HYBRID 3-D
ROCKET TRAJECTORY PROGRAM

Prepared for
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WALLOPS STATION
WALLOPS ISLAND, VIRGINIA 23337

CONTRACT NO. NAS6-2369
WORK ORDER 7

(NASA-CR-137472) HYBRID 3-D ROCKET
TRAJECTORY PROGRAM. PART 1: FORMULATION
AND ANALYSIS. PART 2: COMPUTER
(Computer Sciences Corp., Wallops Island,
Va.) 112 p HC $8.75
CSCL 22C 63/30 50856

CSC
COMPUTER SCIENCES CORPORATION
Hybrid 3-D
Rocket Trajectory Program

Part One: Formulation and Analysis
by
Louis C. P. Huang

Part Two: Computer Programming and User's Instruction
by
Ronald A. Cook

Prepared for
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WALLOPS STATION
WALLOPS ISLAND, VIRGINIA 23337
1973
Part One

Formulation and Analysis
A. INTRODUCTION

A rocket in flight has three linear and three angular degrees of freedom. A mathematical model must be devised which describes the motions within the six degrees of freedom and this '6-D' model may be programmed for solution by a digital computer. Given a set of initial conditions and tables of rocket characteristics, the equations which make up the model may be evaluated repeatedly with respect to time to give an accurate simulation of the flight of a rocket.

The '6-D' model, however, has two deficiencies:

1. Significant effort is required in preparing the input data to be used by the computer program. A specimen '6-D' simulation shows that over 700 input data items are required. When trajectories must be predicted for a new type of rocket, only part of this data set may be available from the manufacturer. The remaining items must be calculated from aerodynamic theory. All the data must then be transcribed in the exact form required. As a result, preparation of the input data usually takes at least two man-weeks.

2. Each '6-D' computer run may require as much as four hours of expensive computer time.

For these reasons, models utilizing various subsets of the six degrees of freedom are used in trajectory simulation. A '3-D' model with only linear degrees of freedom is especially attractive, since the coefficients for the angular degrees of freedom are the most difficult to determine and the angular equations are the most time consuming for the computer to evaluate.

Of course, the '3-D' model is less accurate than the '6-D' model. This is because the model lacks angular motions and the thrust vector orientation is assumed to be aligned with the velocity vector. Unless the angle of attack is zero, this is not true.
Figure 1 shows a typical angle of attack versus time history of an unguided rocket. The time scale is divided into three periods. In Period I the rocket has a finite angle of attack and is said to be untrimmed. In Period II the angle of attack has been trimmed out to zero and in Period III the rocket is again untrimmed.

The manner in which angle of attack is generated is shown in Figure 2, a vector diagram of the forces on the rocket immediately after launch. Angle BOC is the launch elevation angle. OC is the thrust minus drag vector, aligned with the principal body axis. Vector CD represents the acceleration of gravity. Vector OD is the sum of OC (the thrust minus drag) and CD (gravity). The rocket is accelerated from rest in direction OD, but it is pointing in direction OC. Angle DOC represents the angle of attack.

This angle of attack is gradually reduced to zero by the stabilizing moment (Mo) produced by the rocket fins. Trimming does not take place immediately because the stabilizing tendency is resisted by the large moment of inertia about the rocket's principal axis.

If the angle of attack is assumed to be zero during Period I (i.e., before the rocket trims out), large errors in the predicted trajectory may result. This is the case when standard three linear degree of freedom simulations are used. Further errors may result from assumptions about other initial and launch conditions. To reduce these errors, the computer program described in this paper uses three separate subsections to predict trajectories. A launch rail subsection is used until the rocket has left its launcher. The program then switches to a special '3-D' section which computes motions in two linear and one angular degrees of freedom. This permits accurate simulation of Period I flight when the angle of attack is finite. When the rocket trims out, the program switches to the standard, three linear degrees of freedom model. This model is used throughout Periods II and III.
Velocity

PERIOD

UNTRIMMED

TRIMMED

UNTRIMMED

Low Velocity

Low Density

FLIGHT TIME (t)

FIGURE 1: Angle-of-Attack/Time History

VECTOR FORCE

OC = Thrust-Drag
+CD = Gravity
OD = Resultant

FIGURE 2: Forces on Rocket immediately after Launch
The standard '3-D' model can be used accurately in Period III despite the untrimmed condition because the rocket is not thrusting. This untrimmed condition results from the lack of restoring moment due to low atmospheric density. The aerodynamic force on the fins varies as the product of velocity squared and atmospheric density. Thus, although the velocity is high in Period III, the density is low enough that the restoring moment is insufficient to maintain a zero angle of attack.

In the absence of thrust, therefore, the untrimmed flight of Period III has no effect on the trajectory. Should there be an upper stage to be fired in Period III, the "HYBRID 3-D" program has a thrust vector control option to handle this case. There is also an option to change the aerodynamic drag on the spent rocket during this period.

By using the launch rail subsection, the special '3-D' section, the standard '3-D' section, and the thrust vector control and aerodynamic drag options, the program can calculate the rocket's impact point with sufficient accuracy for range safety hazard estimation and for the planning of payload recovery operations.

B. BASIC ASSUMPTIONS

The mathematical model used in the "HYBRID 3-D" program includes such effects as the variation of aerodynamic coefficients with Mach number, the change of engine thrust with time, and the change of rocket mass due to propellant burning and stage separation. But some simplifying assumptions have been necessary to speed calculation. These are detailed below.

Please note that where equations are given, the notation is similar to that of FORTRAN, with the provision that all variables are of type 'REAL'.

1. Earth's Geometric Shape

   The shape of the earth is a "pear" shaped spheroid with a slightly smaller northern hemisphere. The oblate spheroid is usually used as an approximation to the earth's shape. However, for sounding rocket trajectories,
where the range of the rocket is short in comparison to the circumference of the earth, a simplified model may be utilized: the "local spherical earth" in which we approximate the earth's shape by a sphere with a radius equal to the geocentric radius at the launch site. In the program, the local geocentric radius of the launch site (RE) is set to 20899262 feet, a value appropriate for Wallops Island.

2. Earth's Gravitational Field

The gravitational potential of the earth is simplified to the inverse square law without harmonics, i.e.:

\[
GG = GM/R^3
\]  

(B-1)

where:  
\( GG \) - is the gravitational potential;  
\( GM \) - is defined as \( 1.4076576 \times 10^{16} \) feet\(^3\)/seconds\(^2\);  
\( R \) - is the distance from the geocenter to the vehicle.

The components of the gravitational potential are:

\[
GX = -GG\times X \\
GY = -GG\times Y \\
GZ = -GG\times Z
\]

where X, Y, and Z are the geocentric coordinates of the vehicle.

3. Atmosphere

We have selected an atmospheric model which closely approximates the 1962 U.S. Standard Atmosphere. (Reference 6)

The subroutine ATMSPH is called with altitude as the argument. The atmospheric parameters pressure, density, temperature, viscosity, and speed of sound are returned.

4. Earth's Rotation

During the short duration of Period I, the rotation of the earth may be neglected without appreciable error. In Periods II and III, the effect of
the earth's rotation is accounted for in a transformation from a Launch Inertial Coordinate System to an Earth Fixed Coordinate System.

C. COORDINATE SYSTEMS

In the trajectory calculation, five sets of Cartesian coordinates are used. They are discussed in detail below:

1. Inertial Coordinate System \((X, Y, Z)\) (Inertial at Launch):
   The origin of the Inertial System is the earth's center. Once the \(X\) and \(Y\) axes are determined at launch time, the Inertial coordinates are fixed in earth-centered space, and do not rotate with the earth.
   \(X\) - on the earth's equatorial plane, pointing to zero longitude at launch.
   \(Y\) - on the earth's equatorial plane, pointing to 90° East longitude at launch.
   \(Z\) - perpendicular to the equatorial plane, pointing to the North Pole.

2. Earth-Fixed Coordinate System \((X_E, Y_E, Z_E)\):
   The origin of the Earth-Fixed coordinate system is the center of the earth. However, unlike the Inertial system, whose origin is also the earth's center, the \(X_E\) and \(Y_E\) axes of the Earth-Fixed system are the Greenwich and 90° longitudes respectively and rotate with the earth.
   At the moment of launch, the Earth-Fixed axes coincide with the Inertial axes.
   \(X_E\) - on the earth's equatorial plane, always pointing to the Greenwich longitude.
   \(Y_E\) - on the earth's equatorial plane, always pointing to 90° East longitude.
   \(Z_E\) - perpendicular to the equatorial plane, pointing to the North Pole.
3. **Instantaneous-Topocentric Coordinate System \((x, y, z)\):**

The origin of the Instantaneous-Topocentric coordinates is the projection point of the moving rocket on the earth's surface, the point at which the geocentric radius vector to the rocket intersects the earth's surface.

- **x** - on the local horizon plane tangent to the instantaneous projection of the rocket, directed along the local geocentric north.
- **y** - on the local horizon plane tangent to the instantaneous projection of the rocket, directed along the local geocentric east.
- **z** - perpendicular to the instantaneous local tangent plane, directed along the geocentric radius vector, and pointing toward the earth's center to complete the right-hand system.

4. **Launch Coordinate System \((x_L, y_L, z_L)\):**

The origin of the Launch coordinate system is the launch site. The \(x_L\) axis is chosen to point in the direction of launch.

- **\(x_L\)** - on the launch-tangent plane, pointing in the direction of launch.
- **\(y_L\)** - on the launch-tangent plane, pointing normal to the launch azimuth in the direction which with \(x_L\) and \(z_L\) forms a right-hand system.
- **\(z_L\)** - perpendicular to the launch-tangent plane, positive upward from the earth's center.

5. **Body Coordinate System \((x_B, y_B, z_B)\):**

The origin of the Body coordinate system is the center of mass of the rocket.

- **\(x_B\)** - along the rocket principle axis, positive forward.
- **\(y_B\)** - normal to the \(x_B - z_B\) plane in the direction which completes the right-hand system.
- **\(z_B\)** - perpendicular to the \(x_B\) axis and contained in the plane of symmetry of the rocket, positive downward.
D. **DYNAMIC EQUATIONS OF THE ROCKET**

"HYBRID 3-D" simulates the trajectory of the rocket with two models, one for Period I and another for the rest of the flight. The dynamic equations for each model are given here. A FORTRAN-like notation is used (all variables are type "REAL").

1. The Equations for Period I, using two linear and one angular degrees of freedom:

   - \[ \text{AXL} = \frac{((\text{THRUST-DRAG}) \cdot \text{CS} - \text{FNORM} \cdot \text{SN})/\text{MASS}}{\text{AXL}} \]
   - \[ \text{AZL} = \frac{((\text{THRUST-DRAG}) \cdot \text{SN} + \text{FNORM} \cdot \text{CS})/\text{MASS-G0}}{\text{AZL}} \]
   - \[ \text{THDD} = \frac{-\text{LSM} \cdot \text{FNORM}}{\text{INERT}} \]

   where
   - AXL - acceleration in the XL direction, which is downrange at launch
   - THRUST - the thrust of the rocket motor. Found from table look-up, with time as the argument
   - DRAG = CDS \cdot Q
   - Q = \frac{(\text{DENS} \cdot \text{VL} \cdot 2)}{2.0}
   - VL = \sqrt{\text{VXL} \cdot 2 + \text{VZL} \cdot 2}
   - CDS - is given by sub-routine TAB1 which enters a look-up table with Mach number as the argument to find CD, the coefficient of drag, and multiplies this by SAREA, the reference area
   - Q - is called the "dynamic pressure".
   - DENS - is given by the Standard Atmosphere sub-routine 'ATMSPH', entered with altitude as the argument. This sub-routine also gives
     - TEMP - temperature
     - PRES - atmospheric pressure
     - VISC - viscosity
     - SOUNP - the speed of sound.
VXL, VZL - are velocity components in the downrange and vertical directions. Found by integration of AXL and AZL.

CS = \cos(\text{THETA})

THETA - is the inclination angle of the rocket, the angle between the principal body axis and the horizontal. It is originally set to the launch elevation angle, and is modified by double integration of THDD, THETA double dot.

FNORM = CNA \cdot SAREA \cdot Q \cdot ALPH

CNA - is the slope of the coefficient of the normal force acting on the center of pressure.

SAREA - is the reference area.

Q - is defined above.

ALPH - THETA - GAMA, and is called the "angle of attack". The program switches from the Period I model to the model used for the rest of the flight when ALPH becomes zero (found by interpolation as it crosses from plus to minus). The duration of Period I is about 0.8 second for a NIKE-CAJUN, and about 5 seconds for a SCOUT.

GAMA = \arctan(VZL/VXL)

GAMA is the flight path angle which defines the direction in which the rocket is moving.

MASS - is the weight of the rocket divided by the force of gravity.

AZL - is acceleration in the Z direction, which is vertical at the launch site.

THDD - is THETA double dot, the acceleration in the rocket's inclination angle to the XL axis.

LSM = LCP-LCG.

LCP - is the distance from the reference position (usually the nose of the rocket) to the center of pressure.

LCG - is the distance from the reference position to the rocket's center of gravity.

INERT - is the rocket's pitch moment of inertia.

The equations above have been simplified by the elimination of effects such as pitch damping and jet damping which were found to have no significant
effect on the solution, since Period I is of such short duration. Some
variables are made constants, with the same justification. Some of
these are: GO; CNA; LSM; and INERT.

2. The Equations for Periods II and III:
The angular motion of the rocket is neglected; only the linear motion
is considered. The equations of motion are written in the Inertial
Coordinate System.

\[
\begin{align*}
X\text{DDOT} &= \frac{G_X + (T_H - U_X)}{M} \\
Y\text{DDOT} &= \frac{G_Y + (T_Y - U_Y)}{M} \\
Z\text{DDOT} &= \frac{G_Z + (T_Z - U_Z)}{M} \\
\end{align*}
\]

- \(X\text{DDOT}\) is \(X\) double dot; the acceleration is the direction of the
Inertial \(X\)-axis (please refer to Section C for a definition
of the various coordinate systems used).

- \(G_X\) is the \(X\) component of the acceleration of gravity, as
felt at the rocket's center of gravity.

- \(T_H\) is the rocket's component of thrust in the \(X\)-direction.
This component of thrust would normally be taken in the
direction of rocket motion, except that it may be modified
by the thrust control option. The thrust control option
may be specified to hold the thrust direction of the rocket
constant after a given time of flight or rocket altitude.
This option is useful when the rocket attains an altitude
where the atmospheric density is so low that the rocket's
gyroscopic stability tends to keep it pointing in a constant
direction and there is to be an upper-stage firing (Period III).

- \(U_X\) is the component of drag in the inertial, \(X\) direction.

- \(M\) is the present weight of the rocket, derived from weight
tables, divided by the sea-level acceleration due to gravity.

- \(Y\text{DDOT}, Z\text{DDOT}\) are the components of acceleration in the inertial \(Y\) and \(Z\)
directions. Their equations are symmetrical with the one
for \(X\text{DDOT}\).
3. The Equations for the Launch Rail:

The user may specify the type of launcher to be used for the simulation. The simulated rocket will then be accelerated from rest along the launch rail and will have an initial velocity at release. This initial velocity will decrease the duration of Period I. The equation of motion is:

\[ \text{LACC} = \frac{(\text{THRUST} - \text{WEIGHT} \cdot \text{SEL} - \text{DRAG})}{\text{MASS}} \]

- \text{LACC} is acceleration in the direction of the launch rail
- \text{THRUST} is found by table look-up, with time as the argument
- \text{WEIGHT} is the rocket weight, found by table look-up; time is the argument.
- \text{SEL} is the sine of the elevation angle of the launcher.
- \text{DRAG} is the aerodynamic drag of the rocket. This term is quite small, and could be neglected. The friction drag of the launcher is not included, although the equation might be reformulated to add it to the aerodynamic drag. In the practical case of launch elevation angles of perhaps 80°, the normal force on the launch rail (which is multiplied by the coefficient of friction to give the friction drag) is very low, resulting in negligible friction drag.

\[ \text{MASS} = \frac{\text{WEIGHT}}{\text{GO}} \]

- \text{GO} is the acceleration due to gravity at sea level.

The initial conditions are entered and then this equation is integrated repeatedly to give the velocity and distance traveled on the launch rail. When this distance equals the length of the launcher, the program switches to solution of the equations for Period I.
E. SUMMARY

A computer program, "HYBRID 3-D", has been described which simplifies trajectory simulation.

A comparison of a "6-D" trajectory, a "standard" 3-d trajectory, and "HYBRID 3-D" is shown in Figure 3. "HYBRID 3-D" achieves good accuracy through the special treatment of the untrimmed period near launch.

"HYBRID 3-D" also contains a simulation of the dynamics of the launch rail to enhance accuracy in simulating the trajectory of slow-accelerating rockets.
Figure 3.
Part Two

Computer Programming and User's Instruction
# Table of Contents

1. Introduction  
2. Program Variable Definitions  
3. Integration Routine  
4. General Flow Chart  
5. Subroutine Description  
6. Input Description  
7. Output Description  
8. Deck Setup  
9. Sample Input and Output  
10. Program Listing
I. INTRODUCTION

This trajectory program is written in Fortran IV and was tested on the Honeywell 625 computer. Input data is transferred from punched cards to disc (file code 05). The program reads this disc, prints the data, rewinds the disc, and rereads the disc as data is needed for calculations. The math model includes aerodynamic drag, rotating earth, launch rail simulation, rigid body dynamics at lift-off, and standard 3D point-mass equations. After leaving the launcher, the rocket is restricted to motion in two dimensions until the angle of attack goes to zero; then, the motion becomes three dimensional with thrust and velocity vectors aligned. The equations of motion are integrated by the Runge-Kutta method, utilizing variable step size for better efficiency. Processing time is approximately 0.7 minutes for each 100 seconds of trajectory time. Output data may be printed in English or metric units, and written on tape (file code 11).
2. PROGRAM VARIABLE DEFINITIONS

A = semi-major axis of IIP ellipse
ACX,ACY,ACZ = components of acceleration with respect to topocentric system
ACCI = inertial acceleration
ALIM = altitude where pressure and density are set equal to zero (400,000 ft.)
ALPH = angle of attack (α)
ALT = altitude
ALTC = input altitude at which thrust vector remains in constant direction
AREA = rocket nozzle exit area (Ae)
ARRAY = matrix for storing all output data (in English units)
ARREY = matrix for storing all output data in metric units
AXL = acceleration along launch azimuth in launch coordinate system
AZ = flight azimuth angle; or launch azimuth (λ)
AZL = acceleration in vertical direction in launch coordinate system
AZLCH,ELLCH = look azimuth and look elevation angle with respect to launch site
AZRAD,ELRAD = look azimuth and look elevation angle with respect to radar
CAPT = time from launch to impact at IIP
CD = drag coefficient (Cd)
CDD = input table of drag coefficients (Cd); (includes drag area, if SAREA = 1.0)
CNA = normal force coefficient
CONV = π/180°
C PHI = cos (ϕ)
DELTA = range angle (δ)
DENS = atmospheric density
DIFF = thrust minus drag
DIST = distance traveled on launch rail
DRAG = drag force (D)
DT = integration time interval
EDLCH, NDLCH, ZDLCH = east, north, vertical velocity components with respect to launch site
EDRAD, NDRAD, ZDRAD = east, north, vertical velocity components with respect to radar
EL = flight elevation angle or launch elevation angle (r)
ELCH, NLCH, ZLCH = east, north, vertical coordinates with respect to launch site
ELPREV = previous value of velocity elevation angle
EPBIG = maximum relative error allowed in Runge-Kutta method (1.0 x 10^-5)
EPS = geodetic latitude minus geocentric latitude
EPTINY = minimum relative error allowed in Runge-Kutta method (5.0 x 10^-7)
ERAD, NRAD, ZRAD = east, north, vertical coordinates of rocket with respect to radar
FNORM = normal force on rocket
FRAC = interpolation fraction for Mach number
GAMA = velocity elevation angle (r)
GO = gravity acceleration at Earth surface (32.174 ft/sec^2)
GDLAT = geodetic latitude
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM</td>
<td>gravitation constant (1.4076576 \times 10^{16} \text{ ft}^3/\text{sec}^2)</td>
</tr>
<tr>
<td>GX, GY, GZ</td>
<td>components of gravity acceleration</td>
</tr>
<tr>
<td>HIGH</td>
<td>maximum integration interval = one half the print time interval</td>
</tr>
<tr>
<td>IEND</td>
<td>control constant (set = 1 at impact)</td>
</tr>
<tr>
<td>IIPLAT, IIPLON</td>
<td>latitude (GD), longitude, ground range, and impact time for instantaneous impact point</td>
</tr>
<tr>
<td>IIPR, IIPRTIM</td>
<td>latitude (GD), longitude, ground range, and impact time for instantaneous impact point</td>
</tr>
<tr>
<td>INERT</td>
<td>moment of inertia of rocket ((I))</td>
</tr>
<tr>
<td>INEXT</td>
<td>control constant (set = 1 at each phase time)</td>
</tr>
<tr>
<td>IOPT</td>
<td>thrust reference option (=1 for sea level, =2 for vacuum)</td>
</tr>
<tr>
<td>IPRINT</td>
<td>control constant (set = 1 when printout is desired)</td>
</tr>
<tr>
<td>K</td>
<td>matrix for storing (K) values for Runge-Kutta method</td>
</tr>
<tr>
<td>KCON</td>
<td>1 for main rocket calculations; 2 for spent stage calculations</td>
</tr>
<tr>
<td>KDEL</td>
<td>increment for subscript of phase control constant</td>
</tr>
<tr>
<td>KPAG</td>
<td>output page number</td>
</tr>
<tr>
<td>KSPENT</td>
<td>present spent stage number</td>
</tr>
<tr>
<td>KSTOP</td>
<td>0 for no stop at apogee, 1 for stopping calculations at apogee</td>
</tr>
<tr>
<td>LACC</td>
<td>acceleration on launch rail</td>
</tr>
<tr>
<td>LAMDA</td>
<td>angle from (X) axis in (X, Y) plane (\lambda)</td>
</tr>
<tr>
<td>LAMDAO</td>
<td>launch longitude (\lambda_0)</td>
</tr>
<tr>
<td>LAMDAL</td>
<td>radar longitude (\lambda_1)</td>
</tr>
<tr>
<td>LAT</td>
<td>geocentric latitude (\phi)</td>
</tr>
<tr>
<td>LENGTH</td>
<td>length of launch rail</td>
</tr>
<tr>
<td>LINE</td>
<td>number of printout line</td>
</tr>
</tbody>
</table>
LONG = longitude (\(\lambda\))
LOW = smallest integration time interval (1.0 x 10^{-4} sec)
LSM = distance from center of gravity to center of pressure (\(\ell_{sm}\))
LVEL = velocity on launch rail
MACH = Mach number (M)
MASS = total rocket mass
MCH = input table of Mach numbers corresponding to table of drag coefficients
MESS = array for storing printout message
NAP = control constant (= 1 at apogee)
NCD = number of drag coefficients in table
NDEL = increment for subscript of phase time
NGEO = 0 for geocentric elevation angle and azimuth (input)
        1 for geodetic elevation angle and azimuth (input)
NID = control constants for phase changes
       = 1, read phase message only
       = 2, read phase message and drag table
       = 3, read phase message and thrust, weight, drag tables
       = 4, nothing is read in
NIIP = input constant for special output tape
       = 0, tape time interval same as print time interval
       = 1, tape time interval equal 0.1 sec, and PTI must be an integer multiple of 0.1 sec
       = 2, no tape output
NMT = number of rocket motors
NPAGE = 1 for output page A
        2 for output page B
        3 for output page C
        4 for output page D
        5 for output page E
        6 for output page F
NPH = present phase number
NPHS = number of phase times
NSEP = number of motor separations
NSKP = input control constant
      = 0, ALPHA routine performed
      = 1, ALPHA routine skipped
NSPENT = number of spent stages left to be calculated
NSTOP = 0 for TSTOP = 0.0 (no termination on time)
        1 for TSTOP ≠ 0 (trajectory terminates on time)
NSYS = output control constant
       1, for English units
       2, for metric units
       3, both English and metric
NTH = number of thrust values in table
NWT = number of propellant weights in table
OMEGA = earth rotation speed (7.29211 x 10^-5 rad/sec)
        (ω)
PO = atmospheric pressure at surface of earth
     (2115.666 lbs/ft^2)
PHAS = output message in column 2 to 7
PHI = angle from X, Y plane to R vector (φ)
PHIO = launcher geodetic latitude
PHI1 = radar site geodetic latitude
PHT = phase times (input)
PI = 3.14159265 (π)
PI2 = 2π
PRES = atmospheric pressure (p)
PTI = print time interval
PTIME = print time
Q = dynamic pressure
R = distance from earth center to rocket
RA = equatorial radius (20925741 ft)
RANGE = ground range from launch site to rocket
RB = polar radius (20855591 ft)
RE = earth radius (20899262 ft)
RLCH = slant range from launch site to rocket
RRAD = slant range from radar to rocket
S = matrix for storing \( X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}, \ddot{X}, \ddot{Y}, \ddot{Z} \) in Runge-Kutta routine
SAREA = drag area (= 1.0 when area is included in drag coefficient)
SA = matrix for storing all output data
SMAT = transformation matrix, topocentric to inertial
SOUND = speed of sound
SPHI = \( \sin (\phi) \)
SPS = array for storing initial spent stage variables
STPTI = stored value of the print time interval
TO = launch time = 0.0 sec \( (t_0) \)
T1 = time to begin constant thrust direction in inertial system
TEMP = atmospheric temperature
TFRAC = time interpolation fraction
THCON = 1 for no thrust control
2 for constant thrust direction beginning at time T1, or at altitude ALTC
THDD = angular acceleration in launch coordinate system
THETA = thrust elevation angle
THREF = thrust at sea level or in vacuum
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>THRUST =</td>
<td>rocket thrust (TH)</td>
</tr>
<tr>
<td>THS =</td>
<td>thrust table input</td>
</tr>
<tr>
<td>TIM =</td>
<td>time table corresponding to thrust table</td>
</tr>
<tr>
<td>TIM1, TIM2 =</td>
<td>times used to control writing output tape</td>
</tr>
<tr>
<td>TIME =</td>
<td>time from launch (t)</td>
</tr>
<tr>
<td>TMAT =</td>
<td>transformation matrix, inertial to topocentric</td>
</tr>
<tr>
<td>TSEP =</td>
<td>table of separation times for single stage rocket, set TSEP = 0.0</td>
</tr>
<tr>
<td>TSTOP =</td>
<td>time to stop calculations</td>
</tr>
<tr>
<td>TWT =</td>
<td>time table corresponding to propellant weight table</td>
</tr>
<tr>
<td>VEL =</td>
<td>total speed of rocket (inertial)</td>
</tr>
<tr>
<td>VELLCH =</td>
<td>rocket speed with respect to launch site</td>
</tr>
<tr>
<td>VELRAD =</td>
<td>rocket speed with respect to radar</td>
</tr>
<tr>
<td>VISC =</td>
<td>atmospheric viscosity</td>
</tr>
<tr>
<td>VT =</td>
<td>total velocity (topocentric)</td>
</tr>
<tr>
<td>VX, VY, VZ =</td>
<td>topocentric velocity components</td>
</tr>
<tr>
<td>WEIGHT =</td>
<td>total rocket weight</td>
</tr>
<tr>
<td>WPL =</td>
<td>payload weight</td>
</tr>
<tr>
<td>WPP =</td>
<td>present propellant weight</td>
</tr>
<tr>
<td>WPP2 =</td>
<td>total propellant weight of all unfired motors</td>
</tr>
<tr>
<td>WPR =</td>
<td>input table for propellant weight</td>
</tr>
<tr>
<td>WRM =</td>
<td>present rocket motor weight</td>
</tr>
<tr>
<td>WTM =</td>
<td>table of rocket motor weights</td>
</tr>
<tr>
<td>WTP =</td>
<td>initial propellant weight in each rocket</td>
</tr>
<tr>
<td>WTS =</td>
<td>spent stage weight</td>
</tr>
<tr>
<td>X, Y, Z =</td>
<td>inertial coordinates</td>
</tr>
</tbody>
</table>
\( \text{XDOT, YDOT, ZDOT} = \) inertial velocity components \((X, Y, Z)\)

\( \text{XDDOT, YDDOT, ZDDOT} = \) inertial acceleration components \((\ddot{X}, \ddot{Y}, \ddot{Z})\)

\( \text{XIP, YIP, ZIP} = \) inertial position of the instantaneous impact point

\( \text{XL, YL, ZL} = \) launch system coordinates where \(XL\) is along launch azimuth

\( \text{XT, YT, ZT} = \) topocentric coordinates
3. RUNGE-KUTTA INTEGRATION ROUTINE

For a discussion of the Runge-Kutta formula of order 4, see Reference 7.

Here, only the equations in X are listed, since the Y and Z equations correspond exactly.

(NOTE: the actual subroutine uses matrix S to store all the inertial variables, X, Y, Z, \( \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z} \))

Step 1: Input \( x_0, \dot{x}_0, \ddot{x}_0 \); time = \( T_0 \)

\[
K_{11} = (DT) \dot{x}_0 \quad K_{41} = (DT) \ddot{x}_0
\]

Step 2: time = \( T_1 = T_0 + DT/2 \)

\[
x_1 = x_0 + \left(\frac{DT}{2}\right) \dot{x}_0
\]

\[
\dot{x}_1 = \dot{x}_0 + \left(\frac{DT}{2}\right) \ddot{x}_0
\]

\[
\ddot{x}_1 = \ddot{x} (T_1, x_1, y_1, z_1, \dot{x}_1, \dot{y}_1, \dot{z}_1)
\]

\[
K_{12} = (DT) \dot{x}_1 \quad K_{42} = (DT) \ddot{x}_1
\]
Step 3: \[ \text{time} = T_2 = T_0 + \frac{DT}{2} \quad \text{(NOTE:} \ T_2 = T_1) \]

\[
\begin{align*}
X_2 &= X_0 + \left(\frac{DT}{2}\right) X_1 \\
\dot{X}_2 &= \dot{X}_0 + \left(\frac{DT}{2}\right) \ddot{X}_1 \\
\ddot{X}_2 &= \dddot{X} (T_2, X_2, \dot{X}_2, \dddot{Z}_2) \\
K_{13} &= (DT) \ddot{X}_2 \\
K_{43} &= (DT) \dddot{X}_2
\end{align*}
\]

Step 4: \[ \text{time} = T_3 = T_0 + DT \]

\[
\begin{align*}
X_3 &= X_0 + (DT) \dot{X}_2 \\
\dot{X}_3 &= \dot{X}_0 + (DT) \ddot{X}_2 \\
\ddot{X}_3 &= \dddot{X} (T_3, X_3, \dot{X}_3, \dddot{Z}_3) \\
K_{14} &= (DT) \dot{X}_3 \\
K_{44} &= (DT) \dddot{X}_3
\end{align*}
\]

Solution at time = \( T_4 = T_0 + DT \)

\[
\begin{align*}
X &= X_0 + (1/6) (K_{11} + 2K_{12} + 2K_{13} + K_{14}) \\
\dot{X} &= \dot{X}_0 + (1/6) (K_{14} + 2K_{42} + 2K_{43} + K_{44}) \\
\dddot{X} &= \dddot{X} (T_4, X, \dot{X}, \dddot{Z})
\end{align*}
\]

Error Analysis:

The Runge-Kutta method of order 4 will result in an error of order 5, which is reduced by using "extrapolation to the limit".
First, find a solution using DT = H; then find a corresponding solution by using DT = (H/2) twice. Finally, combine the two solutions in such a way that most of the order 5 error is eliminated.

\[ X_{\text{EX}} = \text{exact solution} \]
\[ X(1) = \text{solution using DT = H} \]
\[ X(2) = \text{solution using DT = H/2 twice} \]
\[ X(1) = X_{\text{EX}} + AH^5 + \text{order 6} \]
\[ X(2) = X_{\text{EX}} + B(H/2)^5 + C(H/2)^5 + \text{order 6} \]

The factors A, B, and C are composed of derivatives of the acceleration function. The X(2) solution has two order 5 errors, because twice as many steps are required when DT = (H/2).

If H is small, A, B, and C will be approximately equal, resulting in:

\[ X(1) - X_{\text{EX}} = AH^5 \]
\[ X(2) - X_{\text{EX}} = \frac{AH^5}{16} \]
\[ X_{\text{EX}} = X(2) + (1/15) \left[ X(2) - X(1) \right] \]

The right hand side is an improved approximation for the exact solution. Similar equations are used for Y, Z, X, Y, Z.
Optimum DT
In the Runge-Kutta formula, the theoretical error term is of order 5, and the smaller the step size, DT, the smaller this error becomes. However, when DT is made smaller, the number of integration steps increases, resulting in more computer round-off error.

To find an "optimum DT", the program proceeds as follows:

1. Find the velocity components \( \dot{x}(1), \dot{y}(1), \dot{z}(1) \), using \( DT = H \).

2. Find \( \dot{x}(2), \dot{y}(2), \dot{z}(2) \), using \( DT = H/2 \) twice.

3. Calculate the relative errors in velocity:

\[
E = \left| \frac{\dot{x}(1) - \dot{x}(2)}{\dot{x}(2)} \right| , \text{ etc.}
\]

4. If \( E < \text{EPTINY} \), then DT is too small, and computer time would be wasted. Thus, DT is increased for the next integration interval.

5. If \( E > \text{EPBIG} \), then DT is too large, and the velocity error is unacceptable. Thus, the calculation is repeated using a smaller DT.

The choice of EPTINY and EPBIG is quite arbitrary. Good results have been obtained with EPTINY = \( 10^{-7} \) and EPBIG = \( 10^{-5} \).
4. GENERAL FLOW CHART

BEGIN

USE "BULK MEDIA CONVERSION ROUTINE" (BMC) TO TRANSFER INPUT DATA FROM CARDS TO TAPE

CALL INPUT TO READ GENERAL INPUT DATA FROM TAPE

COMPUTE INITIAL WEIGHT. CALL TABLE TO READ FIRST PHASE DATA.

CALL LAUNCH TO SIMULATE RAIL

NSKIP=0?

no

CALL ALFA TO SOLVE ANGLE OF ATTACK EQUATIONS USING RUNGE-KUTTA METHOD

yes

CALL ACCEL 3 TO COMPUTE ACCELERATION

CALL DATA 1 TO COMPUTE OUTPUT DATA
CALL SETSA TO STORE DATA TEMPORARILY
CALL OUT TO INTERPOLATE OUTPUT DATA AND TO STORE IN OUTPUT ARRAY

CALL RUNGE TO CALCULATE NEXT TRAJECTORY POINT AT T + DT

CALL DATA 1 TO COMPUTE OUTPUT DATA
CALL SETSA TO STORE DATA TEMPORARILY
1. TIME TO STOP CALCULATIONS
   - yes → 2
   - no → 3

2. ALTITUDE = 0?
   - yes → SET IEND = 1
   - no → STOP AT APOGEE?
     - yes → CALL OUT TO STORE OUTPUT DATA
     - no → 4

3. CALL OUT TO STORE OUTPUT DATA

4. CALL OUT TO STORE PRESENT VALUES OF OUTPUT DATA

5. TIME ≥ REGULAR PRINTOUT TIME?
   - yes → CALL OUT TO INTERPOLATE & PRINTOUT
   - no → END OF CALCULATIONS? (IEND = 1 ?)
     - yes → ANY SPENT STAGES ?
       - no → STOP
       - yes → 3
     - no → 3

6. END OF PREVIOUS PHASE?
   - yes → CALL OUT TO PRINTOUT
   - no → 4

7. HAS NEXT PHASE STARTED?
   - yes → CALL OUT TO PRINTOUT
   - no → 5

8. PHASE TIME CHECK; TIME TO BEGIN NEXT PHASE?
   - yes → CALL TABLE TO READ-IN NEXT PHASE DATA
   - no → 3
5. SUBROUTINE DESCRIPTION

ACCELO --- Subroutine ACCELO computes the relative linear acceleration and angular acceleration, assuming the rocket flies in the launch plane (2 dimensions) with an angle of attack.

ACCEL1 --- Subroutine ACCEL1 computes the three components of inertial acceleration which determine the main trajectory.

ACCEL2 --- Subroutine ACCEL2 computes the relative linear acceleration (one dimension) for motion along the launch rail, assuming there is no frictional force.

ACCEL3 --- Subroutine ACCEL3 computes the acceleration using elevation and azimuth rather than velocity components.

ALFA --- Subroutine ALFA determines the trajectory while the rocket flies with an angle of attack during the launch phase. Linear motion is constrained to the launch plane until the angle of attack goes to zero (or until the phase is nearly completed).

ATMSPH --- Subroutine ATMSPH computes temperature, pressure, viscosity, density, and speed of sound using an eighth degree polynomial approximation for the 1962 Standard Atmosphere data.
CNALFA --- Subroutine CNALFA interpolates to find the normal force coefficient and pitch damping coefficient which are used in the angle of attack equations.

DATAl --- Subroutine DATAl computes most of the output data.

DIRECT --- Subroutine DIRECT determines the direction of the thrust vector; the thrust is either lined up with velocity, or the thrust maintains a constant direction with respect to the inertial system. Normally, the first option is used; however, for spin stabilized rockets, the second option can be used at high altitudes. (NOTE: This subroutine is not called during the ALFA routine.)

IIP --- Subroutine IIP computes the instantaneous impact points assuming zero thrust and vacuum trajectory.

INPUT --- Subroutine INPUT reads data from input tape 05 and prints it.

LAUNCH --- Subroutine LAUNCH simulates a frictionless launch rail but includes aerodynamic drag in the equations of motion.

MAT --- Subroutine MAT calculates the elements of the transformation matrix for a coordinate system rotation. The inverse matrix is also calculated.
MATLCH --- Subroutine MATLCH calculates the transformation from the inertial system to the launch site system.

MATRAD --- Subroutine MATRAD calculates the transformation from the inertial system to the radar site system.

MATROC --- Subroutine MATROC calculates the transformation from the inertial system to the instantaneous topocentric system.

OUT --- Subroutine OUT interpolates all output data, stores it, and prints it out.

RUNGE --- Subroutine RUNGE integrates the equations of motion, using the fourth order Runge-Kutta method with variable step-size. Refer to Part 2, Section 3.

SETSA --- Subroutine SETSA transfers output data to a temporary storage array.

TABLE --- Subroutine TABLE reads in thrust, weight, and drag tables and phase messages.

TAB1 --- Subroutine TAB1 interpolates to find the drag coefficient.

TAB2 --- Subroutine TAB2 interpolates to find the thrust.

TAB3 --- Subroutine TAB3 interpolates to find the total weight.

TPOUT --- Subroutine TPOUT writes a special output tape with time interval controlled by NIIP. (See input description.)
TRANS1 --- Subroutine TRANS1 transforms from the inertial system to one of the rotating systems.

TRANS2 --- Subroutine TRANS2 transforms from a rotating system to the inertial system.
6. HYBRID 3-D INPUT DESCRIPTION.

Input cards must be in FORTRAN NAMELIST format or in the form of a message (see sample); all cards must be punched only in columns 2 through 72. Each input value must be followed by a comma, except the last item in a list, which is followed by a $.

1. The following lists must always be input: NAMQ, NAMR, NAMS, NAMT, NAMU, NAMV. Within a list, if a particular variable is not required, it should be omitted.

2. The lists NAM2, NAM3, NAM4, NAM5 may be input depending upon the trajectory and type of rocket. Thrust, weight, and drag table input is controlled by the phase time array, PHT, and the array NID.

3. All thrust, weight, and drag tables have a limit of 50 values and must have at least two values. The input arrays PHT and NID have a limit of 14 values; the arrays WTM and WTP have a limit of 6 values.

4. For a thrusting phase, the thrust, weight, and drag tables are preceded by one card containing a "phase message"; this may be any phrase to identify the phase; e.g. STAGE 1 THRUSTING. A phase message card must also precede the drag table for a coasting phase. The phase messages may be punched in column 2 through 72, but only the characters in column 2 through 7 will appear on the output (in column 2 through 7) (See sample input and output.)
5. To simulate the rocket launcher, set LENGTH equal to the length of the launch rail. This option restricts the motion to one dimension only.

6. To simulate angle of attack near lift-off, set NSKIP = 0. Motion will be in two dimensions until the angle of attack goes to zero; then, the program switches to three dimensions, with thrust and velocity vectors aligned. (Angle of attack can be calculated only during first stage thrusting.)

7. If NSKIP = 1, then the angle of attack routine is skipped, and the motion will be in three dimensions with thrust and velocity aligned.

8. The trajectory terminates (1) at altitude equal zero; (2) at apogee; (3) or at a given time. See the control constants NSTOP, TSTOP, KSTOP. If the trajectory is not stopped at apogee or at a given time, then it terminates when altitude equals zero.

9. For tape output, see control constant NIIP. In the deck for execution, include a tape control card with file code 11.
INPUT DESCRIPTION

FIRST CARD: title phrase in column 2 to 72

NAMQ List:
$NAMQ in column 2 through 6 followed by:

VT = initial speed (ft/s-c)
EL = initial flight elevation angle or launch elevation angle (c'-g.)
AZ = initial flight azimuth angle or launch azimuth angle (deg.)
NCEO = 0 if EL, AZ are geocentric angles
       1 if EL, AZ are geodetic angles
PHI = initial geodetic latitude (deg.)
LAMDA = initial longitude (deg.)
TIME = initial time (not less than 0.0), (sec)
       (launch time should equal zero)
ALT = initial altitude (ft.), (NOTE: launch altitude should equal 0.0 only)
LENGTH = length of launch rail (ft.)

NAMR List:
$NAMR in column 2 through 6 followed by:

PTI = print time interval (not less than .01), (sec)
NSPENT = number of spent stages to be run
NSTOP = 1 if TSTOP not equal 0.0 (otherwise, omit)
TSTOP = time at which calculations are terminated (sec.),
        (if not desired, omit)
NSKIP = 0 alpha routine calculations (beginning at launch)
        1 alpha routine skipped
KSTOP = 1 to stop calculations at apogee (otherwise, omit)
THCON = 1 for no thrust control
        2 for constant thrust direction beginning at time = T1 or at altitude ALTC
        (the constant direction is determined by the program)
$T1 = \text{time to begin constant thrust direction (sec.)}, \\
    \text{(if not used, omit)}$

$ALTC = \text{altitude where thrust control begins (ft.)}, \text{(if not used, omit)}$

$NSYS = \begin{cases} 
    1 & \text{for English output} \\
    2 & \text{for metric output} \\
    3 & \text{for both English and metric output} 
\end{cases}$

$NIIP = \begin{cases} 
    0, & \text{special tape time interval same as print time interval} \\
    1, & \text{tape time interval equal 0.1 sec., and PTI equal an integer multiple of 0.1 sec.} \\
    2, & \text{no tape output} 
\end{cases}$

$NPAGE = \begin{cases} 
    1 & \text{for output page A} \\
    2 & \text{for output page B} \\
    3 & \text{for output page C} \\
    4 & \text{for output page D} \\
    5 & \text{for output page E} \\
    6 & \text{for output page F} 
\end{cases}$

[e.g., if $NPAGE = 2, 4, 5$; pages B, D, E will be printed out]

$\text{NAMS List:}$

$\text{NAMS in column 2 through 6 followed by:}$

$PHIO = \text{launcher geodetic latitude (deg.)}$

$LAMDAO = \text{launcher longitude (deg.)}$

$PHI1 = \text{radar site geodetic latitude (deg.) (if not used, omit)}$

$LAMDAl = \text{radar site longitude (deg.) (if not used, omit)}$

$\text{NAMT List:}$

$\text{NAMT in column 2 through 6 followed by:}$

$\text{NPHS/PHT= phase time array (sec.) (e.g. 1st stage ignition time, 1st stage burnout time, 2nd stage ignition, 2nd stage burnout, etc.)}$
NTAB/NID = array of control constants for phase changes

1, to read in phase message only
2, to read phase message and drag table
3, to read in phase message and thrust, weight, and drag tables
4, nothing is read in

[NOTE: There must be a value of NID for each value of PHT, plus one additional value corresponding to apogee.]

NAMU List:

$NAMU in column 2 through 6 followed by:

NMT/WTM = list of inert rocket motor weights (lbs.) (includes miscellaneous weight)

WPL = payload weight (lbs.)

NSEP/TSEP = table of motor separation times (sec.)

[NOTE: For single stage rocket, omit]

[Each value of TSEP must appear in the PHT array.]

WTP = list of propellant weights (one weight for each stage), (lbs.)

NAMV List:

$NAMV in column 2 through 6 followed by:

LSM = distance from center of gravity to center of pressure (ft.)

INERT = moment of inertia (slug ft.²)

CNA = normal force coefficient (rad⁻¹)

NAM2 List: (thrust table, input if NID = 3)

$NAM2 in column 2 through 6 followed by:

AREA = rocket nozzle exit area (ft.²)

ILOPT = 1 for sea level thrust table,
2 for vacuum thrust table

TIM = time table corresponding to thrust table (sec.)

NTH/THS = thrust table (lbs.)
NAM3 List: (propellant weight table, input if NID = 3)
$NAM3 in column 2 through 6 followed by:
TWT = time table corresponding to weight table (sec.)
NWT/WPR = propellant weight table (lbs.)

NAM4 List: (drag data, input if NID = 2 or 3)
$NAM4 in column 2 through 6 followed by:
SAREA = drag area (ft.$^2$)
\[ (=1.0 \text{ when the area is included in the drag coefficient}) \]
MCH = table of Mach numbers corresponding to drag coefficient table
NCD/CDD = table of drag coefficients

NAM5 List: (input if NSPENT > 0)
$NAM5 in column 2 through 6 followed by:
PTI = print time interval for spent stage printout (sec.)
WTS = spent stage weight (lbs.)
NTAB/NID = array of phase control constants for spent stage trajectory:
\begin{itemize}
\item 1, to read phase message only
\item 2, to read phase message plus drag table
\item 4, nothing is read in
\end{itemize}

\textbf{NOTE:} there should be one or two values for NID--
one value for the start of the spent stage trajectory and a second value for apogee. If the trajectory begins after apogee, then only one value is needed.

Last card of input data:
Z99999 in column 7 through 12
7. HYBRID 3-D OUTPUT DESCRIPTION

Printed output can be in English and/or metric units by setting the input constant NSYS. In addition, there are six different page options which can be selected using the array NPAGE. Tape output is in English units only. Printout peculiarities are listed below:

1. If the launch rail simulation is used (when LENGTH > 0), output will be suppressed while the rocket moves along the rail.

2. At phase changes, there is double printout plus a phase message from the input data.

3. Built-in messages include:
   - **TSTOP**, printed when the trajectory is terminated at a given time.
   - **APOGEE**, printed out near the actual apogee.
   - **ALT=0**, printed at the end of the trajectory when the altitude equals zero.

4. IIP output is suppressed for spent stage trajectories.

5. Above the atmosphere limit (400,000 ft.), the MACH number will be blank.

Notes on Metric Units:

- nt = newton  lbs. = pounds  1 ft. = 0.3048 m
- kg = kilogram  ft. = feet  1 N.M. = 1.852 km
- m = meter  N.M. = nautical mile  1 lb. = 4.4482 nt

1 slug = 14.594 kg
1 lb./ft² = 47.880 nt/m²
PAGE A:
TIME = time from launch (sec.)
ALPHA = angle of attack (deg.)
TH EL = thrust elevation angle (deg.)
FL EL = velocity elevation angle (deg.)
FL AZ = velocity azimuth angle (deg.)
ALT = altitude (ft.)
RANGE = surface range (N,M.)
LAT GD = geodetic latitude (deg.)
LONG = longitude (deg.)

PAGE B:
TIME = time from launch (sec.)
THRUST = rocket thrust (lbs.)
WEIGHT = rocket weight (lbs.)
DRAG = aerodynamic drag (lbs.)
MACH = Mach number (suppressed when altitude > 400,000 ft.)
DYN PR = dynamic pressure (lb./ft.²)
REL ACC = acceleration relative to earth (ft./sec.²)
REL VEL = velocity relative to earth (ft./sec.)
MASS = mass of rocket (slugs)

PAGE C: Launch Site Coordinate System
TIME = time from launch (sec.)
XL,YL,ZL = position coordinates (ft.)
VXL,VYL, VZL = velocity components (ft./sec.)
RXY = tangent plane range (ft.)
GAML = velocity elevation angle with respect to tangent plane (deg.)
VEL-L = velocity with respect to launch site (ft./sec.)
RLDOT = rate of change of slant range
PAGE D: Radar Site Coordinate System

| TIME    | time from launch (sec.) |
| SL RANGE | slant range from radar (N.M.) |
| LOOK AZ  | look azimuth (deg.) |
| LOOK EL  | look elevation angle (deg.) |
| VEL     | velocity relative to radar (ft./sec.) |
| EAST    | distance to the East (+) or West (-) in the tangent plane (N.M.) |
| NORTH   | distance to the North (+) or South (-) in the tangent plane (N.M.) |
| VERT    | distance (+) above the tangent plane (N.M.) |
| VEL-E   | East (+) or West (-) component of velocity (ft./sec.) |
| VEL-N   | North (+) or South (-) component of velocity (ft./sec.) |
| VEL-V   | vertical component of velocity (ft./sec.) |

PAGE E: Instantaneous Impact Points (Vacuum Trajectory)
[NOTE: Page E is suppressed for spent stages]

| TIME    | time from launch (sec.) |
| LAT GD  | geodetic latitude of IIP (deg.) |
| LONG    | longitude of IIP (deg.) |
| RANGE   | range of IIP from launch site (N.M.) |
| IP TIME | time from launch to instantaneous impact (sec.) |

PAGE F: Inertial Coordinate System

| TIME    | time from launch (sec.) |
| R       | distance from earth center (ft.) |
| VEL     | inertial velocity (ft./sec.) |
| ACCEL   | inertial acceleration (ft./sec.²) |
SPECIAL TAPE OUTPUT

This tape is high density (800 BPI) and contains 71 words per record, as listed below (those not listed are equal to zero):

<table>
<thead>
<tr>
<th>Word Number</th>
<th>Contents (English Units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TIME</td>
</tr>
<tr>
<td>2</td>
<td>WEIGHT</td>
</tr>
<tr>
<td>3</td>
<td>THRUST</td>
</tr>
<tr>
<td>4</td>
<td>VEL (inertial)</td>
</tr>
<tr>
<td>5</td>
<td>R (inertial)</td>
</tr>
<tr>
<td>6</td>
<td>ALT</td>
</tr>
<tr>
<td>7</td>
<td>RANGE</td>
</tr>
<tr>
<td>8</td>
<td>MACH</td>
</tr>
<tr>
<td>9</td>
<td>Q</td>
</tr>
<tr>
<td>10</td>
<td>LAT GD</td>
</tr>
<tr>
<td>12</td>
<td>LONG</td>
</tr>
<tr>
<td>14</td>
<td>ALPHA</td>
</tr>
<tr>
<td>18</td>
<td>FL EL</td>
</tr>
<tr>
<td>20</td>
<td>FL AZ</td>
</tr>
<tr>
<td>23</td>
<td>TH EL</td>
</tr>
<tr>
<td>37</td>
<td>REL VEL</td>
</tr>
<tr>
<td>55, 56, 57</td>
<td>XL, YL, ZL</td>
</tr>
<tr>
<td>58, 59, 60</td>
<td>VXL, VYL, VZL</td>
</tr>
<tr>
<td>61</td>
<td>RXY</td>
</tr>
<tr>
<td>62</td>
<td>GAML</td>
</tr>
<tr>
<td>63</td>
<td>VEL-L</td>
</tr>
<tr>
<td>64</td>
<td>RLDOT</td>
</tr>
<tr>
<td>65</td>
<td>IP TIME (IIP)</td>
</tr>
<tr>
<td>66</td>
<td>LAT GD (IIP)</td>
</tr>
<tr>
<td>67</td>
<td>LONG (IIP)</td>
</tr>
<tr>
<td>68</td>
<td>RANGE (IIP)</td>
</tr>
</tbody>
</table>
8. Deck Set-Up with Object Program on Tape

```bash
$ IDENT 153200, HYBRID 3D
$ EXECUTE
$ LIMITS 100, 30K, 0, 10K
$ TAPE R, X1D, 3969
$ DATA 05
$ INCODE IBMF
(data cards)
$ ENDJOB
***EOF
```

Deck Set-Up with Object Program on Cards

```bash
$ IDENT 153200, HYBRID 3D
$ OPTION FORTRAN
(object deck)
$ EXECUTE
$ LIMITS 100, 30K, 0, 10K
$ DATA 05
$ INCODE IBMF
(data cards)
$ ENDJOB
***EOF
```
9. SAMPLE INPUT AND OUTPUT

This is a Nike-Cajun trajectory simulation using a 2-inch launch rail and initial velocity equal to zero. Four phases are shown: stage 1 thrusting, stage 2 coasting, stage 2 thrusting, and stage 2 coasting. Input tables must be in the order specified by the NTAB/NID array:

3 for stage 1 thrust; weight, drag input (NAM2, NAM3, NAM4)
2 for stage 2 drag input for coasting (NAM4)
3 for stage 2 thrust; weight, drag input (NAM2, NAM3, NAM4)
2 for stage 2 drag input for coasting (NAM4)
4 for no input at apogee

The beginning of each phase is given by the NPHS/PHT array:
-0.017 sec for stage 1 ignition
3.523 for stage 1 burnout (and separation)
17.0 for stage 2 ignition
22.0 for stage 2 burnout

Note that the times used for the thrust and weight tables are relative times which are added to the phase times by the program. If the first stage thrust table begins with a value less than the lift-off weight, then the ignition time must be adjusted so that lift-off will occur at zero time.
NIK-F-CAJUN TRAJECTORY SIMULATION

<table>
<thead>
<tr>
<th>SMAMO</th>
<th>VT = 0.0, FL = 98.0, AZ = 120.0, NGFO = 1,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PHI = 37.8480, LAMDA = -75.4736, TIME = 0.0, ALT = 0.0</td>
</tr>
<tr>
<td></td>
<td>LENGTH = 0.1667</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SMAMR</th>
<th>PIJ = 0.5, NSPENT = 0, NSTOP = 1, STOP = 25.0, NSKIP = 0,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KSTOP = 0, THCON = 1, TSTOP = 0.0, ALT = 0.0,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SMAMS</th>
<th>PH1 = 37.8412, LAMDA = -75.4736,</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMAMT</td>
<td>NPHS = 0.0, 9.523, 17.0, 22.0,</td>
</tr>
<tr>
<td></td>
<td>NTAB = 1, NTAB = 2,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SMAMU</th>
<th>NMT = 64.0, 97.4, WPL = 60.0,</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMAMV</td>
<td>LSM = 2.38, INFRT = 1500.0, CNA = 15.47</td>
</tr>
</tbody>
</table>

1ST STAGE, THRUSTERING

<table>
<thead>
<tr>
<th>SMAM2</th>
<th>AREA = 1.5, ID = 1,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM = 0.0, 0.01, 0.04, 0.05, 0.09,</td>
</tr>
<tr>
<td></td>
<td>0.15, 0.84, 1.14, 1.74, 2.04,</td>
</tr>
<tr>
<td></td>
<td>2.34, 2.69, 2.64, 2.79, 2.99,</td>
</tr>
<tr>
<td></td>
<td>3.09, 3.21, 3.28, 3.34, 3.40,</td>
</tr>
<tr>
<td></td>
<td>3.46, 2.54,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NTM/THS = 1024.0, 264.9, 9941.7, 3616.8, 42397.9,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>42580.0, 43263.0, 45351.0, 44705.0, 45667.0,</td>
</tr>
<tr>
<td></td>
<td>46820.0, 46472.0, 45570.0, 42974.0, 37974.0,</td>
</tr>
<tr>
<td></td>
<td>33648.0, 24034.0, 16342.0, 10864.0, 6538.0,</td>
</tr>
<tr>
<td></td>
<td>3268.0, 205.0,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SMAM3</th>
<th>TWT = 0.0, 0.01, 0.04, 0.05, 0.09,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.15, 0.84, 1.14, 1.74, 2.04,</td>
</tr>
<tr>
<td></td>
<td>2.34, 2.49, 2.64, 2.79, 2.99,</td>
</tr>
<tr>
<td></td>
<td>3.09, 3.21, 3.28, 3.34, 3.40,</td>
</tr>
<tr>
<td></td>
<td>3.46, 2.54,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NWT/WPR = 738.0, 737.26, 732.10, 729.98, 721.20,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>707.87, 553.07, 493.54, 446.83, 275.78,</td>
</tr>
<tr>
<td></td>
<td>203.7, 162.60, 130.42, 95.72, 53.41,</td>
</tr>
<tr>
<td></td>
<td>34.69, 16.60, 9.22, 4.95, 2.22,</td>
</tr>
<tr>
<td></td>
<td>48, 4005,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SMAM4</th>
<th>SARE = 1.4741,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCH = 0.0, 0.75, 1.00, 1.20, 1.60,</td>
</tr>
<tr>
<td></td>
<td>2.0, 2.40, 2.80, 3.30, 4.00,</td>
</tr>
<tr>
<td></td>
<td>5.0,</td>
</tr>
</tbody>
</table>
### COAST STAGE 2 COASTING

<table>
<thead>
<tr>
<th>$\text{SNAM4}$</th>
<th>SARFA</th>
<th>.23</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCH</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>NCD/CDD</td>
<td>.50</td>
<td>.60</td>
</tr>
<tr>
<td></td>
<td>.70</td>
<td>.80</td>
</tr>
</tbody>
</table>

### 2ND TH STAGE 2 THRUSTING

<table>
<thead>
<tr>
<th>$\text{SNAM2}$</th>
<th>ARFA</th>
<th>.223</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM</td>
<td>0.0</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>.08</td>
<td>.15</td>
</tr>
<tr>
<td></td>
<td>.50</td>
<td></td>
</tr>
</tbody>
</table>

### $\text{SNAM3}$ TWT

|                | 0.0   | 0.04 |
|                | .08   | .15  |
|                | .50   |     |

### NWT/WPP

|                | 119.0 | 118.80 |
|                | 117.91 | 115.50 |
|                | 103.73 |

### $\text{SNAM4}$ SARFA

<table>
<thead>
<tr>
<th></th>
<th>.23</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCH</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>5.0</td>
</tr>
<tr>
<td>NCD/CDD</td>
<td>.50</td>
</tr>
<tr>
<td></td>
<td>.60</td>
</tr>
<tr>
<td></td>
<td>.70</td>
</tr>
</tbody>
</table>

### COAST STAGE 2 COASTING

<table>
<thead>
<tr>
<th>$\text{SNAM4}$</th>
<th>SARFA</th>
<th>.23</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCH</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>NCD/CDD</td>
<td>.50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>.60</td>
<td></td>
</tr>
</tbody>
</table>

### Z00090

<p>|                | .42   |
|                | .36   |</p>
<table>
<thead>
<tr>
<th>TIME (SEC)</th>
<th>ALPHA (DEG)</th>
<th>TH EL (DEG)</th>
<th>FL EL (DEG)</th>
<th>FL: AZ (DEG)</th>
<th>ALT (FT)</th>
<th>RANGE (NM)</th>
<th>LAT GD (DEG)</th>
<th>LONG GD (DEG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>0.295</td>
<td>90.038</td>
<td>77.743</td>
<td>119.076</td>
<td>104.0</td>
<td>0.05</td>
<td>37.8489</td>
<td>75.4735</td>
</tr>
<tr>
<td>1.00</td>
<td>0.0</td>
<td>79.524</td>
<td>79.624</td>
<td>119.071</td>
<td>432.0</td>
<td>0.01</td>
<td>37.8479</td>
<td>75.4734</td>
</tr>
<tr>
<td>1.50</td>
<td>0.0</td>
<td>79.480</td>
<td>79.480</td>
<td>119.079</td>
<td>1002.0</td>
<td>0.03</td>
<td>37.8479</td>
<td>75.4730</td>
</tr>
<tr>
<td>2.00</td>
<td>0.0</td>
<td>79.352</td>
<td>79.182</td>
<td>119.079</td>
<td>1832.0</td>
<td>0.06</td>
<td>37.8478</td>
<td>75.4726</td>
</tr>
<tr>
<td>2.50</td>
<td>0.0</td>
<td>79.359</td>
<td>79.307</td>
<td>119.129</td>
<td>2045.0</td>
<td>0.09</td>
<td>37.8473</td>
<td>75.4721</td>
</tr>
<tr>
<td>3.00</td>
<td>0.0</td>
<td>79.253</td>
<td>79.253</td>
<td>119.143</td>
<td>4358.0</td>
<td>0.13</td>
<td>37.8469</td>
<td>75.4711</td>
</tr>
<tr>
<td>3.50</td>
<td>0.0</td>
<td>79.204</td>
<td>79.204</td>
<td>119.157</td>
<td>5978.0</td>
<td>0.18</td>
<td>37.8465</td>
<td>75.4702</td>
</tr>
<tr>
<td>4.00</td>
<td>0.0</td>
<td>79.202</td>
<td>79.202</td>
<td>119.158</td>
<td>6053.0</td>
<td>0.19</td>
<td>37.8465</td>
<td>75.4702</td>
</tr>
<tr>
<td>4.50</td>
<td>0.0</td>
<td>79.135</td>
<td>79.159</td>
<td>119.171</td>
<td>7576.0</td>
<td>0.24</td>
<td>37.8461</td>
<td>75.4697</td>
</tr>
<tr>
<td>5.00</td>
<td>0.0</td>
<td>79.103</td>
<td>79.103</td>
<td>119.185</td>
<td>9130.0</td>
<td>0.28</td>
<td>37.8457</td>
<td>75.4684</td>
</tr>
<tr>
<td>5.50</td>
<td>0.0</td>
<td>79.050</td>
<td>79.050</td>
<td>119.190</td>
<td>10639.0</td>
<td>0.38</td>
<td>37.8449</td>
<td>75.4675</td>
</tr>
<tr>
<td>6.00</td>
<td>0.0</td>
<td>78.975</td>
<td>78.999</td>
<td>119.212</td>
<td>11039.0</td>
<td>0.42</td>
<td>37.8443</td>
<td>75.4666</td>
</tr>
<tr>
<td>6.50</td>
<td>0.0</td>
<td>78.906</td>
<td>78.936</td>
<td>119.226</td>
<td>11353.0</td>
<td>0.47</td>
<td>37.8442</td>
<td>75.4659</td>
</tr>
<tr>
<td>7.00</td>
<td>0.0</td>
<td>78.839</td>
<td>78.879</td>
<td>119.240</td>
<td>11926.0</td>
<td>0.51</td>
<td>37.8438</td>
<td>75.4650</td>
</tr>
<tr>
<td>7.50</td>
<td>0.0</td>
<td>78.775</td>
<td>78.795</td>
<td>119.267</td>
<td>11624.0</td>
<td>0.56</td>
<td>37.8435</td>
<td>75.4642</td>
</tr>
<tr>
<td>8.00</td>
<td>0.0</td>
<td>78.690</td>
<td>78.690</td>
<td>119.281</td>
<td>18012.0</td>
<td>0.60</td>
<td>37.8431</td>
<td>75.4634</td>
</tr>
<tr>
<td>8.50</td>
<td>0.0</td>
<td>78.624</td>
<td>78.624</td>
<td>119.299</td>
<td>20163.0</td>
<td>0.64</td>
<td>37.8428</td>
<td>75.4626</td>
</tr>
<tr>
<td>9.00</td>
<td>0.0</td>
<td>78.556</td>
<td>78.554</td>
<td>119.308</td>
<td>21427.0</td>
<td>0.68</td>
<td>37.8424</td>
<td>75.4618</td>
</tr>
<tr>
<td>9.50</td>
<td>0.0</td>
<td>78.485</td>
<td>78.485</td>
<td>119.322</td>
<td>22944.0</td>
<td>0.72</td>
<td>37.8421</td>
<td>75.4611</td>
</tr>
<tr>
<td>10.0</td>
<td>0.0</td>
<td>78.413</td>
<td>78.413</td>
<td>119.338</td>
<td>23842.0</td>
<td>0.76</td>
<td>37.8418</td>
<td>75.4603</td>
</tr>
<tr>
<td>10.5</td>
<td>0.0</td>
<td>78.339</td>
<td>78.339</td>
<td>119.349</td>
<td>25014.0</td>
<td>0.80</td>
<td>37.8415</td>
<td>75.4596</td>
</tr>
<tr>
<td>11.0</td>
<td>0.0</td>
<td>78.253</td>
<td>78.263</td>
<td>119.363</td>
<td>26165.0</td>
<td>0.84</td>
<td>37.8411</td>
<td>75.4588</td>
</tr>
<tr>
<td>11.5</td>
<td>0.0</td>
<td>78.155</td>
<td>78.185</td>
<td>119.377</td>
<td>27294.0</td>
<td>0.88</td>
<td>37.8408</td>
<td>75.4581</td>
</tr>
<tr>
<td>12.0</td>
<td>0.0</td>
<td>78.105</td>
<td>78.105</td>
<td>119.390</td>
<td>28407.0</td>
<td>0.92</td>
<td>37.8405</td>
<td>75.4574</td>
</tr>
<tr>
<td>12.5</td>
<td>0.0</td>
<td>78.023</td>
<td>78.023</td>
<td>119.404</td>
<td>29498.0</td>
<td>0.96</td>
<td>37.8402</td>
<td>75.4567</td>
</tr>
<tr>
<td>13.0</td>
<td>0.0</td>
<td>77.930</td>
<td>77.930</td>
<td>119.417</td>
<td>30572.0</td>
<td>0.99</td>
<td>37.8399</td>
<td>75.4560</td>
</tr>
<tr>
<td>13.5</td>
<td>0.0</td>
<td>77.853</td>
<td>77.853</td>
<td>119.433</td>
<td>31657.0</td>
<td>1.03</td>
<td>37.8396</td>
<td>75.4553</td>
</tr>
<tr>
<td>14.0</td>
<td>0.0</td>
<td>77.765</td>
<td>77.769</td>
<td>119.444</td>
<td>32667.0</td>
<td>1.07</td>
<td>37.8393</td>
<td>75.4548</td>
</tr>
<tr>
<td>TIME (UTC)</td>
<td>ALPHA (DEG)</td>
<td>TH EL (DGR)</td>
<td>FL EL (DEG)</td>
<td>FL' AZ (DEG)</td>
<td>ALT (FT)</td>
<td>RANGE (NM)</td>
<td>LAT GD (DEG)</td>
<td>LONG (DEG)</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>14.00</td>
<td>0.00</td>
<td>77.679</td>
<td>77.679</td>
<td>119.458</td>
<td>33689</td>
<td>1.11</td>
<td>37.8390</td>
<td>75.4533</td>
</tr>
<tr>
<td>15.00</td>
<td>0.00</td>
<td>77.587</td>
<td>77.587</td>
<td>119.471</td>
<td>34696</td>
<td>1.14</td>
<td>37.6387</td>
<td>75.4526</td>
</tr>
<tr>
<td>16.00</td>
<td>0.00</td>
<td>77.488</td>
<td>77.488</td>
<td>119.489</td>
<td>35688</td>
<td>1.16</td>
<td>37.6384</td>
<td>75.4520</td>
</tr>
<tr>
<td>17.00</td>
<td>0.00</td>
<td>77.391</td>
<td>77.391</td>
<td>119.490</td>
<td>36665</td>
<td>1.21</td>
<td>37.6381</td>
<td>75.4513</td>
</tr>
<tr>
<td>18.00</td>
<td>0.00</td>
<td>77.293</td>
<td>77.293</td>
<td>119.511</td>
<td>37627</td>
<td>1.25</td>
<td>37.6378</td>
<td>75.4507</td>
</tr>
<tr>
<td>19.00</td>
<td>0.00</td>
<td>77.191</td>
<td>77.191</td>
<td>119.529</td>
<td>38576</td>
<td>1.28</td>
<td>37.6375</td>
<td>75.4500</td>
</tr>
<tr>
<td>20.00</td>
<td>0.00</td>
<td>77.090</td>
<td>77.090</td>
<td>119.538</td>
<td>39605</td>
<td>1.32</td>
<td>37.6372</td>
<td>75.4493</td>
</tr>
<tr>
<td>21.00</td>
<td>0.00</td>
<td>77.321</td>
<td>77.321</td>
<td>119.550</td>
<td>40844</td>
<td>1.37</td>
<td>37.6368</td>
<td>75.4485</td>
</tr>
<tr>
<td>22.00</td>
<td>0.00</td>
<td>76.758</td>
<td>76.758</td>
<td>119.562</td>
<td>42358</td>
<td>1.43</td>
<td>37.6364</td>
<td>75.4479</td>
</tr>
<tr>
<td>23.00</td>
<td>0.00</td>
<td>76.028</td>
<td>76.028</td>
<td>119.574</td>
<td>44519</td>
<td>1.50</td>
<td>37.6359</td>
<td>75.4462</td>
</tr>
<tr>
<td>24.00</td>
<td>0.00</td>
<td>75.295</td>
<td>75.295</td>
<td>119.596</td>
<td>46295</td>
<td>1.56</td>
<td>37.6351</td>
<td>75.4447</td>
</tr>
<tr>
<td>25.00</td>
<td>0.00</td>
<td>74.575</td>
<td>74.575</td>
<td>119.610</td>
<td>48022</td>
<td>1.62</td>
<td>37.6343</td>
<td>75.4439</td>
</tr>
<tr>
<td>26.00</td>
<td>0.00</td>
<td>73.964</td>
<td>73.964</td>
<td>119.621</td>
<td>51651</td>
<td>1.70</td>
<td>37.6334</td>
<td>75.4430</td>
</tr>
<tr>
<td>27.00</td>
<td>0.00</td>
<td>73.322</td>
<td>73.322</td>
<td>119.633</td>
<td>54495</td>
<td>1.78</td>
<td>37.6325</td>
<td>75.4429</td>
</tr>
<tr>
<td>28.00</td>
<td>0.00</td>
<td>72.701</td>
<td>72.701</td>
<td>119.645</td>
<td>57332</td>
<td>1.86</td>
<td>37.6316</td>
<td>75.4429</td>
</tr>
<tr>
<td>29.00</td>
<td>0.00</td>
<td>72.089</td>
<td>72.089</td>
<td>119.656</td>
<td>60164</td>
<td>2.00</td>
<td>37.6307</td>
<td>75.4429</td>
</tr>
<tr>
<td>30.00</td>
<td>0.00</td>
<td>71.499</td>
<td>71.499</td>
<td>119.666</td>
<td>62900</td>
<td>2.11</td>
<td>37.6307</td>
<td>75.4429</td>
</tr>
<tr>
<td>31.00</td>
<td>0.00</td>
<td>70.926</td>
<td>70.926</td>
<td>119.676</td>
<td>65777</td>
<td>2.22</td>
<td>37.6298</td>
<td>75.4429</td>
</tr>
<tr>
<td>32.00</td>
<td>0.00</td>
<td>70.364</td>
<td>70.364</td>
<td>119.680</td>
<td>68640</td>
<td>2.33</td>
<td>37.6289</td>
<td>75.4429</td>
</tr>
<tr>
<td>33.00</td>
<td>0.00</td>
<td>79.804</td>
<td>79.804</td>
<td>119.680</td>
<td>71507</td>
<td>2.44</td>
<td>37.8260</td>
<td>75.4429</td>
</tr>
<tr>
<td>34.00</td>
<td>0.00</td>
<td>79.337</td>
<td>79.337</td>
<td>119.703</td>
<td>74316</td>
<td>2.55</td>
<td>37.8271</td>
<td>75.4429</td>
</tr>
<tr>
<td>35.00</td>
<td>0.00</td>
<td>78.864</td>
<td>78.864</td>
<td>119.715</td>
<td>77127</td>
<td>2.66</td>
<td>37.8262</td>
<td>75.4429</td>
</tr>
<tr>
<td>36.00</td>
<td>0.00</td>
<td>78.390</td>
<td>78.390</td>
<td>119.715</td>
<td>77127</td>
<td>2.76</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>37.00</td>
<td>0.00</td>
<td>77.916</td>
<td>77.916</td>
<td>119.715</td>
<td>77127</td>
<td>2.86</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>38.00</td>
<td>0.00</td>
<td>77.432</td>
<td>77.432</td>
<td>119.715</td>
<td>77127</td>
<td>2.96</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>39.00</td>
<td>0.00</td>
<td>76.948</td>
<td>76.948</td>
<td>119.715</td>
<td>77127</td>
<td>3.06</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>40.00</td>
<td>0.00</td>
<td>76.464</td>
<td>76.464</td>
<td>119.715</td>
<td>77127</td>
<td>3.16</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>41.00</td>
<td>0.00</td>
<td>75.980</td>
<td>75.980</td>
<td>119.715</td>
<td>77127</td>
<td>3.26</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>42.00</td>
<td>0.00</td>
<td>75.496</td>
<td>75.496</td>
<td>119.715</td>
<td>77127</td>
<td>3.36</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>43.00</td>
<td>0.00</td>
<td>74.012</td>
<td>74.012</td>
<td>119.715</td>
<td>77127</td>
<td>3.46</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>44.00</td>
<td>0.00</td>
<td>73.528</td>
<td>73.528</td>
<td>119.715</td>
<td>77127</td>
<td>3.56</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
<tr>
<td>45.00</td>
<td>0.00</td>
<td>73.044</td>
<td>73.044</td>
<td>119.715</td>
<td>77127</td>
<td>3.66</td>
<td>37.8253</td>
<td>75.4423</td>
</tr>
</tbody>
</table>
**FORMAT**(5X,12A6) /
DIMENSION TITLE(12) /
DIMENSION TSLER(9),HALTB(9),TFMPRI(9),PRSBD(9) /
DIMENSION SMAT(3,3),TMAT(3,3),S(9,6),ST(20,10)
DIMENSION SPS(15,5) /
DIMENSION SAL(10,2,10),ARRAY(10,30,10)
DIMENSION PHT(14),NID(14),WTM(6),TSEP(6),WTP(6)
DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50)
DIMENSION MFS(30)
DIMENSION NPAGE(10)
DIMENSION CON(10,7)

INTEGER THCON
REAL TILITL,TILON,ITOL,TITIM
REAL LAMDA,MASS,LOW,LOWG,LAT,MACH,MCH,LAMDA,LMAD1,LMAD
REAL KI
REAL LENGTH
REAL LSM,INFRT
REAL NRAD,NLCH,NDRAD,NDLCH
DOUBLE PRECISION CES
EXTERNAL ACCEL,ACCEL0,ACCEL3

NAMFILST /NAMO/VT,FL,A2,NGF0,PHT,LAMDA,TIME,ALT,LENGTH
NAMFILST /NAMR/PTI,NSFNT,NISTOP,NSKIP,KSTOP,
1 THFON,TI,ALT,CNSYS,NTP,NPAGE
NAMFILST /NAMS/PHTO,LAMDA0,PHI1,LAMDA1
NAMFILST /NAMT/NPHS,PHT,NTAR,NTD
NAMFILST /NAMV/NMT,WTM,WP,WPL,NSWP,TSEP,WTMP
NAMFILST /NAMV/LSM,INFRT,CNA
NAMFILST /NAMS/PTI,WT,NTP,NTAR,NTD

DATA BLK/0202020
DATA APO/6HAPGEE,TST/SHTOPE/ALTO/SHALT=0 /

C

C

C

 INITIALIZE CONSTANTS
PI = 3.14159265
OMEGA = 7.29211F-5
CONV = PI/180.
PI2 = 2.*PI
TO = 0
PO = 2115.666
EPSI = 5.0E-7
EPS1G = 1.0E-5
LOW = 1.0E-4
GM = 1.4076576F16
RE = 20899262. 
<table>
<thead>
<tr>
<th>RA</th>
<th>20925741.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB</td>
<td>20855591.</td>
</tr>
<tr>
<td>GO</td>
<td>32.174</td>
</tr>
<tr>
<td>ALTMS</td>
<td>2.0E+0</td>
</tr>
<tr>
<td>LINF</td>
<td>0</td>
</tr>
<tr>
<td>IFNO</td>
<td>0</td>
</tr>
<tr>
<td>NCYC</td>
<td>0</td>
</tr>
<tr>
<td>NLIN</td>
<td>0</td>
</tr>
<tr>
<td>KPAR</td>
<td>0</td>
</tr>
<tr>
<td>KON</td>
<td>1</td>
</tr>
<tr>
<td>KSPENT</td>
<td>0</td>
</tr>
<tr>
<td>NAP</td>
<td></td>
</tr>
<tr>
<td>LSTEP</td>
<td>0</td>
</tr>
<tr>
<td>DO 141</td>
<td>J=1,14</td>
</tr>
<tr>
<td>PHT(J)</td>
<td>9000000.</td>
</tr>
<tr>
<td>141 CONT</td>
<td></td>
</tr>
<tr>
<td>DO 142</td>
<td>J=1,10</td>
</tr>
<tr>
<td>NPAGE(J)</td>
<td>8</td>
</tr>
<tr>
<td>142 CONT</td>
<td></td>
</tr>
<tr>
<td>PHAS</td>
<td>BLKK</td>
</tr>
<tr>
<td>DO 14</td>
<td>I=1,10</td>
</tr>
<tr>
<td>DO 14</td>
<td>J=1,7</td>
</tr>
<tr>
<td>CON(I,J)</td>
<td>1.0</td>
</tr>
<tr>
<td>140 CONT</td>
<td></td>
</tr>
<tr>
<td>CON(5,1)</td>
<td>3048</td>
</tr>
<tr>
<td>CON(6,1)</td>
<td>1.852</td>
</tr>
<tr>
<td>CON(1,2)</td>
<td>4.4482</td>
</tr>
<tr>
<td>CON(2,2)</td>
<td>4.4482</td>
</tr>
<tr>
<td>CON(3,2)</td>
<td>4.4482</td>
</tr>
<tr>
<td>CON(5,2)</td>
<td>47.88</td>
</tr>
<tr>
<td>CON(6,2)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(7,2)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(8,2)</td>
<td>14.594</td>
</tr>
<tr>
<td>CON(1,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(2,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(3,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(4,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(5,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(6,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(7,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(9,3)</td>
<td>30.048</td>
</tr>
<tr>
<td>CON(1,4)</td>
<td>1.852</td>
</tr>
<tr>
<td>CON(4,4)</td>
<td>30.048</td>
</tr>
</tbody>
</table>
CON(5, 4) = 1.852
CON(6, 4) = 1.852
CON(7, 4) = 1.852
CON(8, 4) = 0.3048
CON(9, 4) = 0.3048
CON(10, 4) = 0.3048
CON(11, 4) = 1.852
CON(11, 6) = 0.3048
CON(13, 6) = 0.3048
CON(14, 6) = 0.3048

C
C READ GENERAL INPUT DATA FROM TAPE

CALL INPUT

PRINT 12
READ(5, 11) TITLE
READ(5, 1) AM1
READ(5, 1) AM2
READ(5, 1) AM3
READ(5, 1) AM4
READ(5, 1) AM5
READ(5, 1) AM6
READ(5, 1) AM7
PRINT 13, TITLE
PRINT 10, EL, AZ, WPL
STPTI = PTI
IF(N1P .EQ. 2) NTAP = 2
IF(N1P .EQ. 0 .OR. N1P .EQ. 2) GO TO 200
NCYC = 10 * PTI
PTI = .7

200 CONTINUE
HIGH = PTI/2.
DTO = PTI/8.
DTO = AMIN1(0.1, DTO)
DL = DTO
EL = EL * CONV
AZ = AZ * CONV
PHI0 = PHI0 * CONV
EPS = PHI0
PHI1 = PHI1 * CONV
PHI2 = PHI2 * CONV
LAMMA = LAMMA * CONV
LAMMA = LAMMA * CONV
CAZ = COS(AZ)
SAZ = SIN(AZ)
C GEOCENTRIC TO GEOCENTRIC CONVERSIONS
ARC = (RR/RA)**2 * SIN(PHI0) / COS(PHI0)
PHI0 = ATAN(ARC)
FPS = FPS - PHI0
APG = (RR/RA)**2 * SIN(PHI1) / COS(PHI1)
PHI1 = ATAN(APG)
ARC = (RR/RA)**2 * SIN(PHI1) / COS(PHI1)
PHI1 = ATAN(APG)
SEL = SIN(EL)
SELGD = SEL
CFL = COS(EL)
IF (NGEO * FO. 0) GO TO 250
CEPS = COS(EPS)
SEPS = SIN(EPS)
SFL = SEL * FPS - SEPS * CFL * CAZ
EL = ATAN2(SFL * SQRT(1.0 - SFL**2))
AZ = ATAN2(CFL * CAZ, CEPS * CFL * CAZ + SEPS * SELGD)
IF (AZ * LT. 0.0) AZ = AZ + PI2
CAZ = COS(AZ)
SAZ = SIN(AZ)
250 CONTINUE
C INITIALIZE ROCKET MOTOR WEIGHT
IF (NMT * FO. 1) GO TO 316
A SUM = .0
DO 31 J = 1, NMT
SUM = SUM + WTM(J)
310 CONTINUE
WPM = SUM
NMOT = 1
C C INITIALIZE WEIGHT OF PROPELLANT
SUM = .0
DO 315 J = 1, NMT
SUM = SUM + WTP(J)
315 CONTINUE
WPP2 = SUM
GO TO 317
316 WPP2 = WTP(1)
WPM = WTM(1)
NMOT = 1
317 CONTINUE
KDFL = 1
C
C \n\text{READ FIRST PHASE DATA} \n\text{CALL TABLEF} \n\text{NP} = \text{NP} + \text{NP} \n\text{KP} = \text{KP} + \text{KP} \n\text{SPH} = \text{S}\text{IN(PHI)} \n\text{CPH} = \text{C}\text{OS(PHI)} \n\text{SPH} = \text{S}\text{IN(GHI)} \n\text{CPH} = \text{C}\text{OS(GHI)} \n\text{X} = \text{RF} \text{CPHI} \text{CS} \text{(LAMDA)} \n\text{Y} = \text{RF} \text{CPHI} \text{SI} \text{(LAMDA)} \n\text{Z} = \text{RF} \text{SPHI} \n\text{SL} = \text{SI} \text{(LAMDA)} \n\text{CL} = \text{CO} \text{(LAMDA)} \n\text{SPHI} = \text{SO} \text{(PHI)} \n\text{CPI} = \text{CO} \text{(PHI)} \n\text{LONG} = \text{LAMDA} \n\text{CALL LAUNCH} \n\text{VX} = \text{VT} \text{CO} \text{(EL)} \text{CO} \text{(A)} \n\text{VY} = \text{VT} \text{CO} \text{(FL)} \text{SI} \text{(A)} \n\text{VZ} = \text{VT} \text{SI} \text{(FL)} \n\text{IF (NSKIP = EQ. 0) GO TO 318} \n\text{R} = \text{RF} + \text{ALT} \n\text{LBAR} = \text{LAMDA} + \text{OMEGA} \text{(TMEF - T0)} \n\text{Z} = \text{RSPHI} \n\text{Y} = \text{RCPHI} \text{SI} \text{(LBAR)} \n\text{X} = \text{RCPHI} \text{CO} \text{(LBAR)} \n\text{CALL MATROC} \n\text{NLTN} = -1 \n\text{GO TO 319} \n\text{318 CONTINUE} \n\text{CAZ} = \text{CO} \text{(AZ)} \n\text{SAZ} = \text{SI} \text{(AZ)} \n\text{VXL} = \text{VX} \text{CAZ} + \text{VY} \text{SAZ} \n\text{VZL} = \text{VZ} \n\text{XL} = \text{LENGTH} \text{CFL} \n\text{ZL} = \text{ALT} \n\text{THETA} = \text{FL} \n\text{CALL ATMSPH} \n\text{SOLVE ANGLE OF ATTACK EQUATIONS}
CALL ALFA
CALL MATCH
NLCH = X*L*CZ
FLCH = X*L*SAZ
ZLCH = ZL
ZZ = ZLCH - RF
CALL TRANS2(NLCH, FLCH, ZZ, UX, UY, UZ)
X = UX
Y = UY
Z = UZ
R = SORT(X**2 + Y**2 + Z**2)
ALT = R - RF
VX = VX*L*CZ
VY = VX*L*SAZ
VZ = VZ*L
310 CONTINUE
C
C COMPUTE INITIAL VELOCITY
CALL TRANS2(VX, VY, VZ, UX, UY, UZ)
XDOT = -OMEGA*Y + UX
YDOT = OMEGA*X + UY
ZDOT = UZ
C
C INITIALIZE S ARRAY
N = 1
S(1, N) = X
S(2, N) = Y
S(3, N) = Z
S(4, N) = XDOT
S(5, N) = YDOT
S(6, N) = ZDOT
C
C COMPUTE ACCELERATION
NK = 1
CALL ACCL9
THETAM = FL
FPLPFEV = FL
IF(VT .LT. 1.0E-5) GO TO 320
C
C COMPUTE OUTPUT DATA
CALL DATA1
NN = 2
C
C STORE DATA TEMPORARILY
CALL SETSA

IF(NSKIP .EQ. 0) GO TO 320

TIME = TIMF

TFRAC = 0.0

NTIPS = NTIP

NTIP = 0

CALL OUT

NTIP = NTIPS

TIM1 = AINT(TIMF/STPT1)*STPT1 + STPT1

TIM2 = AINT(TIMF/PTI)*PTI + PTI

NLIN = NCYC - INT(10.0*(TIM1 - TIM2 + 1.0E-51) - 1

320 CONTINUE

PTIME = AINT(TIMF/PTI)*PTI + PTI

TPREV = TIMF

N = 1

ST(1:N) = X

ST(2:N) = Y

ST(3:N) = Z

ST(4:N) = XDOT

ST(5:N) = YDOT

ST(6:N) = ZDOT

ST(7:N) = XDOTT

ST(8:N) = YDOTT

ST(9:N) = ZDOTT

IF(VTV .GT. 1.0E-3) GO TO 350

DT = LOW

CALL RUNGE(ACC, 3)

DT = DT0

GO TO 331

C

FIND PRESENT ROCKET MOTOR WEIGHT

340 CONTINUE

IF(ABS(1.0 - TSEP(NMOT)/TIME) .LT. 1.0E-6) GO TO 331

GO TO 332

331 WRM = WRM - WTM(NMOT)

SPT = LSPT + 1

DO 33 J = 1, 9

SPT(J) = ST(J, 1)

330 CONTINUE

SPT(11, LSPT) = EL

SPT(12, LSPT) = TIME

NMOT = NMOT + 1

332 CONTINUE
C READ IN DATA FOR NEXT PHASE.
CALL TABLE
KPH = KPH + KDFL
NPH = NPH + NDFL
350 CONTINUE
C CALCULATE NEXT TRAJECTORY POINT AT t + DT.
CALL RUNGF(ACCF1)
351 CONTINUE
TPDT = TIME + DT
IF((TPDT-PTIME) .GT. LOW .AND. TIME .LT. PTIME) DT =
1 PTIME - TIME + LOW
C COMPUTE OUTPUT DATA
CALL DATA1
NN = 1
C STORE DATA TEMPORARILY
CALL SETSA
IF(NSTOP .EQ. 0) GO TO 708
C CHECK STOP TIME
IF(TIME .GT. TSTOP) GO TO 703
IF(ABS(1.0 - TIME/TSTOP) .LT. 1.0E-6) GO TO 701
GO TO 70A
701 TEND = 1
TFRAC = 1.0
PTIME = TIME
GO TO 801
703 TEND = 1
TFRAC = (TSTOP - TPRF)/(TIME - TPRF)
NIP = 0
PTIME = TSTOP
PHAS = TST
GO TO 801
708 CONTINUE
C ALTIMETER CHECK.
IF(ALT .LT. 0.0 .AND. FL .LT. 0.0) GO TO 710
GO TO 720
710 TFRAC = SA(9,2,11)/(SA(9,2,1) - SA(5,1,11))
TEND = 1
PTIME = TPRF + TFRAC*(TIME - TPRF)
NIP = 0
GO TO 720
PHAS = ALTO
GO TO 801
720 CONTINUE
IF(NAP * EQ. 2) GO TO 721
C
C APOGEE CHECK
C IF(FLPREV + LT. 0.0) GO TO 721
PHAS = APO
KDF = 1
NDFL = 0
IF(KSTOP + EQ. 0) GO TO 728
IEND = 1
TFRC = 1.0
PTIME = TIME
GO TO 801
728 CONTINUE
NAP = 1
TFRC = 1.0
PTIME = TIME
NITPST = NITP
NITP = 0
CALL OUT
NITP = NITPST
TIM1 = AINT(TIME/PIT)*STPT1 + STPT1
TIM2 = AINT(TIME/PIT)*PTI + PTI
NLIN = NCYC - INT(10.*(TIM) - TIM2 + 1.0E-5))/ - 1
PTIME = AINT(TIME/PIT)*PTI + PTI
GO TO 805
721 CONTINUE
C
C IF NEXT PHAS HAS STARTED, PRINTOUT DATA
C IF(INFXT + NE. 1) GO TO 780
INFXT = 0
PTIME = TIME
CALL OUT
NITP = NITPST
TIM1 = AINT(TIME/PIT)*STPT1 + STPT1
TIM2 = AINT(TIME/PIT)*PTI + PTI
NLIN = NCYC - INT(10.*(TIM) - TIM2 + 1.0E-5))/ - 1
DT = DT0
PTIME = AINT(TIME/PIT)*PTI + PTI
GO TO 805
780 CONTINUE
AT END OF PHASE, PRINTOUT DATA

IF(INEXT .NE. 2) GO TO 790
IF(ABS(1.0 - PHT(NDH)/TIME) .GT. 1.0E-5) GO TO 790
TFRAC = 1.0
PTIME = TIME
NIIPST = NIIP
NIIP = 0
CALL OUT
INEXT = 4
PTIME = AJINT(TIME/PTI)*PTI + PTI
GO TO 805
790 CONTINUE

CHECK TIME FOR PRINTOUT
IF(END .EQ. 11) GO TO 801
IF(TIME .LT. PTMF) GO TO 805
IF(TIME .EQ. PTIME) GO TO 800

COMPUTE INTERPOLATION FRACTION
TFRAC = (PTIME - TPREV)/(TIME - TPREV)
GO TO 801
800 TFRAC = 1.0
PTIME = TIME
801 CONTINUE

CALL PRINTOUT ROUTINE
CALL OUT
IF(END .EQ. 11) GO TO 840

SET NEW PRINT TIME
PTIME = PTIME + PTI
805 CONTINUE

STORE PRESENT VALUES OF ALL OUTPUT DATA
NN = 2
CALL SFSA
TPREV = TIME
ELPREV = FL
806 CONTINUE

PHASE TIME CHECK
IF(NAP .EQ. 1) GO TO 810
GO TO 811
810 NAP = 2
```
DT = 10.0*LOW
GO TO (340, 332), KCON
A11 CONTINUE
IF(NPH .GT. NTAB) GO TO 350
IF(ABS(1.0 - PHT(NPH)/TIMF) .GT. 1.0E-5) GO TO 725
KDFL = 1
NDFL = 1
INFXT = 1
GO TO 340
A725 CONTINUE
IF((TIME + DT) .GE. PHT(NPH)) GO TO 730
IF(ABS(1.0 - PHT(NPH)/(TIMF + DT)) .LT. 1.0E-5) GO TO 730
GO TO 350
A730 CONTINUE
IF(PHT(NPH) .LE. PTIME) GO TO 713
IF((PHT(NPH) - PTIME) .LT. 4.0*LOW) GO TO 713
DT = PTIME - TIME + LOW
GO TO 350
A713 CONTINUE
DT = PHT(NPH) - TIME
INFXT = 2
GO TO 350
C
C SPENT STAGF ROUTINE
A40 NPHS = 0
NPH = 1
KCON = 2
IF(NSPENT) 999, 999, A50
A50 KSPFNT = KSPFNT + 1
KPH = 1
KDFL = 1
KPAG = 0
DO 65 J = 1, 14
PHT(J) = 900000.
A65 CONTINUE
DO A55 J = 1, 9
IF(NPAGE(J) .EQ. 5) GO TO A56
A54 CONTINUE
GO TO 85R
A56 DO A57 I = J, 9
NPAGE(1) = NPAGF(141)
A57 CONTINUE
```
R58 CONTINUE
    DO R51 J = 1,9
      ST(J,1) = SPS(J,KSPFNT)
      S(J,1) = ST(J,1)
      R51 CONTINUE
      TIME = SPS(10,KSPFNT)
      FLPREV = SPS(11,KSPFNT)
      SA(5*2,1) = 0.0
      NTP = 0
      TEND = 0
      KSTOP = 0
      NSTOP = 0
      NAP =
      THCON = 1
      DT = 10.0*LOW
      TPREV = TIME
      NSPENT = NSPENT - 1
      READ(*,NAM5)
      WTM(KSPFENT) = WTS
      ARG = TIME/DT
      PTIME = AINT(ARG)*PTI + PT
    GO TO 332
    STOP
    END
GAMA = ATAN2(VZL, VXN)
ALPH = THETA - GAMA
CS = COS(THETA)
SN = SIN(THETA)
FNORM = CNA*SAPF*A0*ALPH
AXL = (THRUST*CS - DRAG*CS - FNORM*SN)/MASS
AZL = (THRUST*SN - DRAG*SN + FNORM*CS)/MASS - GO
THDD = (-LS**FNORM1/INFRT
S(17,N) = AXL
S(18,N) = AZL
S(19,N) = THDD
RETURN
END
DOUBL PECEISION TFMP, DENS, VISCI, SOUND
COMMON/STOR/ALT, TFM, DENS, VISC, SOUND
COMMON/STOR2/X, Y, Z, R
COMMON/STOR3/THAT, SMAT, SPT, CPHT, CL, SL
COMMON/STOR4/SAPFA, MCH, CDD, CD, MACH, NC, D, CDS, DRAG
COMMON/STOR5/THT, NHT, WPR, WEIGHT, MASS
COMMON/STOR6/PO, WPL, WPP, WMT
COMMON/STOR7/DXDT, DYDT, DZDT, XDDOT, YDDOT, ZDDOT
COMMON/STOR8/OMEGA, RF, RA, RB, Go, GM, ALIM
COMMON/STOR9/S, NK
COMMON/STOR10/TCON, TI, ALTC, THX, THY, THZ
COMMON/STOR11/THX, THY, THZ, LAT, LONG, ACC, GDLAT
COMMON/STOR12/T1, T2, TO
COMMON/STOR13/APEA, IORD, TIM, NTH, THS, THRUST
COMMON/STOR14/KCON, KSPFNT
DIMENSION TIM(50), THS(50), TWI(50), WPR(50), MCH(50), CDD(50), WMT(50)
DIMENSION SMAT(3,3), TAT(3,3), S(9,6)
INTEGER TCON
REAL LAMDA, MASM, LOW, LONG, LAT, MACH, MCH
N = NK
C INPUT PRESENT POSITION AND VELOCITY
X = S(1:N)
Y = S(2:N)
Z = S(3:N)
DXDT = S(4:N)
DYDT = S(5:N)
CXDT = DXDT + OMEGAXY
CYDT = DYDT - OMEGAXX
DZDT = S(6:N)
CALL MATROC
CALL TRANS1(CXDT, CYDT, DZDT, WX, WY, WZ)  ! TRANSITION TO ACCELERATION
VX = WX
VY = WY
VZ = WZ
VT = SORT(VX**2 + VY**2 + VZ**2)
R = SORT(X**2 + Y**2 + Z**2)
ALT = R - RF
IF(ALT < RF, ALIM) GO TO 80
C CALL ATMSPH TO FIND SPEED OF SOUND, DENSITY, AND PRESSURE
CALL ATMSPH
MACH = VT/SOUND
GO TO 90
80 CONTINUE
MACH = -2.0
PRFS = 0.0
DFNS = 0.0
GO CONTINUE
Q = (DFNS*VT**2)/2.
C COMPUTE DRAG
C COMPUTE THRUST
C COMPUTE TOTAL WEIGHT
GO TO (95,98),KCON
95 CALL TAB1
CALL TAB2
CALL TAB3
GO TO 100
98 CALL TAB1
THRUST = 0.0
WEIGHT = WT*(KSP/NT)
100 CONTINUE
DRAG = CDS*Q
MASS = WEIGHT/GO
C COMPUTE GRAVITATIONAL ACCELERATION
GR = GM/R**3
GX = -GG*X
GY = -GG*Y
GZ = -GG*Z
FX = DRAG*VX/VT
FY = DRAG*VY/VT
FZ = DRAG*VZ/VT
CALL TRANS2(FX, FY, FZ, UX, UY, UZ)
GO TO 100
CALL DIRECT
THRUST CONTROL
C COMPUTE TOTAL INERTIAL ACCELERATION COMPONENTS
XDDOT = GX + (THX - UX)/MASS
YDDOT = GY + (THY - UY)/MASS
ZDDOT = GZ + (THZ - UZ)/MASS
S17,N = XDDOT
S18,N = YDDOT
S19,N = ZDDOT
RETURN
END
CACLE2      SUB. ACCEL2      --- LAUNCH RAIL ACCELERATION
SUBROUTINE ACCEL2
DOUBLE PRECISION ST
COMMON/STOR1/ALT, TEMP, PPS, DFNS, VISC, SOUND
COMMON/STOR4/SAREA, NCH, CDP, NCD, MACH, NCD, CDNS, DRAG
COMMON/STOR5/TWT, NWT, MPR, WEIGHT, MASS
COMMON/STOR8/OMEGA, RF, PA, RB, GO, GM, ALIM
COMMON/STOR9/S, NK
COMMON/STOR10/ST
COMMON/STOR15/GO, VX, VY, VZ, VT, EL, AZ, RANGE, LAT, LONG, ACC, GDLAT
COMMON/STOR17/LENGTH, SFL, DIST, LVEL, LACC
COMMON/STOR19/TIME, T0
COMMON/STOR20/APEA, IOPT, T1M, NTH, THS, THRUST
DIMENSION ST(10, 10)
DIMENSION MCH(50), CDN(50), TWT(50), MPR(50), TIM(50), THS(50), S(9,6)
REAL LVEL, LACC, MACH, MASS
N = NK
DIST = S(1:N)
LVEL = S(4:N)
MACH = LVEL/SOUND
Q = (DFNS*LVEL**2)**2,
C COMPUTE DRAG
CALL TAB1
DRAG = CNS*Q
C COMPUTE THRUST
CALL TAR2
C COMPUTE WEIGHT
CALL TAR3
MASS = WEIGHT/GO
LACC = (THRUST - WEIGHT*SFL - DRAG)/MASS
S(7, N) = LACC
RETURN
END
CACC3
SUR ACCEL3 --- ACCELERATION USING EL,AZ

SUBROUTINE ACCEL3

DOUBLE PRECISION TMAT,SMAT
COMMON/STOR1/ALT,TEMP,PRES,DENS,DISP,NO,VO,HEAT,SOUND
COMMON/STOR2/X,Y,Z,P
COMMON/STOR3/TMAT,SMAT,SPHI,PHI,CL,SL
COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NO,CDS,DRA
COMMON/STOR5/TWT,NWT,WP,WGT,WEI5,MASS
COMMON/STOR7/XDOT,YDOT,ZDOT,XXDOT,YYDOT,ZZDOT
COMMON/STOR8/OMEGA,RF,RA,RR,GO,GM,ALIM
COMMON/STOR9/S,NK
COMMON/STOR15/O,XV,YV,ZV,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
COMMON/STOR20/AREA,IO,P,TM,TN,THS,THS,THRUST
DIMENSION SMAT(3,3),TMAT(3,3),S(9,6)
DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50)
REAL MASS,LONG,LAT,MACH,MCH
N = NK

C INPUT_PRESENT POSITION AND VELOCITY
X = S(1,N)
Y = S(2,N)
Z = S(3,N)
DXDT = S(4,N)
DYDT = S(5,N)
CXDT = DXDT + OMEGA*Y
CYDT = DYDT + OMEGA*X
DZDT = S(6,N)

CALL MATROC
CALL TRANS1(CXDT,CYDT,DZDT,WX,XY,WZ)

VX = WX
VY = XY
VZ = WZ

VT = SQRT(VX**2 + VY**2 + VZ**2)
CFLCAZ = COS(EL)*COS(AZ)
CELSAZ = COS(EL)*SIN(AZ)
SEL = SIN(EL)

IF(VT .LT. 1.0E-5) GO TO 70
CFLCAZ = VX/VT
CELSAZ = VY/VT

SFL = VZ/VT
GO TO 70

70 CONTINUE
R = SQRT(X**2 + Y**2 + Z**2)
ALT = R - RF
IF (ALT .GE. ALIM) GO TO 80

C CALL ATMSPH TO FIND SPEED OF SOUND, DENSITY, AND PRESSURE
CALL ATMSPH
MACH = VT/SOUND
GO TO 90

80 CONTINUE
MACH = -2.0
PRES = 0.0
DENS = 0.0

90 CONTINUE
Q = (DENS*VT**2)/2.
CALL TAB1
DRAG = COS*Q

C COMPUTE THRUST
CALL TARP

C COMPUTE ROCKET WEIGHT
CALL TAB3
MASS = WEIGHT/GO
DIFF = INRUST - DRAG

C COMPUTE GRAVITATION ACCELERATION
GG = GM/R**3
GX = -GG*X
GY = -GG*Y
GZ = -GG*Z

C COMPUTE AERO FORCES IN TOPOCENTRIC SYSTEM
FX = DIFF*CFLCAZ
FY = DIFF*CELSAZ
FZ = -DIFF*SEL

C TRANSFORM AERO FORCES TO INERTIAL SYSTEM
CALL TRANS2 (FX, FY, FZ, UX, UY, UZ)

C COMPUTE INERTIAL ACCELERATION
XDDOT = GX + UX/MASS
YDDOT = GY + UY/MASS
ZDDOT = GZ + UZ/MASS
S(7,N) = XDDOT
S(8,N) = YDDOT
S(9,N) = ZDDOT
RETURN
END
CALFA  SURF, ALFA ---- ANGLE OF ATTACK EQUATIONS
SUBROUTINE ALFA

DOUBLE PRECISION,ST
DOUBLE PRECISION DP1
COMMON/STOR1/ALT, TEMP, PRES, DENS, VISC, SOUND
COMMON/STOR2/X, Y, Z, R
COMMON/STOR4/SAPEA, MCH, CDD, CD, MACH, NCD, CDS, DRAG
COMMON/STOR5/TWT, NWT, WPR, WEIGHT, MASS
COMMON/STOR7/XDOT, YDOT, ZDOT, XDDOT, YDDOT, ZDDOT
COMMON/STOR8/OMEGA, RF, RA, BR, RO, GM, ALIM
COMMON/STOR9/S, NK
COMMON/STOR10/ST
COMMON/STOR13/XL, ZL, THETA, VX, VL, ALPH, GAMA, CAZ, SAZ
COMMON/STOR14/LSM, INERT, CNA
COMMON/STOR15/G, VX, VY, VZ, VT, EL, AZ, RANGE, LAT, LONG, ACC, GDLAT
COMMON/STOR16/SA, ARRAY, NN
COMMON/STOR18/TFRAC, LINE, PTIME, CONV, IEND, IPRINT, NPAGE
COMMON/STOR19/TIMF, TO
COMMON/STOR20/AREA, IOPT, TIM, NTH, THS, THRUST
COMMON/STOR23/SPH10, CPH10, LAMDA0, X0, Y0, Z0
COMMON/STOR24/PH, NTAB, NPH, KPH
COMMON/STOR25/LOW, FPT1NY, EPRIG, PTI, PI, P12, DT, HIGH
DIMENSION TIM(50), THS(50), S(9, 6), TWT(50), WPR(50)
DIMENSION NPAGF(10)
DIMENSION PH(14), NJ(14), MCH(50), CDD(50)
DIMENSION ST(20, 10), APRAY(10, 30, 10), SA(10, 2, 10)
REAL LAT, LONG, LAMDA0, LOW
REAL MCH, MACH
REAL LSM, INERT, MASS
EXTERNAL ACCEL0
PTIME = AINT(TIME/PTI)*PTI + PTI
ILIM = PHT(2) - 1, 0
TPREV = TIMF
CALL ATMSPH
MACH = VT/SOIND
Q = (DENS*VT**2)/2.
CALL TAB1
DRAG = CDD*Q
CALL TAB2
CALL TAB3
MASS = WIGHT/GO
C INITIALIZE LAUNCH VARIABLES
AXL = (THRUSt - DRAg) * COS(THETA) / Mass
AZL = (THRUSt - DRAg) * SIN(THETA) / Mass - GO
THDD = 0.0
IF(VT GT 1.0E-5) GO TO 80
DP1 = VXI
VXI = DP1 + (1.0D-4)*AXL
DP1 = VZI
VZI = DP1 + (1.0D-6)*AZL
TIMF = TIME + 1.0E-6
80 CONTINUE
VL = SQRT(VXl**2 + VZI**2)
C INITIALIZE S ARRAY
N = 1
S(1,N) = XL
S(7,N) = ZL
S(3,N) = THETA
S(4,N) = VXI
S(5,N) = VZI
S(6,N) = 0.0
S(7,N) = AXL
S(8,N) = AZL
S(9,N) = THDD
DO 90 J = 1,N
S(J,1) = S(J,11)
90 CONTINUE
100 CONTINUE
CALL RUNGEKACEL(0)
TPDT = TIME + DT
IF ( (TPDT - PTIME) GT LOW AND TIMF LT PTIME ) DT = 1
PTIME = TIME + LOW
AXL = S(7+1)
AZL = S(8+1)
VT = SQRT(VXl**2 + VZI**2)
CALL MATLCH
XT = XL*CAz
YT = XL*SAz
ZT = -ZL
ZTZ = ZT - RE
CALL TRANS2(XT, YT, ZT, UX, UY, UZ1)
X = UX
Y = UY
Z = UZ
R = SQRT(X**2 + Y**2 + Z**2)
ALT = R - RE
VX = VX*CAZ
VY = VX*SAZ
VZ = - V2L
CALL TRANS2(VX, VY, VZ, UX, UY, UZ)
XDOT = -OMEGA*Y + UX
YDOT = OMEGA*X + UY
ZDOT = UZ
ACX = AXL*CAZ
ACY = AXL*SAZ
ACZ = -AZL
CALL TRANS2(ACX, ACY, ACZ, UX, UY, UZ)
XDDOT = UX - X*OMEGA*YDOT + OMEGA*Z**2*Y
YDDOT = UY + 2*OMEGA*XDOT + OMEGA*Z**2*X
ZDDOT = UZ
CALL DATA1
NN = 1
CALL SSETS
IF(ALPH .GT. 0.0) GO TO 750
TFRAC = SA(1,2,1)/(SA(1,2,1) - SA(1,1,1))
TFRAC = EPS(TFRAC)
TIMEF = TREV + TFRAC*(T12F - TREV)
XL = XL2 + TFRAC*(XL - XL2)
ZL = ZL2 + TFRAC*(ZL - ZL2)
VXL = VXL2 + TFRAC*(VXL - VXL2)
VZL = VZL2 + TFRAC*(VZL - VZL2)
GAMA = ATAN2(VZL, VXL)
EL = GAMA
GO TO 200
750 CONTINUE
IF(TIME .LT. PTIMF) GO TO 805
IF(TIME .EQ. PTIMF) GO TO 800
C COMPUTE INTERPOLATION FRACTION
TFRAC = (PTIMF - TREV)/(TIME - TREV)
GO TO 801
800 TFRAC = 1.0
PTIME = TIME
C 01 CONTINUE
C    CALL PRINTOUT ROUTINE
C    CALL OUT
C    SET NEW PRINT TIME
PTIME = PTIME + PTI
C 05 CONTINUE
C    STORE PRESENT VALUES OF ALL OUTPUT DATA
NN = 2
CALL SETSA
TREV = TIME
XL2 = XL
ZL2 = ZL
VXL2 = VXL
VZL2 = VZL
IF(TIME .GT. 20.0) GO TO 200
IF(TIME .LT. TLIM) GO TO 100
200 RETURN
FND
1ATMSPH  SURATMSPH  ATMOSPHERIC VARIABLES
SURROUTINE ATMSPH
COMMON/STOR1/ALT,TFMP,PRFS,DFNS,VISC,SOUND
COMMON/STOR8/AMFRA,RF,RA,RR,GO,GM,ALTM
COMMON/STOR36/PRES0,DFNS0,TEMP0,SOUND0,K1,CON1,KPRFV
COMMON/STOR37/TSLOPE,HALT,TFMPB,PRFSR
DIMENSION TSLOPE(9),HALTB(9),TEMPB(9),PRFSR(9)
REAL K1
HALT = ALT*(RF/RF+ALT1)*CON1
CALL INTPR(HALT,N,5000,5800)
DHALT = HALT - HALTB(N)
TEMP = TEMPB(N) + TSLOPE(N)*DHALT
IF(TSLOPE(N)1400,500,400)
400 CONTINUE
PRATIO = PRFSR(N)*(TFMPB(N)/TEMP)**(K1/TSLOPE(N))

GO TO 600
400 CONTINUE
PRATIO = PRFSR(N)*EXP((-K1/TEMPB(N))*DHALT)

600 CONTINUE
TRATIO = TEMPB/TEMP0
DFNS = DFNS0*PRATIO/TRATIO
SOUND = SOUND0*SORT(T=TRATIO)
PRFS = PRATIO*PRES0
RETURN
800 PRFS = 0.0
DFNS = 0.0
DENS = 0.0
SOUND = 0.0
RETURN
800 PRFS = PFS0
DFNS = DFNS0
TEMP = TEMP0
SOUND = SOUND0
RETURN
END
BLOCK DATA
COMMON/STOR36/PRES0,DENS0,TEMPO,SOUND0,K1,COM1,KPREV,
COMMON/STOR37/TSLOPE,HALT,TFMPR,PRESA,
DIMENSION TSLOPE(9),HALTB(9),TEMPB(9),PRESB(9)
REAL K1
DATA TSLOPE/ -6.5E-3,0.0,1.0E-3,2.8E-3,0.0,-2.0E-3,-4.0E-3,0.0,
1.0
DATA HALTB/ 0.0,1.0,0F+3,2.0,0E+3,3.0,0E+3,4.0,0F+3,5.0,0F+3,6.0,0F+3
1.7E+3,8.4E+3/
DATA TEMPB/ 288.15,216.65,216.65,228.65,270.65,270.65,270.65,270.65,
1.180.65,180.65/
DATA PRESB/ 1.0,2.0,22361F-1,5.0,40328F-2,2.8,56663F-3,1.09454F-3,
1.5,8289F-4,1.7971E-4,1.0241E-5,1.6223E-6/
DATA PRES0/2116.27,0FNS0/2.3769E-3,TFMPO/28R,15/,
SOUND0/1116.45,K1/34.163194E-3,CON1/3048,1,KPREV/17
END
COMPUTE ADDITIONAL OUTPUT

SUBROUTINE DATA1

DOUBLE PRECISION CCS
COMMON/STOR1/ALT, TEMP, PRES, DENS, VISC, SOUND
COMMON/STOR2/X, Y, Z, R
COMMON/STOR7/VDOT, YDOT, ZDOT, XDOT, ZDOT, ZDOT
COMMON/STOR8/OMEGA, OF, PAV, RR, GO, EM, A1M
COMMON/STOR12/YL, YL + X, YX, GAM, PLOT
COMMON/STOR13/XL, ZL, THETAT, VX, VL, VXL, ALPH, GAM, CAZ, SAZ
COMMON/STOR15/OL, XX, Y, Z, V, T, EL, AZ, RANGE, LAT, LONG, ACC, GDLAT
COMMON/STOR19/TIME, TO
COMMON/STOR25/LOW, EPT, IN, EBP, IG, TP, PI, PI2, DT, HIGH
COMMON/STOR27/IPT, ITP, ITP, ITP1, IPT1, IPT2, IPT3, IPT4
COMMON/STOR28/RZ, AZ, ELCH, FZLCH, FZLCH, FZLCH, NLCH, NLCH, NLCH, NLCH
1 NDLCH, ZNLCH
COMMON/STOR29/RPR, AP, ZP, APL, VELR, VELR, FRAD, FRAD, FRAD, FRAD, SRAD
1 NDRAD, ZPRAD
COMMON/STOR30/DELTA, VEL, ACC1
REAL NRAD, NDLCH, NDRAD, ZNLCH
REAL LAMBDA, MAS, SX, LRL, LRL, LRL, LRL, LRL, LRL, LRL, LRL
VEL = SORT(XDOT**2 + YDOT**2 + ZDOT**2)
ACC = SORT(XDOT**2 + YDOT**2 + ZDOT**2)
FX = XDOT + 2 * OMEGA * YDOT - OMEGA**2 * X
FY = YDOT + 2 * OMEGA * XDOT - OMEGA**2 * Y
EZ = ZDOT
CALL TRANS1(EX, FY, EZ, ACX, ACY, ACZ)
ACC = SORT(ACX**2 + ACY**2 + ACZ**2)
CALL VATRAD
CALL TRANS1(X, Y, Z, WX, WY, WZ)
ZRAD = -WX - RE
FRAD = WY
NPRAD = WX
RRAD = SORT(FRAD**2 + NRPAD + ZRAD**2)
CXDT = XDOT + OMEGA * Y
CYDT = YDOT - OMEGA * X
CZDT = ZDOT
CALL TRANS1(CXDT, CYDT, CZDT, WX, WY, WZ)
EDRAD = WY
NDRAD = WX
ZDRAD = -WX
VELRAD_ = SORT((EDRAD**2 + NDRAD**2 + ZDRAD**2))
SRT_ = SORT((FRAD**2 + NFRAD**2))
ELRAD_ = ATAN2(ZPAD,SRT)
AZRAD_ = ATAN2(EPRAD,NRAD)

IF(AZRAD_ .LT. 0.0) AZRAD_ = AZRAD_ + PI2
CALL MATLCH
CALL TRANS1(X,Y,Z,WX,wy,WZ)
ELCH_ = wy
NLCH_ = wx
ZLCH_ = -wz - _RE
RLCH_ = SORT((ELCH**2 + _NLCH**2 + ZLCH**2))
CALL TRANS1(CXT,CYDT,CZDT,WX,wy,WZ)
EDLCH_ = wy
ZDLCH_ = -wz
NDLCH_ = wx
VELLCH_ = SORT((EDLCH**2 + _NDLCH**2 + ZDLCH**2))
SRT_ = SORT((ELCH**2 + _NLCH**2))
ELLCH_ = ATAN2(ZLCH,SRT)
AZLCH_ = ATAN2(ELCH,NLCH)

IF(AZLCH_ .LT. 0.0) AZLCH_ = AZLCH_ + PI2
XL_ = ELCH*CAZ + NLCH*CAZ
YL_ = ELCH*CAZ + NLCH*SAZ
ZL_ = ZLCH
VXL_ = EDLCH*SAZ + _NDLCH*CAZ
VYL_ = EDLCH*CAZ + _NDLCH*SAZ
VZL_ = ZDLCH

RXY_ = SORT((XL**2 + YL**2))
SRT_ = SORT((VXL**2 + VYL**2))
GAML_ = ATAN2(VZL,SRT)

RLDQ1_ = (X*X + Y*Y + Z*Z)/RLCH
RANGE_ = 0.0

IF(RLCH_ .LT. 5.0) GO TO 700
CCS = (R*R + P**2 - RLC**2)/(2.*R*RE)
IF(DABS1(CCS) .GT. 1.0) CCS = 1.0
DELT_ = DATAN2(DSORT(11.-CCS**2),CCS)
RANGE_ = RE*DELT

700 CONTINUE
SRT_ = SORT(Y**2 + X**2)
PHI_ = ATAN(Z/SRT)
ARG_ = (RA/RR)**2*(Z/SRT)
GDLAT = ATAN(ARG)
LAT = PHI
LAMDA = ATAN2(Y,X)
LONG = LAMDA - OMEGA*(TIME - TO)
IF(LONG .GT. PI) LONG = LONG - PI2
IF(LONG .LT. (-PI1)) LONG = LONG + PI2
EL = ATAN(-VZ/SQRT(VX**2 + VY**2))
ALPH = THETA - EL
AZ = ATAN2(VY,VX)
IF(AZ .LT. 0.) AZ = AZ + PI2
IF(ALT .LT. 5.0 AND EL .LT. 0.01) RETURN
CALL LIP
RETURN
END
SUBROUTINE DIRECT

COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
COMMON/STOR11/THCON,T1,ALTG,THX,THY,THZ
COMMON/STOR13/XL,ZL,THETA,VXL,VZL,ALPH,GAMA,CAZ,SAZ
COMMON/STOR15/O,VX,VY,VZ,VT,EL,AZ,RANGE,LT,LONG,ACC,GDLAT
COMMON/STOP19/TIME,TO
COMMON/STOR20/AFA,1OPT,TIM,NTH,THS,THRUST
DIMENSION TIM(50),THS(50)

INTEGER THCON
EL = ATAN(-VZ/SGRT(VX**2 + VY**2))
XFRAC = VX/VT
YFRAC = VY/VT
ZFRAC = VZ/VT
CALL_TRANS2(XERAC,YERAC,ZERAC,UX,UY,UZ)
XFRAC = UX
YFRAC = UY
ZFRAC = UZ
THETA = EL

IF(TIME .LT. T1) OR. NSET .EQ. J OR. ALT .LT. ALTG) GO TO 100
XFRAC1 = XFRAC
YFRAC1 = YFRAC
ZFRAC1 = ZFRAC
ELST = EL
TIMST = TIME
NSET = J

100 CONTINUE
IF(TIME .LT. T1) OR. ALT .LT. ALTG) GO TO 201
GO TO (201,202), THCON

201 CONTINUE
C NO THRUST CONTROL
THX = THRUST*XFRAC
THY = THRUST*YFRAC
THZ = THRUST*ZFRAC
GO TO 900

202 CONTINUE
C THRUST CONTROL OPTION
THX = THRUST*XFRAC
THY = THRUST*YFRAC
THZ = THRUST*ZFRAC
CALL TRANS1(XFRAC1,YFRAC1,ZFRAC1,WX,WY,WZ)
THETA = ATAN(-WZ/SGRT(WX**2 + WY**2))

000 RETURN
END
C1IP

**SUBROUTINE IIP**

**DOUBLE PRECISION** SIG2, SIG, SQ
**DOUBLE PRECISION** FCF, ESF, FCFIP, FSEIP, SDEL, CDLF, DEL, F, G, A
**DOUBLE PRECISION** XIP, YIP, ZIP, LC, ESQ, X0, Y0, Z0, X, Y, Z, R

**COMMON/STOR2*/XX, YY, ZZ, RR
**COMMON/STOR7*/XDOT, YDOT, ZDOT, XDDOT, YDDOT, ZDDOT
**COMMON/STOR8*/OMEGA, EF, PA, PR, GM, GM, ALIM
**COMMON/STOR13*/XL, ZL, THETA, VX, VY, ALPH, GAMA, CAZ, SAZ
**COMMON/STOR19*/TIME, TO
**COMMON/STOR23*/SPHIO, CDPHO, LAMDA0, XX0, YY0, ZZ0
**COMMON/STOR25*/LCW, FPTINY, EPSIG, PI1, PI2, DT, HIGH
**COMMON/STOR27*/IIPLAT, IIPLO, IIPP, IIPTM

**REAL** LAMDA, MASS, K, LOW, LONG, LAT, MACH, MCH, LAMDA0, LBAR

**REAL** IIPLAT, IIPLO, IIPP, IIPTM

PUBLIC CAN

C

**INITIALIZE** X, Y, Z, XDOT, YDOT, ZDOT, P, VEL

X = XX
Y = YY
Z = ZZ
R = RR

VSO = XDOT**2 + YDOT**2 + ZDOT**2
SFGM = SOR(GM)
ECF = R*VSO/GM - 1.0
A = R/(1.0 - ECF)

ESF = (X*XDOT + Y*YDOT + Z*ZDOT)/(SFGM**DSORT(A))

FSO = FCF**2 + ESF**2
FCFIP = 1.000 - PF/A
SO = FSO - FCFIP**2
IF(SO < 0.000) GO TO 100
SO = 0.000

100 ESEIP = -DSORT(SO)

SDEL = (FSFIP*FCE - FCFIP*ESF)/ESO
CDLF = (FSEIP*FCE + FSEIP*ESF)/ESO
DEL = DANA(2, SDEL, CDLF)

IF(DEL > 0.000) DEL = DEL + PI2
F = (CDLF - FCF)/(1.0 - ECF)
G = (A**1.5/SFGM)**(SDEL + ESF - FSEIP)
TFL = (A**1.5/SFGM)**(DEL + ESF - FSEIP)
CAN = TFL + (TIME - TO)
XIP = FX + G*XDOT
YIP = FY + G*YDOT
ZIP = FZ + G*ZDOT
ZIP = F*Z + G*ZDOT
LBAR = LAMDAO + OMEGA*CAPT
XO = RE*PHIO*COS(LBAR)
YO = RE*PHIO*SIN(LBAR)
Z0 = RE*SPHIO
LC = DSORT((XIP - XO)**2 + (YIP - YO)**2 + (ZIP - Z0)**2)
LAMDA = ATAN2(YIP, XIP)
LONG = LAMDA - OMEGA*CAPT
PHI = ATAN(ZIP/SORT(XIP**2 + YIP**2))
G0PHI = ATAN((RA/RB)**2*SIN(PHI)/COS(PHI))
SSIG2 = LC/(2.*RE)
IF((DABS(SSIG2) .GT. 1.000), SSIG2 = 1.000)
CSIG2 = DSORT(1.0 - SSIG2**2)
SIG = 2.*DATAN(SSIG2/CSIG2)
RANGE = RE*SIG
IIPLAT = G0PHI
IIPLON = LONG
IIPR = RANGE
IIPRIM = CAPT
RETURN
END
SUBROUTINE INPUT

DIMENSION AA(14)
DATA EN/6HZ99999/
REWIND 5
23 CONTINUE
READ(5,19) AA
19 FORMAT(1446)
PRINT 19, AA
 IF(AA(2),EQ.,EN)_GO TO 34.
 GO TO 23
34 REWIND 5
RETURN
END
CINTERP

SUBROUTINE INTERP(QC,KP,**

COMMON/STOR36/PFESO,DENSO,TEMP0,SOUND0,K1,CON1,KPFO

COMMON/STOR37/TSLOPE,HALTMB,TEMPB,PFESR

DIMENSION TSLOPE(9),HALTBL(9),TEMPBL(9),PFESR(9)

KP = KPFO

IF(QQ .LE. HALTB(KP+1) .AND. QQ .GT. HALTR(KP)) GO TO 200

IF(QQ .LE. HALTB(KP)) RETURN1

IF(QQ .GT. HALTB(KP)) RETURN2

IF(QQ .EQ. HPRFV) 150,200,100

100 DO 11 J=KP,8

IF(QQ .GT. HALTB(J) .AND. QQ .LE. HALTR(KP+1)) GO TO 120

110 CONTINUE

120 KP = J

GO TO 200

150 KPP = KP - 1

DO 16 J=1,KPP

160 CONTINUE

180 KP = KP - J

200 CONTINUE

KPRFV = KP

HPRFV = QQ

RETURN

END
CLaunch  SUB , LAUNCH  ----  LAUNCH RAtl

SUBROUTINE LAUNCH

DOUBLE PRECISION ST

COMMON/STOR1/ALT,TFMP,PRE,S,N,FNS,ENC,OSC,SOUND
COMMON/STOR4/SARFA,MCH,CD,D,C,MACH,MC,CDS,DRAG
COMMON/STOR5/TWT,MWT,WRP,WEIGHT,MASS
COMMON/STOR8/OMEGA,PE,PA,RB,GO,CM,ALT
COMMON/STOR9/S,NK
COMMON/STOR10/ST
COMMON/STOR15/O,VX,VY,VZ,ET,EL,AL,AR,AN,ACC,DDLAT
COMMON/STOR17/LENGTH,SEL,DIST,LEVEL,LeCC
COMMON/STOR19/TIME,TO
COMMON/STOR20/AREA,IC,IT,L,D,THRST
COMMON/STOR25/LAIN,EPTINY,EPBIG,PTI,PI,P12,DT,HIGH

DIMENSION ST(70,10)
DIMENSION MCH(50),CD(50),TWT(50),WRP(50),TIM(50),THS(50),S(9,6)
RFAL LEVEL,LCACC,LEVEL,LEVEL2,LACC2,MASS

EXTERNAL ACCL

IF(LENGTH LT 0.1) RETURN

CALL ATMSPH

TIME = 0.

DT = AMIN1(0.01,DT)

CALL TAB2

CALL TAB3

MASS = WEIGHT/GO

LACC = (THRST - WEIght*SL)/MASS

N=1

S(1,N) = 0.
S(4,N) = VT
S(7,N) = LACC
ST(1,1) = S(1,1)
ST(4,1) = S(4,1)
ST(7,1) = S(7,1)

100 CONTINUE

DIST = DIST

ALT = ALT

LEVEL2 = LVEL

TIME2 = TIME

CALL RUNGF(ACCFL2)

DT = AMIN1(0.01,DT)

ALT = DIST*SL

IF(DIST LT LENGTH) GO TO 100

TFRAC = (LENGTH - DIST)/(DIST - DIST)

ALT = ALT2 + TFRAC*(ALT - ALT2)

LEVEL = LEVEL2 + TFRAC*(LEVEL - LEVEL2)

TIME = TIME2 + TFRAC*(TIME - TIME2)

VT = LEVEL

100 RETURN

END
**CMAT**

**SUR. MAT --- ROTATION MATRIX AND INVERSE**

**SUBROUTINE MAT**

**DOUBLE PRECISION TMA T, SMAT**

**DIMENSION TMA T(3,3), SMAT(3,3)**

**COMMON STOR3, TMA T, SMAT, SPHI, CPHI, CL, SL**

**TMAT(1,1) = -SPHI*CL**

**TMAT(1,2) = -SPHI*SL**

**TMAT(1,3) = CPHI**

**TMAT(2,1) = -SL**

**TMAT(2,2) = CL**

**TMAT(2,3) = 0.**

**TMAT(3,1) = -CPHI*CL**

**TMAT(3,2) = -CPHI*SL**

**TMAT(3,3) = -SPHI**

**SMAT(1,1) = TMAT(1,1)**

**SMAT(1,2) = TMAT(2,1)**

**SMAT(1,3) = TMAT(3,1)**

**SMAT(2,1) = TMAT(1,2)**

**SMAT(2,2) = TMAT(2,2)**

**SMAT(2,3) = TMAT(3,2)**

**SMAT(3,1) = TMAT(1,3)**

**SMAT(3,2) = TMAT(2,3)**

**SMAT(3,3) = TMAT(3,3)**

**RETURN**

**END**
CMAILCH     SUB. MATLCH.----- LAUNCH_SITE_MATRIX

SUBROUTINE MAILCH

DOUBLE PRECISION TMAT, SMAT

COMMON/STOR3/TMAT, SMAT, SPHI, CPHI, CL, SL

COMMON/STOR8/OMEGA, RE, PA, RB, GO, GM, ALIM

COMMON/STOR19/TIME, TO

COMMON/STOR23/SPHIO, CPHIO, LAMDAO, XO, YO, Z0

DIMENSION TMAT(3, 3), SMAT(3, 3)

REAL LBAR, LAMDAO

SPHI = SPHIO

CPHI = CPHIO

LBAR = LAMDAO + OMEGA*(TIME - TO)

SL = SIN(LBAR)

CL = COS(LBAR)

CALL MAT

RETURN

END
C MATRAD
SUR MATRAD -- RADAR SITE MATRIX

SUBROUTINE MATRAD

DOUBLE PRECISION THAT, SMAT

COMMON/STOR3/TMAT, SMAT, SPHI, CPHI, CL, SL

COMMON/STOR8/OMEGA, PE, PA, RB, GO, GM, ALIM

COMMON/STOR19/TIME, TO

COMMON/STOR31/SPHI1, CPHI1, LAMDA1

DIMENSION SMAT(3,3), TVAT(3,3)

REAL LBAR, LAMDA1

SPHI = SPHI1

CPHI = CPHI1

LBAR = LAMDA1 + OMEGA*(TIME - TO)

SL = SIN(LBAR)

CL = COS(LBAR)

CALL MAIN

RETURN

END
CMATROC
SUBROUTINE CMATROC

DOUBLE PRECISION TMAT, SMAT

COMMON/STOP2/X, Y, Z, R
COMMON/STOR3/TMAT, SMAT, SPHI, CPHI, CL, SL

DIMENSION TMAT(3,3), SMAT(3,3)

R = SQRT(X**2 + Y**2 + Z**2)
SRT = SQRT(X**2 + Y**2)

SPHI = Z/R
CPHI = SRT/R
CL = X/SRT
SL = Y/SRT

10 CONTINUE
CALL MAT
RETURN
END
106. \text{FORMAT}((12X,6H, PAGE ,J4),HF//)
DATA RLWK/087777777777/
DATA R1/K/020202020202/
DATA NVAR/8,8,10,10,4,3/
IF(NLIP \cdot EQ. \_0_) GO TO 300
L = LINE + 1
KLIN = L
NLIN = NLIN + 1
IF(NLIN \cdot EQ. \_NCYC) GO TO 300
GO TO 301
300 IF(LINE \cdot EQ. \_30) LINF=0
NLIN = 0
LINE = LINE + 1
L = LINE
KLIN = L
MESS(L) = PHAS
PHAS = BLKK
301 CONTINUE
TIME(L) = PTIME
DO 11 J = 1,6
NV = NVAR(J)
DO 10 I = 1, NV
ARRAY(I, L, J) = SA(I, 2, J) \cdot TFRAC(SA(I, 1, J) - SA(I, 2, J))
ARRAY(I, L, J) = ARPA(Y(I, L, J) \cdot CON(I, J)
IF(SA(4, 2, 2) \cdot GE. 0,0 \cdot AND. SA(4, 1, 2) \cdot GE. 0,0) GO TO 10
ARRAY(4, L, 2) = BLNK
ARRAY(4, L, 2) = BLNK
10 CONTINUE
11 CONTINUE
CONDEPT TO DEGREFS
ARRAY(1, L, 1) = ARRAY(1, L, 1) \cdot CONV
ARRAY(2, L, 1) = ARRAY(2, L, 1) \cdot CONV
ARRAY(3, L, 1) = ARRAY(3, L, 1) \cdot CONV
ARRAY(4, L, 1) = ARRAY(4, L, 1) \cdot CONV
ARRAY(7, L, 1) = ARRAY(7, L, 1) \cdot CONV
ARRAY(8, L, 1) = ARRAY(8, L, 1) \cdot CONV
ARRAY(8, L, 3) = ARRAY(8, L, 3) \cdot CONV
ARRAY(2, L, 4) = ARRAY(2, L, 4) \cdot CONV
ARRAY(3, L, 4) = ARRAY(3, L, 4) \cdot CONV
ARRAY(1, L, 5) = ARRAY(1, L, 5) \cdot CONV
VXL, VZL - are velocity components in the downrange and vertical directions. Found by integration of AXL and AZL.

CS = \cos(\theta)

\theta - is the inclination angle of the rocket, the angle between the principal body axis and the horizontal. It is originally set to the launch elevation angle, and is modified by double integration of THDD, \theta double dot.

FNORM = CNA x SAREA x Q x ALPH

CNA - is the slope of the coefficient of the normal force acting on the center of pressure.

SAREA - is the reference area.

Q - is defined above.

ALPH - \theta - \gamma, and is called the "angle of attack". The program switches from the Period I model to the model used for the rest of the flight when ALPH becomes zero (found by interpolation as it crosses from plus to minus). The duration of Period I is about 0.8 second for a NIKE-CAJUN, and about 5 seconds for a SCOUT.

\gamma = \tan^{-1}(VZL/VXL)

\gamma is the flight path angle which defines the direction in which the rocket is moving.

MASS - is the weight of the rocket divided by the force of gravity.

AZL - is acceleration in the Z direction, which is vertical at the launch site.

THDD - is \theta double dot, the acceleration in the rocket's inclination angle to the X_L axis.

LSM = LCP - LCG.

LCP - is the distance from the reference position (usually the nose of the rocket) to the center of pressure.

LCG - is the distance from the reference position to the rocket's center of gravity.

INERT - is the rocket's pitch moment of inertia.

The equations above have been simplified by the elimination of effects such as pitch damping and jet damping which were found to have no significant
```
| CALL TOUT
| IF LINE=30 OR IEND=1 OR IPRINT=1, PRINTOUT |
| IF (LINE .EQ. 30 .OR. IEND .EQ. 1 .OR. IPRINT .EQ. 1) GO TO 50 |
| GO TO 200 |
| KPAG = KPAG + 1 |
| GO TO (151, 152, 151), NSYS |
| 151 CONTINUE |
| I = 1 |
| NPAG = NPAG(I) |
| I = I + 1 |
| GO TO (61, 62, 63, 64, 65, 66, 67, 68), NPAG |
| 61 CONTINUE |
| PRINT 101, KPAG |
| PRINT 30 |
| PRINT 31 |
| PRINT PAGE A (ENGLISH) |
| PRINT 20, ((MESS(LLL), TIMA(LLL), (ARRAY(I, LLL, 1), I=1, 08)), LL=1, L) |
| PRINT 19 |
| NPAG = NPAGF(I) |
| I = I + 1 |
| GO TO (61, 62, 63, 64, 65, 66, 67, 68), NPAG |
| 62 CONTINUE |
| PRINT 102, KPAG |
| PRINT 32 |
| PRINT 33 |
| PRINT PAGE B (ENGLISH) |
| PRINT 21, ((MESS(LLL), TIMA(LLL), (ARRAY(I, LLL, 2), I=1, 08)), LL=1, L) |
```
PRINT 19
NPAG = NPAGF(1)
I = I + 1
GO TO (61, 62, 63, 64, 65, 66, 67, 68), NPAG
63 CONTINUE
PRINT 103, KPAG
PRINT 36
PRINT 24
PRINT 25
C PRINT PAGE C (ENGLISH)
PRINT 23, ((MESS(LL), TIMA(LL), (ARRAY(I, LL, 3), I=1, 10)), LL=1, L)
PRINT 19
NPAG = NPAGF(1)
I = I + 1
GO TO (61, 62, 63, 64, 65, 66, 67, 68), NPAG
64 CONTINUE
PRINT 104, KPAG
PRINT 37
PRINT 34
PRINT 35
C PRINT PAGE D (ENGLISH)
PRINT 22, ((MESS(LL), TIMA(LL), (ARRAY(I, LL, 4), I=1, 10)), LL=1, L)
PRINT 19
NPAG = NPAGF(1)
I = I + 1
GO TO (61, 62, 63, 64, 65, 66, 67, 68), NPAG
65 CONTINUE
PRINT 105, KPAG
PRINT 28
PRINT 42
PRINT 43
C PRINT PAGE E (ENGLISH)
PRINT 21, ((MESS(LL), TIMA(LL), (ARRAY(I, LL, 5), I=1, 10)), LL=1, L)
PRINT 19
NPAG = NPAGF(1)
I = I + 1
GO TO (61, 62, 63, 64, 65, 66, 67, 68), NPAG
66 CONTINUE
PRINT 106, KPAG
PRINT 38
PRINT 44
PRINT 45
C PRINT PAGE F (ENGLISH)
PRINT 27, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,6),I=1,03)),LL=1,L)
PRINT 19
67 CONTINUE
68 CONTINUE
GO TO (159,152,152),NSYS
152 CONTINUE
I = 1
NPAG = NPAGE(I)
I = I + 1
GO TO (81,82,83,84,85,86,87,88),NPAG
81 CONTINUE
PRINT 101,KPAG
PRINT 30
PRINT 51
C PRINT PAGE A (METRIC)
PRINT 20, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,1),I=1,08)),LL=1,L)
PRINT 19
NPAG = NPAGE(I)
I = I + 1
GO TO (81,82,83,84,85,86,87,88),NPAG
82 CONTINUE
PRINT 102,KPAG
PRINT 32
PRINT 53
C PRINT PAGE B (METRIC)
PRINT 21, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,2),I=1,08)),LL=1,L)
PRINT 19
NPAG = NPAGE(I)
I = I + 1
GO TO (81,82,83,84,85,86,87,88),NPAG
83 CONTINUE
PRINT 103,KPAG
PRINT 36
PRINT 24
PRINT 56
C PRINT PAGE C (METRIC)
PRINT 23, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,3),I=1,10)),LL=1,L)
PRINT 19
NPAG = NPAGE(I)
I = I + 1
GO TO (81, 82, 83, 84, 85, 86, 87, 88), NPAG
84 CONTINUE
PRINT 104, KPAG
PRINT 37
PRINT 34
PRINT 55
C PRINT PAGE D (METRIC)
PRINT 22, ((MESS(LL), TIMA(LL), (ARREY(I, LL, 4), I=1, 10)), LL=1, L)
PRINT 19
NPAG = NPAGE(I)
I = I + 1
GO TO (81, 82, 83, 84, 85, 86, 87, 88), NPAG
85 CONTINUE
PRINT 105, KPAG
PRINT 28
PRINT 42
PRINT 93
C PRINT PAGE E (METRIC)
PRINT 26, ((MESS(LL), TIMA(LL), (ARREY(I, LL, 5), I=1, 10)), LL=1, L)
PRINT 19
NPAG = NPAGE(I)
I = I + 1
GO TO (81, 82, 83, 84, 85, 86, 87, 88), NPAG
86 CONTINUE
PRINT 106, KPAG
PRINT 38
PRINT 44
PRINT 95
C PRINT PAGE F (METRIC)
PRINT 27, ((MESS(LL), TIMA(LL), (ARREY(I, LL, 6), I=1, 10)), LL=1, L)
PRINT 19
87 CONTINUE
88 CONTINUE
100 CONTINUE
DO 75 J=1, 30
MESS(J) = BLKK
75 CONTINUE
IPRINT = 0
LINF = 0
900 RETURN
END
CRUNGE  SURP, RUNGF  --  4TH ORDER R-K INTEGRATION

SUBROUTINE RUNGE(ACCEL)

DOUBLE PRECISION ST

DOUBLE PRECISION DP

COMMON/STOR8/CMEA,RE,RA,RB,GO,GM,ALIM

COMMON/STOR9/S,NK

COMMON/STOR10/ST

COMMON/STOR19/TIME,T0

COMMON/STOR25/LOW,EPTINY,EPBIG,PTI,P1,P12,DT,HIGH

DIMENSION S(9,6),ST(20,10),K(6,4)

DIMENSION DP(6)

REAL LOW,K

BEGIN RUNGE-KUTTA INTEGRATION

N = 1

I = 2

350 CONTINUE

TIME0 = TIME

DO 400 J=1,6

K(J,1) = DT*S(J+3,1)

S(J+2) = S(J,1) + K(J,1)/2.

400 CONTINUE

TIME = TIME0 + DT/2.

NK = 2

CALL ACCEL

DO 404 J=1,6

K(J,2) = DT*S(J+3,2)

S(J+3) = S(J,1) + K(J,2)/2.

404 CONTINUE

TIME = TIME0 + DT/2.

NK = 3

CALL ACCEL

DO 408 J=1,6

K(J,3) = DT*S(J+3,3)

S(J,4) = S(J,1) + K(J,3)

408 CONTINUE

TIME = TIME0 + DT

NK = 4

CALL ACCEL

DO 411 J = 1,6

K(J,4) = DT*S(J+3,4)

ST(J+10,:1) = (K(J,1) + 2*K(J,2) + 2*K(J,3) + K(J,4))/6.
411. CONTINUE
C........ STORE THE SOLUTION
DO 490 L=1,6
ST(L+1) = ST(L*N) + ST(L+10*L)
490 CONTINUE
C
C
494. IF(I.EQ.4) GO TO 600
IF(I.EQ.2) GO TO 495
GO TO 497
495 CONTINUE
TIME = TIME - DT
DT = DT/2.
I = I + 1
N = 1
GO TO 350
497 CONTINUE
I = I + 1
N = 3
DO 498 J = 1,6
S(J+1) = ST(J+3)
498 CONTINUE
NK = 1
CALL ACCEL
GO TO 350
400 CONTINUE
DT = 2*DT.
DO 602 J=1,6
IF(DABS(ST(J+2)) .LT. 1.E-07
1 * OR * DABS(ST(J+4)) .LT. 1.E-07) GO TO 601
DPL(J) = DABS(L*0 - ST(J+4) + T(J+2))
GO TO 602
601 DPL(J) = 0.0
602 CONTINUE
FRAC = DMAX1(DP(4),DP(5),DP(6))
C........ COMPARE THE DT SOLUTION WITH THE DT/2 SOLUTION.
IF(FRAC .LT. EPSINY) GO TO 610
IF(FRAC .GT. EPRIG) GO TO 620
GO TO 630
610 DT = 2*DT.
IF(DT .GT. HIGH) DT = HIGH
GO TO 630
620 IF(DT .LT. LOW) GO TO 630
TIME = TIME - DT
DT = DT/2
N = 1
I = 2
DO 622 J=1, 9
ST(J,11) = ST(J,11)
622 CONTINUE
DO 621 L=1, 6
ST(L,2) = ST(L,3)
621 CONTINUE
GO TO 350
630 CONTINUE
IF(DT .LT. LOW) DT = LOW
I = 2
N = 1
ST(10,N) = TIME
DO 635 M=1, 6
C STORE THE FINAL SOLUTION
ST(M,N) = ST(M+4) + (ST(M+4) - ST(M+2))/15
635 CONTINUE
DO 650 J=1, 6
ST(J,N) = ST(J,N)
650 CONTINUE
TIME = ST(10,N)
NK = 1
C COMPUTE NEXT ACCELERATION
CALL ACCEL
RETURN
END
COMMON /ST03/YL/VY.L.$;XY-?.GP,
COMMON /STOR1/ALT;TEMP,PRES;DENS;VISC;SOUND,
COMMON /STOR2/X,Y,Z,R,
COMMON /STOR4/AREA,MCH,CDD,CD,MACH,NCN,CD;DRAG,
COMMON /STOR5/TWT;NT,MPR,WEIGHT,MASS,
COMMON /STOR7/XDOT,YS,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT,
COMMON /STOR12/YL,VYL,LYX,GAML,RLDOT,
COMMON /STOR13/XV,VL;THETA,VXL,VZL;ALPH,GAMA,CAZ,SAZ,
COMMON /STOR15/O,VX,VY,VZ,XT,EL,AZ,PANGE,LAT,Long,ACC,DLAT,
COMMON /STOR16/SA ARRAY,NN,
COMMON /STOR19/TIME,TO,
COMMON /STOR20/AREA,IOPT,TIM,NTH,THS,THRUST,
COMMON /STOR27/IPLAT,IIPL,LIPL,IIPTIM,
COMMON /STOR28/PLCH,ALZCH,ELLCH,VELCH,FLCH,NLCH,Z1CH,FDLCH,1
NDLCH,2DLCH
COMMON /STOR29/RRAD,AZRAD,ERAD,ELRAD,VELRAD,ERAD,RRAD,EDPAD,
1 NRRAD,2RRAD,
COMMON /STOR30/DELTA,VEL,ACCI,
REAL,IIPLAN,IIPL,LIPL,IIPTIM,
REAL,RLAD,NLCH,2RLAD,NDLCH
RFAL,LA,MOS,MASS,LOW,LONG,LAT,MACH,MCH
DIMENSION,TIM(50),THS(50),TWT(50),MPR(50),MACH(50),CDD(50),
DIMENSION,SA(10,2,10),ARRAY(10,30,10),
REAL,MASS,
N=NN,
J=1
SA(1*N,J)=ALPH
SA(2*N,J)=THETA
SA(3*N,J)=EL
SA(4*N,J)=AZ
SA(5*N,J)=ALT
SA(6*N,J)=RANGE/6076.155
SA(7*N,J)=DLAT
SA(8*N,J)=LONG,
J=2
SA(1*N,J)=THRUST
SA(2*N,J)=WEIGHT
SA(3*N,J)=DRAG
SA(4*N,J)=MACH
SA(5*N,J)=O
SA(6,N,J) = ACC
SA(7,N,J) = VT
SA(8,N,J) = MASS
J = 3
SA(1,N,J) = XL
SA(2,N,J) = YL
SA(3,N,J) = ZL
SA(4,N,J) = VXL
SA(5,N,J) = VYL
SA(6,N,J) = VZL
SA(7,N,J) = RXY
SA(8,N,J) = GAML
SA(9,N,J) = VEILCH
SA(10,N,J) = RLDOT
J = 4
SA(1,N,J) = RPAD/6076,1155
SA(2,N,J) = AZRAD
SA(3,N,J) = FLRAD
SA(4,N,J) = VELPAD
SA(5,N,J) = ERAD/6076,1155
SA(6,N,J) = NPAD/6076,1155
SA(7,N,J) = ZRAD/6076,1155
SA(8,N,J) = EDRAD
SA(9,N,J) = NDRAD
SA(10,N,J) = ZDRAD
J = 5
SA(1,N,J) = IIPLAT
SA(2,N,J) = IIPLOX
SA(3,N,J) = IIPRX/6076,1155
SA(4,N,J) = IIPTX
J = 6
SA(1,N,J) = R
SA(2,N,J) = VEL
SA(3,N,J) = ACCI
RETURN
END
CTABLE     SUB TABLE --- THrust, Weight, Drag INPUT

SUBROUTINE TABLE
COMMON/STOR/5/NEW, WRT, WPR, WGT
COMMON/STOR/6/PO, WTM, WPP2, WTP
COMMON/STOR/20/AREA, IOPT, TIM, NTH, THS, THRUST
COMMON/STOR/21/MESS, PHAS
COMMON/STOR/24/PHT, NID, NTA, NPH, KPH

10 FORMAT(1X,46)
DIMENSION MFSS(50)
DIMENSION PHT(14), NID(14), PTM(6), TSEP(6), WTP(6)
DIMENSION TIM(50), THS(50), TWT(50), WPR(50), MCH(50), CDD(50)
REAL LAMDA, VASS, K, LOW, LONG, LAT, MACH, MCH, LAMDAO
NAMFLIST /NAM2/APFA, TIM, NTH, THS, IOPF
NAMFLIST /NAM3/TWT, NWT, WPR
NAMFLIST /NAM4/SARFA, MCH, NCD, CDD
IF(NPH.GT. NTA) GO TO 900
NNNN = NID(KPH)
GO TO (343, 344, 345, 349), NNNN
243 READ(5, 10) PHAS
GO TO 342
244 CONTINUE
C READ IN DRAG TABLE
READ(5, 10) PHAS
READ(5, NAM4)
GO TO 340
245 CONTINUE
C READ IN THRUST, WEIGHT, AND DRAG TABLES
READ(5, 10) PHAS
READ(5, NAM2)
DO 346 J=1, NTH
TIM(J) = TIM(J) + PHT(NPH)
246 CONTINUE
READ(5, NAM3)
DO 347 J=1, NWT
TWT(J) = TWT(J) + PHT(NPH)
247 CONTINUE
WPP2 = WPP2 - WPR(1)
READ(5, NAM4)
GO TO 349
240 CONTINUE
900 RETURN
END
COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
DIMENSION MCH(50),CDD(50)

REAL MACH,MCH

IF(MACH .LE. MCH(1)) GO TO 116
IF(MACH .GE. MCH(NCD)) GO TO 115

DO 100 J = 2, NCD

IF(MACH .LT. MCH(J)) GO TO 110

100 CONTINUE

110 FRAC = (MACH - MCH(J-1))/(MCH(J) - MCH(J-1))
CD = CDD(J-1) + FRAC*(CDD(J) - CDD(J-1))
GO TO 200

115 CD = CDD(NCD)
GO TO 200

116 CD = CDD(1)

200 CDS = CD*SAREA
RETURN
END
SUBROUTINE TAB2

COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
COMMON/STOR6/PO,KRM,NDL,KPP2,WTM
COMMON/STOR19/TIM,TO
COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
DIMENSION TIM(50),THS(50),WTM(6)

IF(TIME.LT.TIM(J)) GO TO 115

IF(TIME.GE.TIM(NTH)) GO TO 115

DO 100 J=2,NTH

IF(TIME.LT.TIM(J)) GO TO 110

CONTINUE

110 TFRAC = (TIME - TIM(J-1))/(TIM(J) - TIM(J-1))

THRF = THS(J-1) + TFRAC*(THS(J) - THS(J-1))

GO TO (210,211)*IOPT

210 THRUST = THRFF + AREA*(PO - PRFS)

GO TO 900

211 THRUST = THRFF - AREA*PRES

GO TO 900

115 THRUST = 0.0

900 RETURN

END
SUBROUTINE TAB3

COMMON/STOR5/TWT,NWT,WRPR,WEIGHT
COMMON/STOR6/PO,WIM,WPL,WPP2,WTM
COMMON/STOR19/TIME,T0

DIMENSION TWT(50),WRPR(50),WTM(6)

IF(TIME.GE.TWT(NWT)) GO TO 115

DO 100 J=2,NWT
   IF(TIME.LT.TWT(J)) GO TO 110

  100 CONTINUE

  110 TFRAC = (TIME - TWT(J-1))/(TWT(J) - TWT(J-1))
       WPP = WRPR(J-1) + TFRAC*(WRPR(J) - WRPR(J-1))
       GO TO 116

  115 WPP = WRPR(NWT)

  116 WEIGHT = WPP + WPP2 + WRM + WPL

  200 RETURN

END
SPECIAL TAPE OUTPUT

**CTP**

**SUP**

**TOUT**

**SOUTIME**

**TOUT**

**COMMON/STOR16/SA,ARRAY,NN**

**COMMON/STOR18/FRAC,LINF,PTIME,CONV,END,IPRT,NPAGE,KPAG**

**COMMON/STOR34/KLIN**

**DIMENSION TPA(75),SA(10,2,10),ARRAY(10,30,10),NPAGE(10)**

**L = KLIN**

**TPA( 1) = PTIME**

**TPA( 2) = ARRAY(2,L,2)**

**TPA( 3) = ARRAY(1,L,2)**

**TPA( 4) = ARRAY(3,L,6)**

**TPA( 5) = ARRAY(1,L,6)**

**TPA( 6) = ARRAY(5,L,1)**

**TPA( 7) = ARRAY(6,L,1)**

**TPA( 8) = ARRAY(4,L,7)**

**TPA( 9) = ARRAY(5,L,2)**

**TPA(10) = ARRAY(7,L,2)**

**TPA(12) = ARRAY(8,L,1)**

**TPA(14) = ARRAY(1,L,1)**

**TPA(18) = ARRAY(3,L,1)**

**TPA(20) = ARRAY(4,L,1)**

**TPA(23) = ARRAY(2,L,1)**

**TPA(37) = ARRAY(7,L,2)**

**TPA(55) = ARRAY(1,L,1)**

**TPA(56) = ARRAY(2,L,3)**

**TPA(57) = ARRAY(3,L,2)**

**TPA(58) = ARRAY(4,L,3)**

**TPA(59) = ARRAY(5,L,3)**

**TPA(60) = ARRAY(6,L,3)**

**TPA(61) = ARRAY(7,L,3)**

**TPA(62) = ARRAY(8,L,3)**

**TPA(63) = ARRAY(9,L,3)**

**TPA(64) = ARRAY(10,L,3)**

**TPA(65) = ARRAY(4,L,5)**

**TPA(66) = ARRAY(1,L,5)**

**TPA(67) = ARRAY(2,L,5)**

**TPA(68) = ARRAY(3,L,5)**

**WRJTE(11),TPA(J),J=1,75**

**DO 100 J=1,75**

**TPA(J) = 0.0**

**100 CONTINUE**

**RETURN**

**END**
**CTRANS1**

**SUBROUTINE**: CTRANS1

**PARAMETERS**: EX, EY, EZ, WX, WY, WZ

**DOUBLE PRECISION**: TMA, SMA

**COMMON/STOR3**

**DIMENSION**: TMA(3,3), SMA(3,3)

```
WX = TMA(1,1)*EX + TMA(1,2)*EY + TMA(1,3)*EZ
WY = TMA(2,1)*EX + TMA(2,2)*EY + TMA(2,3)*EZ
WZ = TMA(3,1)*EX + TMA(3,2)*EY + TMA(3,3)*EZ
RETURN
END
```