376
                                                2 0377
 206 HIGH_M_2              SCALAR               DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0206  4 0648  2 0651
                                                6 0655  2 0658  4 0662
 208 HIGH_NU               SCALAR               DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0208  4 0652  2 0653
 800 HOLD_L_X              7 - VECTOR ARRAY     ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0800  2 2513
                                                4 2588
 799 HOLD_M                SCALAR ARRAY         ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0799  2 2512
                                                4 2587
 806 HOLD_NEG_TIME         SCALAR ARRAY         ARRAY(3), DOUBLE, ALIGNED, STATIC         XREF: 0 0806  2 2519
                                                4 2594
 801 HOLD_P                10 - VECTOR          DOUBLE, ALIGNED, STATIC   XREF: 0 0801  2 2514  4 2572  4 2589
 805 HOLD_POS_TIME         SCALAR ARRAY         ARRAY(3), DOUBLE, ALIGNED, STATIC         XREF: 0 0805  2 2518
                                                4 2593
1902 HOLD_PSI              5 - VECTOR           DOUBLE, ALIGNED, STATIC   XREF: 0 1902  4 2079  2 2337  2 2507
 803 HOLD_T                INTEGER ARRAY        ARRAY(3), SINGLE, ALIGNED, STATIC         XREF: 0 0803  2 2517
                                                4 2591
 804 HOLD_U                2 - VECTOR ARRAY     ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0804  2 2515
                                                4 2592
 802 HOLD_U_T              SCALAR ARRAY         ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0802  4 2066
                                                2 2516
 798 HOLD_X                7 - VECTOR ARRAY     ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0798  2 2511
                                                4 2586
   8 HYDROCARBON_DENSITY   SCALAR               DOUBLE, ALIGNED, CONSTANT         XREF: 0 0003  2 0271  2 2041
   7 H2_DENSITY            SCALAR               DOUBLE, ALIGNED, CONSTANT         XREF: 0 0007  2 0273  2 2042
 241 H2_TANK_MASS          SCALAR               DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0241  4 0273  2 0274
 105 H2_TANK_VOL           SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0105  4 0272  2 0273  2 2042
 196 I_BETA                INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0196  4 0621  4 0657
                                                NOT REFERENCED
 789 I_CL                  INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0789  2 0854  2 1192  4 2165
                                                4 2176  4 2189  4 2200
 719 I_FD                  INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0719  2 1669  2 1671  2 1673
                                                2 1675  2 1677  2 1682  2 1684  2 1686  2 1688  2 1690  2 1695
                                                2 1697  2 1699  2 1781  4 1782  4 1783  2 1797  4 1800  4 1801
                                                4 1803  2 1625  4 1826  4 1827  4 2636
1055 I_FK                  INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 1055  4 1202  1 1203  1 1206
                                                4 1226  1 1227  4 1400  2 1401  2 1405  2 1437  2 1442  2 1444
                                                2 1446  2 1448  1 1454  1 1456  1 1457  1 1458  2 1460  1 1462
                                                1 1463  1 1464  1 1465  1 1466  2 1467  1 1468  1 1470  4 1574
                                                2 1575  1 1577  1 1578  1 1579  1 1582  1 1583  1 1588  1 1589
                                                1 1591  1 1592  1 1593  1 1596  1 1597  2 1598  1 1599  1 1600
                                                2 1602  1 1604  1 1605  1 1606  1 1607  1 1608  4 1617  1 1618
                                                1 1619  1 1620  1 1621  1 1622  1 1624  1 1625
1874 I_INIT                INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 1874  4 2090  1 2091
 759 I_J_J                 SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0759  2 0961  2 1768  4 2093
                                                4 2244
 760 I_J_J_DOT             SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0760  4 1707  2 1726
 765 I_J_J_FLAG            BIT(1)               ALIGNED, STATIC   XREF: 0 0765  2 1702  2 1706  2 1725  2 1767
```

```
 DCL  NAME                      TYPE                    ATTRIBUTES & CROSS REFERENCE

                                                       4 1910  4 2240  4 2245
 775  I_LOOP                    INTEGER                SINGLE, ALIGNED, STATIC   XREF: 0 0775  4 2327  1 2328  4 2332
                                                       1 2333
 865  I_MAT                     13 X 5 MATRIX          DOUBLE, ALIGNED, STATIC   XREF: 0 0865  4 0956  2 0969
 248  I_MCR                     INTEGER                SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0248  4 0283  1 0284
                                                       2 0287
1884  I_OMEGA                   INTEGER                SINGLE, ALIGNED, STATIC   XREF: 0 1884  4 1951  1 1952  4 1956
                                                       2 1958  2 1963  2 1973  2 1990  4 1991  3 1992  4 1994  1 1995
 738  I_PSI_J                   5 - VECTOR             DOUBLE, ALIGNED, STATIC   XREF: 0 0738  2 0955  2 0961  2 1771
                                                       4 2094  4 2252
 745  I_PSI_J_DOT               5 - VECTOR             DOUBLE, ALIGNED, STATIC   XREF: 0 0745  4 1709  2 1728
 766  I_PSI_J_FLAG              BIT(1)                 ALIGNED, STATIC   XREF: 0 0766  2 1702  2 1708  2 1727  2 1769
                                                       4 1911  4 2248  4 2254
 736  I_PSI_PSI                 5 X 5 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0736  2 0880  2 1776  4 2095
                                                       4 2262  6 2273
 746  I_PSI_PSI_DOT             5 X 5 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0746  4 1713  2 1730
 767  I_PSI_PSI_FLAG            BIT(1)                 ALIGNED, STATIC   XREF: 0 0767  2 1702  2 1710  2 1729  2 1773
                                                       2 1788  2 1807  2 1817  2 1831  4 1912  4 2257  4 2265
1878  I_RCP                     INTEGER                SINGLE, ALIGNED, AUTOMATIC        XREF: 0 1878  4 2342  1 2343
                                                       1 2344  1 2346  1 2347  1 2348  1 2352  1 2353  1 2355  1 2359
                                                       1 2360  1 2363  1 2364  4 2456  2 2457  2 2459  1 2460  1 2465
                                                       1 2466  1 2468  1 2473
 988  I_RK                      INTEGER                SINGLE, ALIGNED, STATIC   XREF: 0 0988  2 1157  1 1165  1 1718
                                                       1 1720  1 1722  1 1724  1 1728  1 1730  4 1738  1 1739  4 1748
                                                       1 1749  4 1752  1 1754  4 1758  1 1760  4 1764  1 1765  4 1770
                                                       1 1771  4 1774  2 1775  1 1776  4 1787  2 1789  1 1792  1 1793
                                                       1 1794  4 1806  2 1808  1 1811  1 1812  4 1816  2 1818  1 1821
                                                       1 1822  4 1830  2 1832  1 1835  1 1836  4 1842  1 1843
 228  I_SEARCH                  INTEGER                SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0228  4 0314  1 0315
                                                       2 0317  4 0327  1 0328  2 0330  4 0418  1 0419  2 0421  4 0445
                                                       1 0446  2 0448
 776  I_STORE                   INTEGER                SINGLE, ALIGNED, STATIC   XREF: 0 0776  4 2167  1 2168  4 2178
                                                       1 2180  4 2191  1 2193  4 2212  1 2213  4 2251  1 2252  4 2260
                                                       2 2261  1 2262  4 2271  2 2272  1 2273  4 2339  1 2340  4 2368
                                                       1 2369  1 2370  1 2373  1 2374  1 2375  1 2376  1 2378  1 2380
                                                       1 2381  1 2382  4 2541  2 2542  3 2545  4 2549  2 2550  3 2552
                                                       4 2555  2 2556  3 2558  4 2562  1 2563  1 2564  1 2565  1 2566
 121  I_TIME                    INTEGER                SINGLE, ALIGNED, STATIC   XREF: 0 0121  1 0513  1 0572  1 0573
                                                       1 0574  1 0577  1 0578  1 0698  1 0839  1 0842  1 0843  1 0844
                                                       1 0845  1 0846  2 0852  6 1186  6 1187  2 1193  1 1195  2 1196
                                                       2 1201  2 1203  2 1211  2 1220  1 1222  1 1223  1 1230  1 1233
                                                       1 1355  1 1357  1 1360  1 1496  1 1497  1 1499  1 1500  1 1507
                                                       1 1555  1 1556  1 1557  1 1558  1 1561  1 1562  1 1563  1 1564
                                                       1 1604  1 1605  1 1612  2 1615  1 1618  1 1619  1 1621  1 1624
                                                       2 1657  2 1660  1 1661  1 1666  1 1739  6 1841  1 1843  1 1846
                                                       2 1847  1 1848  2 1857  2 1859  2 1863  2 1865  4 1915  1 1939
                                                       1 1940  1 1941  2 1946  1 1948  6 1950  1 1952  1 1966  1 1969
                                                       1 1970  1 1971  1 1975  6 1981  1 1985  1 1986  1 1992  2 1994
                                                       2 1995  2 2004  2 2009  2 2013  2 2018  2 2021  2 2026  2 2028
                                                       2 2033  1 2039  1 2040  1 2041  1 2042  1 2043  4 2053  2 2150
                                                       4 2151  2 2153  2 2155  2 2157  4 2172  4 2185  4 2198  6 2202
                                                       2 2203  2 2205  4 2207  2 2216  4 2221  2 2228  4 2229  2 2231
                                                       2 2233  2 2235  4 2246  4 2255  2 2256  4 2280  2 2284  2 2287
                                                       1 2293  1 2294  2 2299  2 2305  2 2308  1 2309  2 2310  1 2311
                                                       1 2313  1 2314  1 2322  4 2433
```

```
 DCL  NAME                     TYPE                 ATTRIBUTES & CROSS REFERENCE

 783  I_TIME_STORE             INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0783  2 1206  4 2146  2 2151
                                                    2 2172  2 2185  2 2198  2 2207  4 2216  2 2217  4 2222  2 2229
                                                    2 2246  2 2255  4 2266  2 2267
1881  I_U                      INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 1881  4 2050  2 2051  2 2053
                                                    2 2058  1 2062  2 2063  1 2064  4 2430  2 2431  2 2433  1 2438
                                                    4 2443  2 2444  2 2446  2 2449  2 2451  2 2452  2 2453  2 2456
                                                    1 2460  1 2465  1 2466  4 2488  2 2489  1 2490  4 2492  2 2493
                                                    2 2495  1 2496  1 2497  4 2501  2 2502  3 2503  3 2504
 862  I_VEC_1                  5 - VECTOR           DOUBLE, ALIGNED, STATIC   XREF: 0 0862  4 0953  2 0961  2 0967
                                                    2 0969
 863  I_VEC_2                  5 - VECTOR           DOUBLE, ALIGNED, STATIC   XREF: 0 0863  4 0955  2 0961  2 0967
                                                    2 0969
 864  I_VEC_3                  5 - VECTOR           DOUBLE, ALIGNED, STATIC   XREF: 0 0864  4 0954  2 0955  2 0961
 243  I_ZLD                    INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0243  4 0366  1 0367
                                                    1 0369  1 0370  1 0371  1 0372
 780  IIT                      INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0780  4 2544  1 2545  4 2582
                                                    2 2583
 870  IN_I                     INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0370  4 0885  1 0887
                                                    1 0889  2 0890  4 0897  1 0898  1 0899  1 0900  4 0902  1 0906
                                                    1 0907  4 0909  2 0910  1 0911  4 0913  2 0914  1 0917  4 0928
                                                    2 0929  1 0932  1 0933  4 0937  1 0938  1 0939  1 0940
 871  IN_J                     INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0871  4 0886  1 0887
                                                    1 0889  2 0891  4 0895  1 0899  1 0900  4 0904  1 0905  1 0906
                                                    1 0907  4 0915  2 0916  1 0917  4 0920  2 0921  1 0922  4 0930
                                                    1 0931  1 0932  1 0933  4 0935  2 0936  1 0939  1 0940
 872  IN_K                     INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0872  4 0881  3 0882
                                                    3 0883  1 0884  2 0885  2 0886  1 0890  1 0891  1 0895  3 0896
                                                    1 0898  1 0899  1 0902  3 0903  1 0905  1 0906  2 0910  1 0911
                                                    2 0914  2 0916  1 0917  2 0921  1 0922  1 0924  2 0926  6 0927
                                                    1 0928  2 0929  1 0931  1 0932  1 0935  2 0936  1 0938  1 0939
 787  INTEG_L                  SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0787  2 1741  4 1845  4 1927
                                                    4 1957  2 2031  2 2507  2 2573  2 2600  2 2601
 779  ITER_FLAG                BIT(1)               ALIGNED, STATIC   XREF: 0 0779  4 2579  2 2582  4 2603  4 2615
  71  ITERATION                INTEGER              SINGLE, ALIGNED   XREF: 0 0071  2 0943  2 2063  2 2410  2 2422
                                                    4 2540  3 2542  6 2543  1 2545  4 2548  3 2550  6 2551  1 2552
                                                    4 2554  3 2556  6 2557  1 2558  4 2561  2 2581  6 2590
1870  ITERATION_DRIVER         PROCEDURE              XREF: 0 1870  2 2573  2 2595
 869  J_DU                     INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0869  4 0963  1 0964  2 0965
                                                    1 0967
 866  J_MAT                    2 X 5 MATRIX         DOUBLE, ALIGNED, STATIC   XREF: 0 0866  4 0964  2 0967
1883  J_OMEGA                  INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 1883  4 1960  4 1932  2 1983
                                                    1 1935  1 1986
1879  J_RCP                    INTEGER              SINGLE, ALIGNED, AUTOMATIC        XREF: 0 1879  4 2459  1 2469
                                                    1 2470  1 2474  1 2475
 989  J_RK                     INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0989  1 1720  1 1722  1 1730
                                                    4 1753  1 1754  4 1759  1 1760  4 1775  1 1776  4 1790  1 1792
                                                    1 1793  1 1794  4 1809  1 1811  1 1812  4 1819  1 1821  1 1822
                                                    4 1833  1 1835  1 1836
  78  J_SCALE_FACTOR           SCALAR               DOUBLE, ALIGNED, INITIAL  XREF: 0 0078  2 2075
 777  J_STORE                  INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0777  4 2179  1 2180  4 2192
                                                    1 2193  4 2261  1 2262  4 2272  1 2273
 785  J_TIME                   INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 0785  1 0855  1 0856  4 1188
                                                    1 1618  1 1619  1 1620  1 1622  1 1683  1 1685  1 1687  1 1689
                                                    1 1691  1 1696  1 1698  1 1700  2 1704  1 1707  1 1709  1 1712
                                                    1 1749  1 1754  4 2080  1 2097  1 2098  1 2099  1 2100  1 2101
```

```
 DCL NAME                       TYPE                    ATTRIBUTES & CROSS REFERENCE

                                                        1 2102  1 2103  1 2104  1 2105  1 2106  1 2107  1 2108  1 2109
                                                        1 2110  1 2111  1 2112  1 2113  1 2114  4 2152  1 2168  4 2173
                                                        1 2180  4 2166  4 2199  4 2203  4 2208  4 2230  4 2247  4 2256
                                                        4 2277  1 2325  1 2326
1880 J_TIME_STORE               INTEGER                 SINGLE, ALIGNED, STATIC   XREF: 0 1880  4 2147  2 2152  2 2173
                                                        2 2186  2 2199  2 2208  4 2217  4 2223  2 2230  2 2247  2 2256
                                                        4 2267
1882 J_U                        INTEGER                 SINGLE, ALIGNED, AUTOMATIC        XREF: 0 1082  4 2446  1 2469
                                                        1 2470  1 2474  1 2475  4 2464  1 2485  1 2486
 741 K                          7 X 10 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0741  4 1573  4 1604  4 1605
                                                        4 1606  4 1607  4 1608  2 1631  2 1633  2 1643  2 1650  2 1655
1053 K_RK                       INTEGER                 SINGLE, ALIGNED, STATIC   XREF: 0 1053  4 1173  1 1175  1 1177
 982 KA                         7 X 10 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0982  4 1631  2 1603  2 1696
 983 KB                         7 X 10 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0983  4 1639  2 1695
 984 KC                         7 X 10 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0984  4 1643  2 1687  2 1698
 985 KD                         7 X 10 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0985  4 1650  2 1689
 986 KE                         7 X 10 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0986  4 1655  2 1691  2 1700
1051 KEEP_MASS                  SCALAR                  DOUBLE, ALIGNED, STATIC   XREF: 0 1051  4 1222  2 1661
 972 K0                         10 X 5 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0972  4 1792  2 1794  2 1812
                                                        2 1836
 973 K1                         10 X 5 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0973  4 1811  2 1812  2 1822
                                                        2 1836
 974 K2                         10 X 5 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0974  4 1821  2 1822  2 1036
 975 K3                         10 X 5 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0975  4 1835  2 1836
 200 L_BETA                     SCALAR                  DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0200  4 0620  2 0624
1048 L_CONST                    SCALAR                  DOUBLE, ALIGNED, STATIC   XREF: 0 1048  4 1216  2 1227  2 1379
                                                        2 1470  2 1624
  21 L_D_SCALE_FACTOR           SCALAR                  DOUBLE, ALIGNED, CONSTANT         XREF: 0 0021  2 0392  2 0478
  73 L_FILE                     INTEGER                 SINGLE, ALIGNED   XREF: 0 0073  4 2501  2 2532
 788 L_FINAL                    SCALAR                  DOUBLE, ALIGNED, STATIC   XREF: 0 0788  4 1998  4 2000  6 2002
                                                        2 2100
1054 L_RK                       INTEGER                 SINGLE, ALIGNED, AUTOMATIC        XREF: 0 1054  4 1176  1 1177
 996 L_SUB_P                    10 - VECTOR             DOUBLE, ALIGNED, STATIC   XREF: 0 0996  4 1213  4 1470  2 1632
                                                        2 1644  2 1656
 997 L_SUB_P_1                  10 - VECTOR             DOUBLE, ALIGNED, STATIC   XREF: 0 0997  4 1632  2 1696
 998 L_SUB_P_3                  10 - VECTOR             DOUBLE, ALIGNED, STATIC   XREF: 0 0998  4 1644  2 1698
 999 L_SUB_P_5                  10 - VECTOR             DOUBLE, ALIGNED, STATIC   XREF: 0 0999  4 1656  2 1700
1150 L_SUB_U                    2 - VECTOR              DOUBLE, ALIGNED, STATIC   XREF: 0 1150  4 1624  4 1625  2 1633
                                                        2 1645  2 1658
 811 L_SUB_U_L                  2 - VECTOR              DOUBLE, ALIGNED, STATIC   XREF: 0 0811  2 0858  4 1658
 809 L_SUB_U_1                  2 - VECTOR              DOUBLE, ALIGNED, STATIC   XREF: 0 0809  2 0855  4 1633
 810 L_SUB_U_3                  2 - VECTOR              DOUBLE, ALIGNED, STATIC   XREF: 0 0810  2 0856  4 1645
 995 L_SUB_X                    7 - VECTOR              DOUBLE, ALIGNED, STATIC   XREF: 0 0995  4 1212  4 1227  4 1385
                                                        4 1386  4 1387  2 1666
 134 L_TIME                     INTEGER                 SINGLE, ALIGNED, STATIC   XREF: 0 0134  1 0581  1 0582  4 1196
                                                        2 1197  4 1198  4 1211  1 1362  1 1363  1 1412  1 1476  1 1500
                                                        4 2085  4 2308
 727 L_X_STORE                  7 - VECTOR ARRAY        ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0727  2 1666
                                                        4 1933  4 1939  4 1940  4 1941  4 1966  4 1969  4 1970  4 1971
                                                        4 2513  2 2583
 748 LAMBDA                     7 - VECTOR ARRAY        ARRAY(1001), DOUBLE, ALIGNED, STATIC      XREF: 0 0748  2 0855
                                                        2 0856  2 0858  2 1696  2 1698  2 1700  2 1749  4 2097  4 2098
                                                        4 2099  4 2100  4 2101  4 2102  4 2103  4 2168  2 2325
 742 LAMBDA_DOT                 7 - VECTOR              DOUBLE, ALIGNED, STATIC   XREF: 0 0742  4 1666  2 1718
 761 LAMBDA_FLAG                BIT(1)                  ALIGNED, STATIC   XREF: 0 0761  2 1172  2 1174  2 1185  2 1190
```

```
 DCL  NAME                       TYPE                ATTRIBUTES & CROSS REFERENCE

                                                     2 1665  2 1717  2 1744  2 1747  2 1825  4 1906  4 2162  4 2171
1049  LAMBDA_HOLD                7 - VECTOR          DOUBLE, ALIGNED, STATIC   XREF: 0 1049  4 1175  2 1666
1895  LI_FLAG                    BIT(1)              ALIGNED, AUTOMATIC        XREF: 0 1895  4 2455  2 2456  4 2462
 144  LIFT                       SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0144  4 0692  2 0696  2 0697
                                                     2 1356  2 1357  2 1358
1893  LI4                        INTEGER             SINGLE, ALIGNED, AUTOMATIC      XREF: 0 1893  4 2447  2 2448
                                                     2 2449  2 2450  2 2452  4 2453  2 2455
1894  LI5                        INTEGER             SINGLE, ALIGNED, AUTOMATIC      XREF: 0 1894  4 2450  4 2451
                                                     2 2456
 251  LOW_A                      INTEGER             SINGLE, ALIGNED, AUTOMATIC      XREF: 0 0251  4 0322  1 0337
                                                     1 0338  1 0341  1 0347  1 0350  1 0352  1 0353  1 0354  1 0355
 509  LOW_ALT                    INTEGER             SINGLE, ALIGNED, AUTOMATIC      XREF: 0 0509  4 0514  2 0515
                                                     2 0522  2 0528  3 0529  1 0530  1 0531  3 0532
 405  LOW_ANGLE                  INTEGER             SINGLE, ALIGNED, AUTOMATIC      XREF: 0 0405  4 0426  1 0429
                                                     1 0430  1 0436  1 0437  1 0454  1 0455  1 0460  1 0463  1 0464
                                                     1 0467  1 0468  1 0469  1 0470
 230  LOW_CD                     SCALAR              DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0230  4 0338  2 0344
                                                     2 0350  4 0353  2 0357  4 0430  4 0437  4 0455  2 0462  2 0464
                                                     4 0468  2 0472
 244  LOW_CD0                    SCALAR              DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0244  4 0369  2 0376
                                                     2 0377
 232  LOW_CL                     SCALAR              DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0232  4 0337  2 0345
                                                     2 0347  4 0352  2 0356  4 0429  4 0436  4 0454  2 0461  2 0463
                                                     4 0467  2 0471
 227  LOW_M                      INTEGER             SINGLE, ALIGNED, AUTOMATIC      XREF: 0 0227  4 0290  1 0291
                                                     1 0292  4 0335  1 0336  1 0337  1 0338  1 0339  1 0340  1 0352
                                                     1 0353  1 0356  1 0357  4 0452  1 0453  1 0454  1 0455  1 0456
                                                     1 0457  1 0467  1 0468  1 0471  1 0472
 245  LOW_M_S                    SCALAR              DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0245  4 0370  2 0376
                                                     2 0377
 205  LOW_M_2                    SCALAR              DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0205  4 0647  2 0648
                                                     2 0658  4 0663
 749  M                          5 X 13 MATRIX       DOUBLE, ALIGNED, STATIC   XREF: 0 0749  2 0880  2 0954  2 0956
                                                     4 2328  4 2333
 233  M_FIND                     BIT(1)              ALIGNED, AUTOMATIC        XREF: 0 0233  4 0326  2 0327  4 0331
                                                     4 0444  2 0445  4 0449
1876  M_VEC                      5 - VECTOR          DOUBLE, ALIGNED, STATIC   XREF: 0 1876  4 2326  2 2328
  23  M_ZLD                      SCALAR ARRAY        ARRAY(16), DOUBLE, ALIGNED, CONSTANT      XREF: 0 0023  2 0359
                                                     2 0362  2 0367  2 0370  2 0372
 987  M_1                        5 X 2 MATRIX        DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0987  4 1712  2 1713
 207  M_2_FLAG                   BIT(1)              ALIGNED, AUTOMATIC        XREF: 0 0207  4 0649  2 0650  4 0654
 771  MASS_FLOW_FINAL_STEP       SCALAR              DOUBLE, ALIGNED, STATIC   XREF: 0 0771  4 2084  2 2100
2535  MAX_AA_INDEX               INTEGER             SINGLE, ALIGNED, STATIC, INITIAL  XREF: 0 2535  2 2550
  42  MAX_ALT                    INTEGER             SINGLE, ALIGNED, CONSTANT       XREF: 0 0042  2 0043  2 0044
                                                     2 0515  1 0518  1 0519
 876  MAX_B                      SCALAR              DOUBLE, ALIGNED, AUTOMATIC      XREF: 0 0376  4 0384  2 0387
                                                     4 0839
  64  MAX_BETA_CYCLES            INTEGER             SINGLE, ALIGNED, CONSTANT       XREF: 0 0064  2 0621  2 0657
2529  MAX_CP_INDEX               INTEGER             SINGLE, ALIGNED, STATIC, INITIAL  XREF: 0 2529  2 2556
  66  MAX_CUTOFF_ITERATIONS      INTEGER             SINGLE, ALIGNED, CONSTANT       XREF: 0 0066  2 1956  2 1963
                                                     2 1978
  28  MAX_FS_AR_INDEX            INTEGER             SINGLE, ALIGNED, CONSTANT       XREF: 0 0028  2 0030  2 0031
                                                     2 0033  1 0310  2 0311  2 0314  1 0342  1 0348
  29  MAX_FS_M_INDEX             INTEGER             SINGLE, ALIGNED, CONSTANT       XREF: 0 0029  2 0030  2 0031
                                                     2 0032  1 0323  2 0324  2 0327
```

```
 DCL NAME                      TYPE              ATTRIBUTES & CROSS REFERENCE

  95 MAX_ITERATIONS            INTEGER           SINGLE, ALIGNED, INITIAL  XREF: 0 0095  2 2582  2 2583
 183 MAX_ROCKET_THRUST         SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0183  2 0410  4 0551  2 0685
  36 MAX_SS_ANGLE_INDEX        INTEGER           SINGLE, ALIGNED, CONSTANT        XREF: 0 0036  2 0038  2 0039
                                                 2 0041  1 0414  2 0415  1 0418
  37 MAX_SS_M_INDEX            INTEGER           SINGLE, ALIGNED, CONSTANT        XREF: 0 0037  2 0035  2 0039
                                                 2 0040  1 0434  1 0436  1 0437  1 0438  1 0439  1 0442  2 0445
  91 MAX_STEP_I                INTEGER           SINGLE, ALIGNED, INITIAL  XREF: 0 0091  2 2074  2 2509
2532 MAX_W_INDEX               INTEGER           SINGLE, ALIGNED, STATIC, INITIAL  XREF: 0 2532  2 2542
  22 MAX_ZLD_INDEX             INTEGER           SINGLE, ALIGNED, CONSTANT        XREF: 0 0022  2 0023  2 0024
                                                 1 0362  1 0363  2 0366
  27 MCR_C                     SCALAR ARRAY      ARRAY(5), DOUBLE, ALIGNED, CONSTANT      XREF: 0 0027  2 0280
                                                 2 0291  2 0292
 249 MCR_FLAG                  BIT(1)            ALIGNED, AUTOMATIC        XREF: 0 0249  4 0282  2 0283  4 0286
  26 MCR_M                     SCALAR ARRAY      ARRAY(5), DOUBLE, ALIGNED, CONSTANT      XREF: 0 0026  2 0279
                                                 2 0284  2 0291  2 0292
  25 MCR_MAX_INDEX             INTEGER           SINGLE, ALIGNED, CONSTANT        XREF: 0 0025  2 0026  2 0027
                                                 1 0279  1 0280  2 0283
 146 MD_MASS                   SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0146  4 0693  6 0700  2 0713
                                                 2 0715  2 1227  2 1379  2 1468
 209 MID_M_2                   SCALAR            DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0209  4 0658  2 0659
                                                 2 0662  2 0663  2 0665
 210 MID_NU                    SCALAR            DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0210  4 0660  2 0661
  92 MIN_COST_RQMT             SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0092  2 2602
  93 MIN_PSI_RQMT              SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0093  2 2602
  90 MIN_PSI_THRESHOLD         SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0090  2 2507
  14 MIN_SCRJ_MASS_PER_FT      SCALAR            DOUBLE, ALIGNED, CONSTANT        XREF: 0 0014  2 0261  2 0262
                                                 2 2134
 172 MODEL_DRIVER              PROCEDURE           XREF: 0 0172  2 1214  2 1930  2 1935  2 1961  2 2086  2 2291
                                                 2 2312
 130 M0                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0130  2 0323  2 0326  2 0336
                                                 2 0359  2 0362  2 0367  2 0376  2 0427  2 0434  2 0442  2 0446
                                                 2 0453  4 0539  2 0595  2 0601  2 0605  2 0617  2 1229  2 1232
                                                 2 1233  2 1234  2 1255  2 1259  2 1264  2 1268  2 1273  2 1274
                                                 2 1276  2 1282  2 1283  2 1285
 407 M0_FLAG                   BIT(1)            ALIGNED, AUTOMATIC        XREF: 0 0407  4 0441  4 0458  2 0465
 135 M0_2                      SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0135  4 0605  2 0609  2 0611
                                                 2 0623  2 0630  2 0631  2 0632  2 0635  2 0637  2 0638  2 0644
                                                 2 0647  2 0656  4 1229  2 1254  2 1255  2 1258  2 1259  2 1262
                                                 2 1266  2 1273  2 1274  2 1276  2 1282  2 1285  2 1412  2 1413
                                                 2 1415  2 1417
 186 M1                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0186  4 0601  4 0670  2 0672
                                                 2 0674
  57 M1_FINAL                  SCALAR            DOUBLE, ALIGNED, CONSTANT        XREF: 0 0057  2 1924
— 772 M1_FLOW_FINAL_STEP        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0772  4 2081  2 2112
 202 M1_2                      SCALAR            DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0202  4 0630  4 0636
                                                 4 0665  2 0666  2 0670
 137 M2                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0137  2 0277  2 0279  2 0284
                                                 2 0291  2 0301  2 0302  4 0584  4 0674  2 1246  2 1281  2 1282
                                                 2 1287  2 1288  2 1313  2 1314  2 1315  2 1414  2 1415  2 1422
                                                 2 1423  2 1424
  58 M2_FINAL                  SCALAR            DOUBLE, ALIGNED, CONSTANT        XREF: 0 0058  2 1925
— 773 M2_FLOW_FINAL_STEP        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0773  4 2082  2 2113
—1124 M2_2                      SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1124  2 1279  4 1281  2 1282
                                                 2 1263  2 1287  2 1288  2 1421  2 1422  2 1423  2 1424
  59 M3_FINAL                  SCALAR            DOUBLE, ALIGNED, CONSTANT        XREF: 0 0059  2 1926
```

```
 DCL NAME                      TYPE                 ATTRIBUTES & CROSS REFERENCE

 774 M3_FLOW_FINAL_STEP        SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0774  4 2003  2 2114
 112 N_AERO                    SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0112  4 0697  2 0713  2 0714
                                                    2 0715  2 1335  2 1386  2 1387  2 1468  2 1480  2 1505  2 1509
                                                    2 1513  2 1624
  67 NEG_TIME_STEP             SCALAR ARRAY         ARRAY(3), DOUBLE, ALIGNED, INITIAL        XREF: 0 0067  2 1858
                                                    2 1864  2 2344  6 2347  2 2352  4 2359  4 2363  4 2375  4 2381
                                                    6 2388  4 2396  6 2401  6 2408  4 2413  6 2420  4 2425  4 2519
                                                    2 2594
 726 NET_MASS                  SCALAR ARRAY         ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0726  2 1222
                                                    6 1223  2 1507  2 1555  2 1556  2 1557  2 1558  2 1561  2 1562
                                                    2 1563  2 1564  2 1604  2 1605  2 1612  2 1613  2 1619  2 1624
                                                    4 1661  4 1646  6 1848  4 1931  2 2004  2 2096  2 2111  2 2293
                                                    2 2294  2 2309  4 2311  2 2313  2 2314  4 2322  4 2512  4 2579
                                                    2 2587  4 2624
 110 NET_R_FORCE               SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0110  4 0713  2 0844  2 1558
                                                    2 1604  2 1612  2 2111  2 2295  2 2313
 111 NET_THETA_FORCE           SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0111  4 0714  2 0846  2 1561
                                                    2 1564  2 1605  2 1611  2 2096  2 2294  2 2314
1154 NET_THRUST                SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 1154  4 1215  2 1468  2 1477
                                                    2 1506  2 1510  2 1514  2 1624
 147 NET_X_FORCE               SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0147  4 0712  2 0713  2 0714
                                                    2 0715  2 1385  2 1386  2 1387
  60 NORM_TIME_STEP            SCALAR               DOUBLE, ALIGNED, CONSTANT             XREF: 0 0060  2 1853  2 1954
                                                    2 2060  2 2354  2 2359  2 2360  2 2363  2 2364  2 2375  2 2376
                                                    2 2381  2 2382  2 2396  2 2397  2 2413  2 2414  2 2425  2 2426
                                                    2 2448  2 2504  2 2630  2 2633
 150 NORMAL_SHOCK_FLAG         BIT(1)               ALIGNED, STATIC   XREF: 0 0150  4 0556  4 0635  2 1271  4 2443
 174 NOSE_ANGLE                SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0174  2 0266  2 0267  4 0542
                                                    2 0606
   6 NOZZLE_ANGLE              SCALAR               DOUBLE, ALIGNED, CONSTANT             XREF: 0 0006  2 0266  2 0267
                                                    2 2116  2 2119  2 2129  2 2130  2 2133
  48 NUM_CONSTANT_PARAMETERS   INTEGER              SINGLE, ALIGNED, CONSTANT             XREF: 0 0048  2 0061  2 0075
                                                    2 0728  2 0731  2 0734  2 0741  2 0744  2 0747  2 0749  2 0750
                                                    2 0801  2 0865  2 0982  2 0983  2 0984  2 0985  2 0986  2 0996
                                                    2 0997  2 0998  2 0999  2 1000  2 1001  2 1002  2 1003  2 1004
                                                    2 1005  2 1006  2 1007  2 1008  2 1009  2 1125  2 1126  2 1127
                                                    2 1128  2 1129  2 1130  2 1131  2 1132  2 1133  2 1400  2 1574
                                                    2 1758  2 1764  2 2090  2 2188  2 2191  2 2212  1 2325  1 2328
                                                    2 2332  2 2339  1 2343  1 2352  1 2353  2 2536
  51 NUM_CONSTRAINTS           INTEGER              SINGLE, ALIGNED, CONSTANT             XREF: 0 0051  2 0062  2 0724
                                                    2 0725  2 0733  2 0734  2 0735  2 0736  2 0738  2 0743  2 0744
                                                    2 0745  2 0746  2 0749  2 0851  2 0852  2 0853  2 0864  2 0865
                                                    2 0866  2 0874  2 0875  2 0881  2 0855  2 0886  2 0897  2 0904
                                                    2 0909  2 0913  2 0915  2 0920  2 0930  2 0937  2 0972  2 0973
                                                    2 0974  2 0975  2 0976  2 0987  2 1050  2 1176  2 1753  2 1759
                                                    2 1770  2 1774  2 1775  2 1076  2 1922  2 2175  2 2179  2 2192
                                                    2 2249  2 2251  2 2258  2 2260  2 2261  2 2271  2 2327  2 2331
  49 NUM_CONTROLS              INTEGER              SINGLE, ALIGNED, CONSTANT             XREF: 0 0049  2 0069  2 0076
                                                    2 0722  2 0723  2 0729  2 0730  2 0732  2 0796  2 0804  2 0809
                                                    2 0810  2 0811  2 0856  2 0987  2 1144  2 1145  2 1146  2 1147
                                                    2 1148  2 1149  2 1150  2 1617  2 1877  2 1899  2 1901  2 1903
                                                    2 2531  2 2544
   2 NUM_STATES                INTEGER              SINGLE, ALIGNED, CONSTANT             XREF: 0 0002  2 0098  2 0727
                                                    2 0732  2 0733  2 0739  2 0740  2 0741  2 0742  2 0743  2 0748
                                                    2 0798  2 0800  2 0977  2 0978  2 0979  2 0980  2 0981  2 0982
```

```
DCL  NAME                     TYPE                 ATTRIBUTES & CROSS REFERENCE

                                                   2 0983  2 0984  2 0985  2 0986  2 0995  2 1049  2 1050  2 1157
                                                   2 1173  2 1738  1 1741  2 1748  2 1752  2 1842  1 1845  2 1875
                                                   2 1913  2 2163  2 2167  2 2178
  50 NUM_TRANS_PTS            INTEGER              SINGLE, ALIGNED, CONSTANT        XREF: 0 0050  2 0061  2 0067
                                                   2 0068  2 0070  2 0731  2 0737  2 0749  2 0750  2 0803  2 0805
                                                   2 0806  2 0818  2 0865  2 1202  2 1854  2 2276  2 2342  2 2368
                                                   2 2447  2 2537
 203 NU0                      SCALAR               DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0203  4 0645  2 0646
 204 NU1                      SCALAR               DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0204  4 0646  2 0653
                                                   2 0661
 149 OBLIQUE_SHOCK_FLAG       BIT(1)               ALIGNED, STATIC   XREF: 0 0149  4 0555  4 0615  2 1247  2 1410
                                                   4 2647
 792 OIT_FLAG                 BIT(1)               ALIGNED, STATIC   XREF: 0 0792  4 1199  4 1207  2 1210  2 1220
                                                   2 1660
2537 OIT_INIT                 INTEGER ARRAY        ARRAY(3), SINGLE, ALIGNED, STATIC, INITIAL       XREF: 0 2537
                                                   2 2568
1898 OLD_FINAL_STEP           INTEGER              SINGLE, ALIGNED, STATIC   XREF: 0 1898  4 2048  2 2440  4 2441
                                                   2 2442  2 2443  2 2449  2 2451  2 2484
1896 OLD_SC3                  SCALAR               DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 1896  4 2454  2 2461
                                                   4 2479
  70 OMEGA_I_TIME             INTEGER ARRAY        ARRAY(3), SINGLE, ALIGNED        XREF: 0 0070  2 1203  2 1206
                                                   2 1220  2 1660  2 1847  2 1857  2 1859  2 1863  2 1865  2 2013
                                                   2 2018  2 2021  2 2026  2 2028  2 2153  2 2155  2 2157  2 2231
                                                   2 2233  2 2235  2 2277  2 2273  2 2280  2 2284  2 2287  2 2299
                                                   2 2305  2 2310  6 2369  2 2370  4 2373  4 2374  2 2378  4 2380
                                                   2 2385  6 2387  2 2391  2 2393  6 2395  6 2400  2 2405  6 2407
                                                   2 2410  6 2412  2 2417  6 2419  2 2422  6 2424  2 2495  4 2517
                                                   4 2568  2 2591
2524 OPTIMIZATION             PROCEDURE             XREF: 0 2524  2 2641
1071 OSC1                     SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 1071  4 1254  2 1258  2 1259
                                                   4 1379  2 1385  2 1386  2 1387
 782 OVER_ITER_FLAG           BIT(1)               ALIGNED, STATIC   XREF: 0 0782  4 2580  2 2582  4 2584
 784 OVER_STEP                BIT(1)               ALIGNED, STATIC   XREF: 0 0784  2 1934  4 2034  2 2037  2 2073
                                                   2 2575  2 2597  4 2631
  75 P                        10 - VECTOR          DOUBLE, ALIGNED   XREF: 0 0075  2 0542  2 0543  2 0544  2 0545
                                                   2 0546  2 0547  2 0548  2 0549  2 0550  2 0551  2 1402  2 1412
                                                   2 1434  2 1435  2 1439  2 1440  2 1443  2 1445  2 1447  2 1450
                                                   2 1451  2 1454  2 1456  2 1521  2 1533  2 2043  2 2116  2 2117
                                                   2 2118  2 2119  2 2122  2 2123  2 2124  2 2125  2 2126  2 2127
                                                   2 2129  2 2131  2 2132  2 2133  2 2134  2 2135  2 2137  2 2138
                                                   2 2139  2 2140  2 2141  2 2142  2 2143  6 2340  4 2514  4 2560
                                                   2 2572  2 2589
 113 P_AERO                   SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0113  4 0696  2 0712  2 1468
                                                   2 1481  2 1506  2 1510  2 1514  2 1624
2536 P_INITIAL                10 - VECTOR          DOUBLE, ALIGNED, STATIC, INITIAL  XREF: 0 2536  2 2560
 721 PARTIAL_DERIV_FLAG       BIT(1)               ALIGNED, STATIC   XREF: 0 0721  2 1180  4 1182  4 1785  4 1805
                                                   4 1829  4 2638
1892 PAST_INTEG_L             SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 1892  4 1928  2 1957  4 2031
  68 POS_TIME_STEP            SCALAR ARRAY         ARRAY(3), DOUBLE, ALIGNED, INITIAL        XREF: 0 0068  2 1860
                                                   2 1866  2 2344  6 2346  2 2353  4 2360  4 2364  4 2376  4 2382
                                                   6 2389  4 2397  6 2402  6 2409  4 2414  6 2421  4 2426  4 2518
                                                   2 2593
 791 PRESENT_TIME_STEP        SCALAR               DOUBLE, ALIGNED, STATIC   XREF: 0 0791  4 1853  4 1858  4 1860
                                                   4 1864  4 1866  2 1944  2 1959  2 1980  2 2055  2 2219  2 2269
                                                   2 2435  4 2633
```

```
DCL  NAME               TYPE              ATTRIBUTES & CROSS REFERENCE

735  PSI                5 - VECTOR        DOUBLE, ALIGNED, STATIC   XREF: 0 0735  2 0953  4 2039  4 2040
                                          4 2041  4 2042  4 2043  2 2044  2 2079  4 2337  2 2507
 96  PSI_CHECK          SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0096  2 2604
 89  PSI_COST_MIX       SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0089  2 0960
778  PSI_MAG            SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0778  2 0946  4 2044  2 2507
                                          2 2574  2 2596  2 2602  2 2604
 79  PSI_SCALE_FACTOR   SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0079  2 2076
 65  PSI_START          SCALAR            DOUBLE, ALIGNED, CONSTANT        XREF: 0 0065  2 0944  2 0946
                                          2 0949  2 0950
868  PSI_VAL            SCALAR            DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0868  4 0946  2 0947
                                          4 0948  2 0949  2 0951
 62  PSI_WEIGHT         5 X 5 MATRIX      DOUBLE, ALIGNED, CONSTANT        XREF: 0 0062  2 2044  2 2507
184  P0                 SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0184  4 0538  2 0598  2 0631
                                          2 0637  2 0667
187  P1                 SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0187  4 0598  4 0631  4 0637
                                          4 0667  2 0671
 84  Q_D                SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0084  2 1160  2 1161  2 1936
                                          2 1938  2 1965  2 1968  2 1999  2 2000
217  R                  SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0217  2 0540  4 0572  2 0574
                                          2 0598
 97  R_CUTOFF_CHECK     SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0097  2 1948
197  R_NEW_BOUND        SCALAR            DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0197  4 0623  2 0624
 19  R_0                SCALAR            DOUBLE, ALIGNED, CONSTANT        XREF: 0 0019  2 0538  2 0539
                                          2 0671  2 0672  2 1231  2 1244
512  REL_RO             SCALAR            DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0512  4 0518  4 0525
                                          4 0533  2 0534  2 0537
122  RESHAPE_FLAG       BIT(1)            ALIGNED, STATIC   XREF: 0 0122  2 0258  2 0408  2 0496  4 0505
                                          2 0552  4 1929
769  RK_COLUMNS         INTEGER           SINGLE, ALIGNED, STATIC   XREF: 0 0769  2 1780  2 1790  2 1809
                                          2 1819  2 1833  4 1914  4 2149  4 2175  4 2210  4 2242  4 2258
990  RK_D_VAL           SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0990  4 1159  4 1161  6 1163
                                          4 1165  4 1718  4 1720  4 1722  4 1724  4 1726  4 1728  4 1730
                                          2 1792  2 1811  2 1821  2 1835
994  RK_DERIV           PROCEDURE          XREF: 0 0994  2 1791  2 1810  2 1820  2 1834
768  RK_ROWS            INTEGER           SINGLE, ALIGNED, STATIC   XREF: 0 0768  2 1780  2 1787  2 1806
                                          2 1816  2 1830  4 1913  4 2163  4 2189  4 2241  4 2249
991  RK_STEP            INTEGER           SINGLE, ALIGNED, STATIC   XREF: 0 0991  6 1160  2 1183  4 1779
724  RK_VAL_N           10 X 5 MATRIX     DOUBLE, ALIGNED, STATIC   XREF: 0 0724  2 1175  2 1177  4 1739
                                          4 1741  4 1749  4 1754  4 1760  4 1765  4 1768  4 1771  4 1776
                                          2 1793  6 1794  6 1812  6 1822
725  RK_VAL_N_PLUS_1    10 X 5 MATRIX     DOUBLE, ALIGNED, STATIC   XREF: 0 0725  4 1836  2 1843  2 1845
                                          2 2168  2 2180  2 2193  2 2213  2 2244  2 2252  2 2262
123  RO_0               SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0123  4 0537  2 0538  2 0592
                                          2 0599  2 1266  2 1269  2 1276  2 1277  2 1296  2 1356  2 1359
                                          2 1413  2 1939  2 1940  2 1941  2 1969  2 1970  2 1971
188  RO_1               SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0188  4 0599  4 0671  2 0676
124  RO_2               SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0124  2 0305  2 0308  4 0585
                                          4 0676  2 1269  2 1277  2 1285  2 1286  2 1288  2 1317  2 1321
                                          2 1322  2 1323  2 1333  2 1334  2 1335  2 1372  2 1373  2 1374
                                          2 1375  2 1376  2 1377  2 1424  2 1430  2 1431  2 1432  2 1433
                                          2 1434  2 1435  2 1457  2 1458
 13  ROCKET_ISP         SCALAR            DOUBLE, ALIGNED, CONSTANT        XREF: 0 0013  2 0711  2 1495
                                          2 1599
173  ROCKET_MAX_T       SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0173  4 0683  4 0685  2 0701
116  ROCKET_THRUST      SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0116  4 0701  2 0711  2 0712
```

```
 DCL  NAME                       TYPE            ATTRIBUTES & CROSS REFERENCE

                                                 2 1215  2 1495
    1 RUN_POOL                   COMPOOL         EXTERNAL, VERSION=1       XREF: 0 0001
  971 RUNGE_KUTTA                PROCEDURE        XREF: 0 0971  2 1945  2 1962  2 2166  2 2177  2 2190  2 2211
                                                 2 2243  2 2250  2 2259
  403 SCND_STAGE                 PROCEDURE        XREF: 0 0403  2 0500  2 0504
  141 SCRAMJET_ISP               SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 0141  4 0302  2 0308  2 0705
                                                 2 1330  2 1333  2 1334  2 1335  2 1432  2 1433  2 1489  2 1491
                                                 2 1492
  129 SCRAMJET_MASS              SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 0129  4 0260  2 0261  4 0262
                                                 6 0263  2 0274  2 2134  2 2137
  119 SCRAMJET_POWER             BIT(1)          ALIGNED, STATIC   XREF: 0 0119  2 0277  2 0306  2 1330  2 1432
                                                 2 1455  2 1489  2 1531  2 1590  4 1918  4 2016  4 2022  4 2029
                                                 4 2089  4 2154  4 2159  4 2227  4 2232  4 2237  4 2265  4 2286
                                                 2 2297  4 2302  4 2304  2 2317
  114 SCRAMJET_THRUST            SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 0114  4 0307  4 0308  4 0686
                                                 2 0705  2 0712  2 1215  2 1332  2 1333  2 1334  2 1335  2 1433
                                                 2 1456  2 1491  2 1492
  109 SCRJ_FUEL_FLOW             SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 0109  4 0705  4 0709  2 0848
                                                 2 2298  2 2318
  128 SCRJ_MASS_CAPTURE_RATIO    SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 0128  4 0280  4 0291  2 0294
                                                 4 0296  4 0300  2 0308  4 0690  2 1330  2 1332  2 1375  2 1376
                                                 2 1377  2 1432  2 1433  2 1435  2 1458
   10 SCRJ_MAX_FUEL_AIR_RATIO    SCALAR          DOUBLE, ALIGNED, CONSTANT        XREF: 0 0010  2 0308  2 1533
                                                 2 1591
 1886 SC1                        SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1886  4 1938  2 1939  2 1940
                                                 2 1941  4 1968  2 1969  2 1970  2 1971  4 1983  2 1984  2 1985
                                                 2 1986  4 2129  2 2132  2 2133  2 2137  2 2143  4 2343  2 2344
                                                 2 2346  2 2347  2 2351  2 2355  2 2357
 1887 SC2                        SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1887  4 1954  2 1984  2 1985
                                                 2 1986  4 2130  2 2131  2 2133  2 2137  2 2142  2 2143  4 2352
                                                 4 2353  2 2354  4 2356  2 2359  2 2360  2 2363  2 2364
 1888 SC3                        SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1888  4 2131  2 2133  2 2137
                                                 2 2143  4 2354  2 2355  2 2356  4 2460  2 2461  2 2463  2 2477
                                                 2 2479
 1889 SC4                        SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1889  4 2049  6 2055  6 2059
                                                 6 2060  2 2062  4 2132  2 2137  2 2142  2 2143  4 2429  6 2435
                                                 6 2437  2 2438  4 2448  2 2454  2 2463  4 2477
 1047 SG                         SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1047  4 1503  2 1504  2 1507
                                                 2 1578
  747 SGV_DOT                    10 - VECTOR     DOUBLE, ALIGNED, STATIC   XREF: 0 0747  4 1696  4 1698  4 1700
                                                 2 1724
  764 SGV_FLAG                   BIT(1)          ALIGNED, STATIC   XREF: 0 0764  2 1693  2 1723  2 1763  2 1799
                                                 4 1909  4 2209  4 2215
 1118 SIN_A                      SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1118  4 1362  2 1364  2 1365
                                                 2 1366  2 1367  2 1368  2 1369  2 1465  2 1466  2 1480  2 1481
 1069 SIN_B_T                    SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1069  4 1252  2 1256  2 1258
                                                 2 1259  2 1414  2 1415
 1066 SIN_BETA                   SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1066  4 1249  2 1251  2 1255
                                                 2 1258  2 1261  2 1262  2 1264  2 1266  2 1268  2 1413  2 1415
                                                 2 1416  2 1417
  212 SIN_BETA_THETA_MAX         SCALAR          DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0212  4 0609  2 0610
                                                 2 0611
 1046 SIN_DELTA                  SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 1046  4 1502  2 1507  2 1508
                                                 2 1511  2 1512  2 1515  2 1516  2 1578  2 1579
  165 SIN_VEHICLE_ANGLE          SCALAR          DOUBLE, ALIGNED, STATIC   XREF: 0 0165  4 0679  2 0713  2 0714
```

```
 DCL  NAME                       TYPE              ATTRIBUTES & CROSS REFERENCE

                                                   2 1611  2 1612  2 1613
1068  SIN_2_BETA                 SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1068  4 1251  2 1254  2 1255
                                                   2 1258  2 1259  2 1261  2 1262  2 1266  2 1412  2 1413  2 1415
                                                   2 1416  2 1417
 750  SMALL_G_VEC                13 - VECTOR       DOUBLE, ALIGNED, STATIC   XREF: 0 0750  2 0954  2 0961  2 0969
                                                   2 1765  4 2091  4 2213  4 2325
1057  SPEED_OF_SOUND_2           SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1057  4 1231  2 1232  2 1233
                                                   2 1234
1052  SR                         SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1052  4 1195  2 1232  2 1234
                                                   2 1355  2 1356  2 1358  2 1359  2 1361  2 1496  2 1498  2 1499
                                                   2 1500  2 1503  2 1507  2 1557  2 1561  2 1562  2 1563  2 1564
                                                   2 1605  2 1619
  41  SS_ANGLE_OF_ATTACK_VAL     SCALAR ARRAY      ARRAY(5), DOUBLE, ALIGNED, CONSTANT      XREF: 0 0041  2 0414
                                                   2 0419  2 0460  2 0463  2 0464
 164  SS_CD                      SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0164  4 0462  6 0480  2 0486
                                                   4 0490  2 1464
  39  SS_CD_MAT                  5 X 16 MATRIX     DOUBLE, ALIGNED, CONSTANT         XREF: 0 0039  2 0430  2 0432
                                                   2 0437  2 0439  2 0455  2 0457  2 0468  2 0470
 163  SS_CL                      SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0163  4 0461  6 0479  2 0485
                                                   2 1463
  38  SS_CL_MAT                  5 X 16 MATRIX     DOUBLE, ALIGNED, CONSTANT         XREF: 0 0038  2 0429  2 0431
                                                   2 0436  2 0438  2 0454  2 0456  2 0467  2 0469
 224  SS_DRAG                    SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0224  4 0465  2 0487  4 0489
                                                   2 0693
 120  SS_DRY_MASS                SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0120  4 0410  2 0693  2 0839
                                                   2 1846  2 1931
 108  SS_FUEL_FLOW               SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0108  4 0711  2 0849  2 2320
 223  SS_LIFT                    SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0223  4 0465  2 0692
  40  SS_M_VAL                   SCALAR ARRAY      ARRAY(16), DOUBLE, ALIGNED, CONSTANT     XREF: 0 0040  2 0427
                                                   2 0434  2 0442  2 0446  2 0453  2 0471  2 0472
 182  SS_MAX_FUEL_LOAD           SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0182  2 0410  2 0411  4 0550
 143  SS_PLANFORM_AREA           SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0143  4 0411  2 0485  2 0486
                                                   2 1356  2 1357  2 1358  2 1359  2 1360  2 1361  2 1478  2 1479
 104  STAGE_SEP                  BIT(1)            ALIGNED, STATIC   XREF: 0 0104  2 0498  2 0593  2 0682  2 0699
                                                   2 0702  2 0840  2 1242  2 1370  2 1403  2 1467  2 1482  2 1517
                                                   2 1575  2 1585  2 1602  2 1647  4 1916  2 2011  4 2015  4 2037
                                                   4 2160  4 2225  4 2238  4 2283  4 2301  2 2319
 993  START_RK_COLUMNS           INTEGER           SINGLE, ALIGNED, STATIC   XREF: 0 0993  4 1733  4 1789  2 1790
                                                   4 1808  2 1809  4 1818  2 1819  4 1832  2 1833
 836  STATE_DERIVS               PROCEDURE           XREF: 0 0836  2 1736
1905  STATE_INTEGRATION          PROCEDURE           XREF: 0 1905  2 2069  2 2506
 133  STATE_INTEGRATION_FLAG     BIT(1)            ALIGNED, STATIC   XREF: 0 0133  2 0575  2 1155  2 1734  2 1839
                                                   2 1855  4 1919  2 1934  4 2005  2 2007  4 2290  4 2292  4 2646
  63  STEP_DIM                   INTEGER           SINGLE, ALIGNED, CONSTANT         XREF: 0 0063  2 0069  2 0076
                                                   2 0077  2 0098  2 0099  2 0726  2 0727  2 0729  2 0730  2 0732
                                                   2 0733  2 0751  2 0752  2 0798  2 0799  2 0800  2 0802  2 0804
                                                   2 1877  2 2033  2 2050  2 2501  2 2525  2 2526  2 2541  2 2549
                                                   2 2555  2 2562  2 2622
 807  STEP_I                     INTEGER           SINGLE, ALIGNED, STATIC   XREF: 0 0807  2 0878  4 2074  2 2075
                                                   2 2076  2 2077  2 2509  4 2623
1900  STEP_I_FLAG                BIT(1)            ALIGNED, STATIC   XREF: 0 1900  4 2047  2 2074  4 2508  2 2509
  88  STEP_SCALE_J               SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0088  2 0959  4 2075
  87  STEP_SCALE_PSI             SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0087  2 0958  4 2076
 152  SUBSONIC_FLAG              BIT(1)            ALIGNED, STATIC   XREF: 0 0152  4 0558  4 0597  2 1290  4 2650
1041  S2                         SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1041  4 1332  2 1333  2 1334
```

```
 DCL NAME                      TYPE              ATTRIBUTES & CROSS REFERENCE

                                                  2 1335  4 1500  2 1501  2 1502
1042 S3                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1042  4 1496  2 1497  2 1498
                                                  2 1499
1043 S5                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1043  4 1505  2 1507  2 1508
                                                  4 1509  2 1511  2 1512  4 1513  2 1515  2 1516  4 1521  2 1522
                                                  2 1523  2 1524  4 1533  2 1534  2 1535  2 1536  4 1577  2 1578
                                                  2 1579
1044 S6                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1044  4 1506  2 1507  2 1508
                                                  4 1510  2 1511  2 1512  4 1514  2 1515  2 1516
1891 S9                        SCALAR            DOUBLE, ALIGNED, STATIC, INITIAL  XREF: 0 1891  2 2134
 793 T_FLAG                    BIT(1)            ALIGNED, AUTOMATIC        XREF: 0 0793  4 1200  4 1205  2 1210
1897 T_MASS                    SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1897  4 2309  2 2311  2 2322
1056 T_VAL                     SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1056  4 1230  2 1232  2 1234
                                                  2 1355  2 1356  2 1358  2 1359  2 1361  2 1496  2 1497  2 1498
                                                  2 1500
1875 T_XDOT                    7 - VECTOR        DOUBLE, ALIGNED, STATIC   XREF: 0 1875  4 2281  4 2293  4 2294
                                                  4 2296  4 2298  6 2321  4 2324  2 2325  2 2326
 214 TAN_THETA_MAX             SCALAR            DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0214  4 0611  2 0612
1872 TANK1                     SCALAR            DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1872  4 2117  2 2120
                                                  2 2122  2 2124  2 2126
1873 TANK2                     SCALAR            DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1873  4 2116  2 2117
                                                  6 2118  2 2121  2 2123  2 2125  2 2127
 718 THESIS_ALGORITHM          PROCEDURE           XREF: 0 0718  2 2651
  82 THETA_DOT_INITIAL         SCALAR            DOUBLE, ALIGNED, INITIAL  XREF: 0 0082  2 1948  2 1975
  54 THETA_FINAL               SCALAR            DOUBLE, ALIGNED, CONSTANT         XREF: 0 0054  2 1922
 215 THETA_MAX                 SCALAR            DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0215  4 0612  2 0613
 139 THETA_NOSE                SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0139  4 0606  2 0607  2 0613
                                                  2 0616  2 0620  2 0630  2 0646  2 1252  2 1253
 755 TIME_INTERVAL             SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0755  4 1737  4 1745  4 1746
                                                  2 1792  2 1811  2 1821  2 1835
1851 TIME_SET                  PROCEDURE           XREF: 0 1851  2 1943  2 1955  2 2054  2 2218  2 2268  2 2434
1885 TIME_SIGN                 SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 1805  4 1976  4 1977  2 1980
 753 TIME_STEP                 SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0753  2 1737  2 1745  2 1746
                                                  4 1944  4 1959  6 1980  2 1983  2 1990  4 2148  4 2219  4 2224
                                                  4 2269
 107 TJ_FUEL_FLOW              SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0107  4 0704  4 0708  2 0847
                                                  2 2296  2 2316
   9 TJ_MAX_FUEL_AIR_RATIO     SCALAR            DOUBLE, ALIGNED, CONSTANT         XREF: 0 0009  2 0305  2 1521
                                                  2 1558
1852 TSI                       INTEGER           SINGLE, ALIGNED, AUTOMATIC        XREF: 0 1852  4 1854  1 1857
                                                  1 1858  1 1859  1 1860  1 1863  1 1864  1 1865  1 1866
 140 TURBOJET_ISP              SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0140  4 0301  2 0305  2 0704
                                                  2 1319  2 1321  2 1322  2 1323  2 1430  2 1431  2 1484  2 1486
                                                  2 1487
 237 TURBOJET_MASS             SCALAR            DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0237  4 0264  2 0274
 118 TURBOJET_POWER            BIT(1)            ALIGNED, STATIC   XREF: 0 0118  2 0303  2 1319  2 1430  2 1453
                                                  2 1484  2 1519  2 1587  4 1917  4 2020  4 2027  4 2088  4 2156
                                                  4 2226  4 2234  4 2288  4 2289  2 2295  4 2306  4 2307  2 2315
 115 TURBOJET_THRUST           SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0115  4 0304  4 0305  4 0687
                                                  2 0704  2 0712  2 1215  2 1321  2 1322  2 1323  2 1431  2 1454
                                                  2 1486  2 1487
 136 T0                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0136  4 0519  4 0524  4 0529
                                                  2 0538  2 0539  2 0600  2 0632  2 0638  2 0668  2 1231  2 1232
                                                  2 1262  2 1265  2 1274  2 1275  2 1283  2 1284  2 1417
 189 T1                        SCALAR            DOUBLE, ALIGNED, STATIC   XREF: 0 0189  4 0600  4 0632  4 0638
```

```
 DCL NAME                    TYPE                  ATTRIBUTES & CROSS REFERENCE

                                                    4 0666  2 0667  6 0668  2 0671  2 0672  2 0677
 148 T2                      SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0148  4 0337  4 0677  2 1244
                                                    2 1245  2 1246  2 1265  2 1275  2 1284  2 1287  2 1423
 796 U                       2 X 2 MATRIX           DOUBLE, ALIGNED, STATIC   XREF: 0 0796  4 0831  4 0832  2 0967
                                                    2 1707  2 1709  2 1713  4 2634
 813 U_A                     INTEGER                SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0813  4 0819  2 0820
                                                    4 0821  2 0826
  76 U_ACTIVE                2 - VECTOR ARRAY       ARRAY(2001), DOUBLE, ALIGNED      XREF: 0 0076  2 0577  2 0578
                                                    2 0581  2 0582  2 1362  2 1363  2 1412  2 1476  2 1500  2 1952
                                                    6 1985  4 1986  2 2470  2 2475  4 2485  6 2490  6 2496  6 2497
                                                    2 2500  6 2503  6 2504  4 2515  4 2563  4 2565  4 2566  2 2592
 834 U_B                     INTEGER                SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0814  4 0822  2 0823
                                                    4 0824  2 0826
 812 U_COMPUTE               PROCEDURE                XREF: 0 0812  2 0966  2 1705
 815 U_I                     INTEGER                SINGLE, ALIGNED, AUTOMATIC        XREF: 0 0815  4 0818  2 0819
                                                    2 0822  4 0826  1 0827  1 0830  1 0831  1 0832
 751 U_J_OLD_TIME            SCALAR ARRAY           ARRAY(1001), DOUBLE, ALIGNED, STATIC      XREF: 0 0751  2 0827
                                                    2 0830  4 1996  4 2627
1901 U_KEEP                  2 - VECTOR             DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1901  4 2500  2 2504
1903 U_NEW                   2 - VECTOR ARRAY       ARRAY(1001), DOUBLE, ALIGNED, AUTOMATIC   XREF: 0 1903  4 2470
                                                    4 2475  2 2486
 752 U_NEW_TIME              SCALAR ARRAY           ARRAY(2001), DOUBLE, ALIGNED, STATIC      XREF: 0 0752  4 2438
                                                    2 2460  2 2465  2 2466  4 2629
  99 U_OLD_TIME              SCALAR ARRAY           ARRAY(2001), DOUBLE, ALIGNED      XREF: 0 0099  6 1992  2 1996
                                                    4 2062  2 2064  2 2066  2 2460  2 2465  2 2466  2 2468  2 2473
                                                    4 2516  4 2628
 218 U_R                     SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0218  4 0573  2 0589  2 0590
  55 U_R_FINAL               SCALAR                 DOUBLE, ALIGNED, CONSTANT         XREF: 0 0055  2 1921
 816 U_SET                   BIT(1)                 ALIGNED, AUTOMATIC        XREF: 0 0816  4 0825  2 0826  4 0829
 220 U_T_AIR                 SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0220  4 0588  2 0589  2 0590
 219 U_THETA                 SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0219  4 0574  2 0583
  56 U_THETA_FINAL           SCALAR                 DOUBLE, ALIGNED, CONSTANT         XREF: 0 0056  2 1923
 795 U_TIME                  INTEGER                SINGLE, ALIGNED, STATIC   XREF: 0 0795  2 0819  2 0822  1 0827
                                                    1 0830  4 0965  4 1704
  77 U_TIME_KEEP             SCALAR ARRAY           ARRAY(1001), DOUBLE, ALIGNED      XREF: 0 0077  2 0827  2 0830
                                                    4 2064  4 2635
 817 U_VAL                   SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0817  4 0830  2 0831
                                                    2 0832
   4 UNIVERSAL_G_CONSTANT    SCALAR                 DOUBLE, ALIGNED, CONSTANT         XREF: 0 0004  2 0540  2 1503
1904 UV1                     SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 1904  4 2465  6 2468
                                                    2 2469  2 2470  6 2473  2 2474  2 2475
 185 U0                      SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0185  2 0539  4 0590  2 0592
                                                    2 0602
 190 U1                      SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0190  4 0602  4 0672  2 0675
 125 U2                      SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0125  2 0305  2 0308  4 0526
                                                    4 0675  2 1317  2 1321  2 1322  2 1323  2 1333  2 1334  2 1335
                                                    2 1372  2 1373  2 1374  2 1375  2 1376  2 1377  2 1430  2 1431
                                                    2 1432  2 1433  2 1434  2 1435  2 1457  2 1458
  61 V                       13 X 13 MATRIX         DOUBLE, ALIGNED, INITIAL  XREF: 0 0061  2 0880  2 0954  2 0956
                                                    2 0961  2 0969
 992 VAL_1                   INTEGER                SINGLE, ALIGNED, STATIC   XREF: 0 0992  2 1183  4 1780
 225 VEHICLE                 PROCEDURE                XREF: 0 0225  2 0553  2 0581
 131 VEHICLE_ANGLE           SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0131  4 0589  2 0679  2 0680
 837 VEHICLE_MASS            SCALAR                 DOUBLE, ALIGNED, AUTOMATIC        XREF: 0 0837  4 0839  6 0841
                                                    2 0844  2 0846
```

```
 DCL  NAME            TYPE                   ATTRIBUTES & CROSS REFERENCE

1111  V0_2            SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 1111  4 1355  2 1356  2 1357
                                             2 1358  2 1359  2 1360  2 1361
  69  W               2 X 2 MATRIX ARRAY     ARRAY(1001), DOUBLE, ALIGNED     XREF: 0 0069  2 0831  2 0832
                                             4 2539  4 2545
2530  W_INDEX         INTEGER ARRAY          ARRAY(5), SINGLE, ALIGNED, STATIC, INITIAL       XREF: 0 2530
                                             2 2542  2 2545
2531  W_INIT          SCALAR ARRAY           ARRAY(5,2), DOUBLE, ALIGNED, STATIC, INITIAL     XREF: 0 2531
                                             2 2545
 175  W_SCRJ          SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0175  2 0263  2 0264  2 0266
                                             2 0267  2 0305  2 0308  4 0543
 142  WING_AREA       SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0142  4 0265  2 0268  2 0309
                                             2 0391  2 0392  2 1356  2 1357  2 1358  2 1359  2 1360  2 1361
                                             2 1447  2 1463  2 1464  2 1478  2 1479
 178  WING_SPAN       SCALAR                 DOUBLE, ALIGNED, STATIC   XREF: 0 0178  2 0265  2 0309  4 0546
 739  X_DOT           7 - VECTOR             DOUBLE, ALIGNED, STATIC   XREF: 0 0739  4 0843  4 0844  4 0845
                                             4 0846  4 0847  4 0848  4 0849  2 1165  4 2302  4 2313  4 2314
                                             4 2316  4 2318  4 2320  2 2321
 838  X_R             SCALAR                 DOUBLE, ALIGNED, AUTOMATIC       XREF: 0 0838  4 0842  2 0844
                                             2 0846
  98  X_STORE         7 - VECTOR ARRAY       ARRAY(2001), DOUBLE, ALIGNED     XREF: 0 0098  2 0513  2 0572
                                             2 0573  2 0574  2 0698  2 0839  2 0842  2 0843  2 0844  2 0845
                                             2 0846  2 1195  2 1230  2 1233  2 1355  2 1357  2 1360  2 1496
                                             2 1497  2 1499  2 1500  2 1555  2 1557  2 1561  2 1562  2 1563
                                             2 1739  4 1843  2 1846  4 1920  4 1921  4 1922  4 1923  4 1924
                                             4 1925  4 1926  2 1939  2 1940  2 1941  2 1948  2 1969  2 1970
                                             2 1971  2 1975  2 2039  2 2040  2 2041  2 2042  2 2043  2 2081
                                             2 2082  2 2083  2 2096  2 2110  2 2111  2 2294  2 2314  2 2507
                                             4 2511  4 2569  2 2577  2 2586  2 2599  4 2645
  24  ZLD             SCALAR ARRAY           ARRAY(16), DOUBLE, ALIGNED, CONSTANT     XREF: 0 0024  2 0360
                                             2 0363  2 0369  2 0371
```

```
B U I L T - I N   F U N C T I O N   C R O S S   R E F E R E N C E

   (CROSS REFERENCE FLAG KEY:  2 = REFERENCE, 1 = SUBSCRIPT USE)

  NAME       CROSS REFERENCE
ABS         XREF: 2 0897  2 2354  2 2460  2 2602  2 2604
COS         XREF: 2 0611  2 0623  2 0680  2 0695  2 1250  2 1253  2 1363  2 1412  2 1450  2 1501  2 2133
EXP         XREF: 2 0302  2 0532  2 0533  2 1315
LOG         XREF: 2 0531
MAX         XREF: 2 0618
MOD         XREF: 2 1201  2 1946  2 1995  2 2009  2 2063  2 2444  2 2457  2 2489  2 2493
SIN         XREF: 2 0266  2 0267  2 0623  2 0629  2 0630  2 0679  2 0694  2 1249  2 1252  2 1362  2 1451  2 1502  2 2116  2 2119
                  2 2129  2 2130  2 2132  2 2141
TAN         XREF: 2 0265  2 0611  2 0620  2 0623  2 2140
SIGN        XREF: 2 2355
SQRT        XREF: 2 0382  2 0478  2 0539  2 0590  2 0609  2 0643  2 0644  2 0651  2 0659  2 0670  2 0672  2 0715  2 0946  2 1244
                  2 1254  2 1273  2 1282  2 1422  2 2565
FLOOR       XREF: 2 0514  2 0823  2 0824  2 0963  2 1188  2 2484  2 2541
ARCSIN      XREF: 2 0610  2 0617
ARCTAN      XREF: 2 0589  2 0612  2 0645  2 0652  2 0660  2 1500
CEILING     XREF: 1 1996  1 2064  2 2080  2 2147  2 2203  2 2217  2 2223  2 2267  2 2277  2 2355  2 2446  2 2459  1 2490
TRUNCATE    XREF: 2 2356
TRANSPOSE   XREF: 2 0880  2 0956  2 0964  2 1666  2 1670  2 1672  2 1674  2 1676  2 1678  2 1683  2 1685  2 1687  2 1689  2 1691
                  2 1696  2 1698  2 1700  2 1709  2 1712  2 1713  2 2326
```

LIST OF REFERENCES

1. "Space Manufacturing Facilities (Space Colonies)," Jerry Grey, editor, American Institute of Aeronautics and Astronautics, Inc., New York, 1977, pp. 51-76.

2. Avery, W.H., "Twenty-five Years of Ramjet Development," Jet Propulsion, Vol. 25, No. 11, Nov. 1955, pp. 604-614.

3. Dugger, G.L., "Comparison of Hypersonic Ramjet Engines with Subsonic and Supersonic Combustion," Combustion and Propulsion, Fourth AGARD Colloquium, High Mach Number Air-Breathing Engines, A.L. Jaumotte, A.H. Lefebvre, A.M. Rothrock, editors, Pergamon Press, New York, 1961, pp. 84-110.

4. Hunter, Maxwell W. II, "Commercial Space Transporation Possibilities," American Astronautical Society paper no. 67-134, 1967.

5. Henry, John R. and Charles H. McLellan, "The Air-Breathing Launch Vehicle for Earth-Orbit Shuttle—New Technology and Development Approach," American Institute of Aeronautics and Astronautics paper no. 70-269, 1970.

6. Gregory, Thomas J., Louis J. Williams, and Darrell E. Wilcox, "The Air-Breathing Launch Vehicle for Earth-Orbit Shuttle—Performance and Operation," American Institute of Aeronautics and Astronautics paper no. 70-270, 1970.

7. Hunter, Maxwell W. II and Dietrich W. Fellenz, "The Hypersonic Transport—The Technology and the Potential," American Institute of Aeronautics and Astronautics paper no. 70-1218, 1970.

8. Johnston, P.J., J.M. Cubbage, and J.P. Weidner, "Studies of Engine-Airframe Integration on Hypersonic Aircraft," American Institute of Aeronautics and Astronautics paper no. 70-542, 1970.

9. Edwards, C.L.W., W.J. Small, J.P. Weidner, and P.J. Johnston, "Studies of Scramjet/Airframe Integration Techniques for Hypersonic Aircraft," American Institute of Aeronautics and Astronatics paper no. 75-58, 1975.

10. Weidner, J.P., W.J. Small, and J.A. Penland, "Scramjet Integration on Hypersonic Research Airplane Concepts, "American Institute of Aeronautics and Astronautics paper no. 76-755, 1976.

11. Wieting, Allan R. and Robert W. Guy, "Preliminary Thermal-Structural Design and Analysis of an Airframe-Integral Hydrogen-Cooled Scramjet," American Institute of Aeronautics and Astronautics paper no. 75-137, 1975.

12. Rogers, R.C., "Effects of Fuel Temperature on Supersonic Mixing and Combustion of Hydrogen, "American Institute of Aeronautics and Astronautics paper no. 77-17, 1977.

13. Hearth, Donald P. and Albert E. Preyss, "Hypersonic Technology—Approach to an Expanded Program," Astronautics and Aeronautics, Vol. 14, No. 12, Dec. 1976, pp. 20-37.

14. Hankey, Wilbur L., "Some Design Aspects of Hypersonic Vehicles," Aerospace Research Laboratories Report no. 70-0049, March 1970.

15. Bryson, A.E., Stanley E. Ross, "Optimum Rocket Trajectories with Aerodynamic Drag," Jet Propulsion, July, 1958, pp. 465-469.

16. Bryson, A.E., W.F. Denham, "A Steepest-Assent Method for Solving Optimum Programming Problems," Journal of Applied Mechanics, June, 1962, pp. 247-257.

17. Wilhite, Alan W., "Optimization of Rocket Propulsion Systems for Advanced Earth-to-Orbit Shuttles," Journal of Spacecraft and Rockets, Vol. 17, No. 2, March-April, 1980, pp. 99-104.

18. Nicolai, Leland M., Fundamentals of Aircraft Design, E.P. Domicone Printing Services, Fairborn, Ohio, pp. E4-E5.

19. C.S. Draper Laboratory Statement Level Simulator, Lift and Drag Data Set.

20. Martin, James Arthur, Aerospaceplane Optimization and Performance Estimation, Master's Thesis, Massachusetts Institute of Technology, August, 1967, p. 9.

21. Liepmann, H.W. and A. Roshko, Elements of Gas Dynamics, John Wiley and Sons, Inc., New York, 1957, p. 59.

22. Ibid., p. 86-87.

23. Ibid., p. 53, p. 99.

24. JP-121 course notes, California Institute of Technology, 1974.

25. "Scramjet Performance Characteristics" paper draft, Langley Research Center, 1978, Figure 16.

26. Ibid.

27. Jones, Robert A. and Paul W. Huber, "Airframe Integrated Propulsion System for Hypersonic Cruise Vehicles", paper presented at the 11th Congress of the International Council of the Aeronautical Sciences, September, 1978, p. 5.

28. Ibid., p. 2.

29. Shuttle Operational Data Book, Volume II, "Mission Mass Properties", September, 1975.

30. C.S. Draper Laboratory Statement Level Simulator, Lift and Drag Data Set.

31. U.S. Standard Atmosphere, 1962, U.S. Government Printing Office, Washington, D.C., December, 1962.

32. Beer, F.P. and E.R. Johnston, Vector Mechanics for Engineers: Statics and Dynamics, McGraw-Hill Book Company, New York, 1962, p. 467.

33. Henry, Beverly Z. and John P. Decker, "Future Earth Orbit Transportation Systems/Technology Implications," Astronautics and Aeronautics, Vol. 14, No. 9, Sept. 1976, p. 25.

34. Voss, Janice, Optimization of Variable Mixture Ratio Rocket Boosters, Masters Thesis, Massachusetts Institute of Technology, May, 1977.

35. Satellite Power Systems (SPS) Concept Definition Study, Final Report (Exhibit C), Volume IV, "Transportation Analysis", Appendix A.

BIOGRAPHY

Philip David Hattis was born in Chicago, Illinois on 18 July 1952. He received his early education at public schools in Winnetka, Illinois and graduated from New Trier East High School in 1970. He entered Northwestern University as an undergraduate in 1970 and received his Bachelor of Science degree in Mechanical Engineering in 1973. Having enrolled in the graduate school of the California Institute of Technology in 1973, he was granted the Master of Science degree in Aeronautics in 1974. Since 1974, Mr. Hattis has been pursuing his doctoral degree in the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology (MIT).

While attending graduate school at MIT, Mr. Hattis has been a Draper Fellow at The Charles Stark Draper Laboratory (CSDL). On two occasions, he has also been a Summer Staff member at CSDL.

Mr. Hattis is a member of the Pi Tau Sigma, Tau Beta Pi, and Sigma Xi honorary societies and is also a member of the American Society of Mechanical Engineers and the American Institute of Aeronautics and Astronautics.