LANDER
Program Manual

A Lunar
Ascent and Descent
Simulation

Prepared for the
National Aeronautics and Space Administration
Johnson Space Center
Advanced Programs Office
as part of the
Advanced Space Transportation Support Contract (ASTS)
and the
Lunar Base Systems Study (LBSS)

Contract Number: NAS 9-17878

by
Eagle Engineering, Incorporated
Report Number: 88-195

September 30, 1988
FOREWORD

This report documents the first edition of LANDER, a lunar ascent and descent trajectory simulation program. The purpose of the program is to provide delta velocity and trajectory information for lunar ascent and descent between low lunar orbit and the lunar surface. This information will aid in the formulation of plans to return to the Moon.

Dr. J.W. Alred was the NASA technical monitor for the ASTS contract. Mr. A. Petro was the NASA task monitor for this activity. The Eagle project manager was Mr. W.R. Stump. Special thanks go to Mr. J. Funk for his helpful advice and valuable assistance. This program was written and documented by Mr. C.C. Varner.
TABLE OF CONTENTS

Foreword ........................................... ii
Table of Contents ................................ iii
List of Figures ....................................... iv
List of Tables ....................................... v
1.0 Introduction ..................................... 1
2.0 Program Operation .............................. 2
3.0 Data Entry Subroutine ......................... 5
4.0 Variable Initialization ......................... 13
5.0 Integration Subroutine ......................... 15
6.0 Equations of Motion ........................... 19
   6.1 Control Procedures ...................... 23
   6.2 Thrust Profile .............................. 25
7.0 Output Subroutine ............................. 26
8.0 Orbit Subroutine ............................... 28
Appendix A -- List of Variables and Arrays .... 32
Appendix B -- Program Listings .................. 35
Appendix C -- Examples of Output .............. 73
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Main Program Flow Chart</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Orbital Elements</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Data Entry Flow Chart</td>
<td>6</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Approach Paths for Landing</td>
<td>11</td>
</tr>
<tr>
<td>Figure 5</td>
<td>The Approach Azimuth</td>
<td>12</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Heading, Azimuth, and Inclination</td>
<td>13</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Variable Initialization Flow Chart</td>
<td>15</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Integrator Flow Chart</td>
<td>18</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Equations of Motion</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Spherical Coordinates</td>
<td>23</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Thrust Pitch Angle</td>
<td>24</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Control Model</td>
<td>25</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Lander Thrust Profile</td>
<td>26</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Output Flow Chart</td>
<td>28</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Orbit Flow Chart</td>
<td>30</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Radial Coordinates</td>
<td>31</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Calculation of the Pseudo-inclination</td>
<td>10</td>
</tr>
<tr>
<td>Table 2</td>
<td>Heading and Azimuth Relationship to Pseudo-inclination</td>
<td>11</td>
</tr>
<tr>
<td>Table A1</td>
<td>Variable Arrays</td>
<td>33</td>
</tr>
<tr>
<td>Table A2</td>
<td>Variables</td>
<td>33</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

LANDER is a computer program used to predict the trajectory and flight performance of a spacecraft ascending or descending between a low lunar orbit of 15 to 500 nautical miles (nm) and the lunar surface. It is a three degree-of-freedom simulation which is used to analyze the translational motion of the vehicle during descent. Attitude dynamics and rotational motion are not considered.

The program can be used to simulate either an ascent from the Moon or a descent to the Moon. For an ascent, the spacecraft is initialized at the lunar surface and accelerates vertically away from the ground at full thrust. When the local velocity becomes 30 ft/s, the vehicle turns downrange with a pitch-over maneuver and proceeds to fly a gravity turn until Main Engine Cut-off (MECO). The spacecraft then coasts until it reaches the requested holding orbit where it performs an orbital insertion burn.

During a descent simulation, the lander begins in the holding orbit and performs a deorbit burn. It then coasts to pericynthion, where it reignites its engines and begins a gravity turn descent. When the local horizontal velocity becomes zero, the lander pitches up to a vertical orientation and begins to hover in search of a landing site. The lander hovers for a period of time specified by the user, and then lands.

Newton-Raphson iteration techniques are used to optimize the pitch-over maneuver and the MECO time for proper orbit insertion. Integration is performed using a Runge-Kutta fourth order integrator. This integrator has been verified with launch simulations of the Titan and Conestoga launch vehicles. LANDER receives input, presents output, and does all calculations in English units. The basic coordinate system is spherical. The moon is modelled as a spherical body of uniform gravity having no atmosphere and no gravitational harmonics.

Even though the output for a descent simulation appears to start at orbit and end at the surface, the mathematical calculations are performed in reverse. The program actually initializes the lander at the lunar surface and proceeds to simulate an ascent using negative mass flow. After the proper orbit has been achieved the data is reorganized and printed in the proper chronological sequence for a descent. Note: that this "reversed flight" is only characteristic of the descent simulations.
2.0 PROGRAM OPERATION

LANDER has a main driver program which accesses nine subroutines, and two function routines. In FORTRAN versions, the function routines are not necessary; the FORTRAN language has standard functions which perform the same operations. The main program controls the flow of operations to and from the subroutines. The subroutines perform activities such as input, output, and analytical calculations. The function routines perform basic numerical or mathematical calculations. A flow chart of the main program is shown in Figure 1.

The program must first define the functions and dimension the arrays that are to be used. BASIC versions of LANDER have a function equation for the ArcCosine since it is not an intrinsic function within this programming language. This function equation has some singularities which are corrected using tests within the ArcCosine function routine at lines 22000-22360. The ArcTangent 360 function routine between lines 21000 and 21250 is also necessary in BASIC versions of LANDER since the intrinsic ArcTangent function does not test for quadrant.

In the Data Entry subroutine, the user enters (or tells the program where to find) data that is used during the simulation. The following information is necessary for operation.

- The longitude and latitude of the landing site
- The weight of the payload to be carried
- The rocket characteristics such as maximum thrust level, specific impulse, inert weight, and propellant weight
- The amount of time that the lander is expected to hover before landing
- The holding orbit apocynthion and pericynthion
- The holding orbit inclination
- The estimated pitch-over angle
- The estimated main engine cut-off time

The Data Entry subroutine returns this information to the main program which immediately transfers control to the Variable Initialization subroutine.

A trajectory may be run numerous times during a simulation. Each time, it must start with the same initial conditions. The Variable Initialization subroutine sets all of the preliminary variables used during the integration calculations.

The Integration subroutine uses the state vector and the Equations of Motion to determine a new state vector at a future time (1 second later). The integration technique is a 4th order Runge-Kutta method, which makes four estimates of how the state vector changes during the time step. These estimates are then weighted and averaged to obtain a state vector change which has fourth order accuracy (0.01%).

Optimal step size control is not utilized. The time step of one second is fixed for the duration of program execution. This simplifies output operations at the expense of integration time efficiency.
Figure 1: Main Program

1. Define Functions
2. Dimension Array
3. Data Entry Subroutine
4. Variable Initialization Subroutine
5. Integration Subroutine
6. Output Subroutine

   - Is Time Less Than Runtime?
     - Yes
     - No
       - Orbit Subroutine
       - Modify Flight Path Angle at "Pitch-Over"
       - Is the Flight Path Angle at MECO Equal to Zero
         - Yes
         - Modify the MECO Time
         - No
           - Is the Apogee of the Ascent Orbit Equal to the Holding Orbit?
             - Yes
             - Output Subroutine
             - No

End
After integrating and obtaining the new state vector, the main program checks to see if the MECO time has been exceeded. If so, then the main program exits the integration loop and begins final orbit calculations. Otherwise, it loops back to the point just after Variable Initialization and performs another integration.

At specific time increments the program outputs important information about the flight. During Variable Initialization, the output time increment is set to five (5) seconds. The information presented to the screen during this intermediate output phase includes the time, altitude, range, velocity, flight path angle, heading, acceleration, thrust, and weight. In addition to this information, data such as rate of change of flight path angle, longitude, and latitude are output to a file called "OUTPUT.PRN" ("LOUTPUT.DAT" in FORTRAN versions).

The final/initial orbit is evaluated in the Orbit subroutine. This subroutine calculates the apocynthion, pericynthion, inclination, longitude of the ascending node, argument of pericynthion, and the eccentricity of the orbit entered. These terms are known as orbital elements and are shown graphicly in Figure 2. They are printed to the screen and to the output file.
Figure 2: Orbital Elements
3.0 DATA ENTRY SUBROUTINE

The Data Entry subroutine is the section of the program that asks the user for the information required to run the simulation. In BASIC versions, the user is first prompted for the letter designation of the storage drives used for input and output.

- Drive for Input data files
- Drive for Output data files
Choose 'F' for File Entry or 'M' for Manual Entry.
Is this to be an Ascent or a Descent simulation?

Different drives may be assigned to perform the input and output functions. Storage drives are only important for the versions of this program that are used by personal computers (PCs). On PCs, it is common to save input and output data on disks when it is being held for archive purposes. For working files, disk access is a slow process; storage and retrieval of data from the Random Access Memory (RAM) is much faster. Therefore it is common to transfer archived data to the RAM drive, and then assign the RAM drive to handle all input and output operations.

BASIC versions of LANDER also allow the lander characteristics to be provided via an input file. If the user selects this option, he or she should type an "F" or "f" when prompted with, "Choose 'F' for File Entry or 'M' for Manual Entry." Any answer other than "F" or "M" will result in the question being restated.

If an ascent simulation is desired, then the user should type an "A" when asked for the type of simulation. A descent simulation can be performed by typing a "D". Entry of any set of characters not beginning with an "A" or a "D" will result in the question being reprompted.

The longitude and latitude of the landing site must be specified on the next input screen (See Below). Longitude meridians are measured in degrees east of the Prime Meridian which passes through the Earth-Moon line on the Earth side of the Moon. Values between 0° and 360° East longitude can be used. The latitude is measured north from the lunar equator. Southern latitudes are indicated as a negative. Latitudes between 90° and -90° North latitude are permissible.
Figure 3: Data Entry Flow Chart

1. Data Entry Subroutine
   - Enter The Drives For Input And Output
   - Enter the Type of Simulation (Ascent or Descent)
   - Will The Rocket Data Be Entered From A File?
     - Yes: Retrieve Lander Characteristics From "LANDER.DAT" File
     - No: Input Rocket Characteristics
   - Save Lander Characteristics To "LANDER.DAT" File
   - Enter Payload
   - Enter Holding Orbit Data
   - Enter the Pitch-Over Angle and the MECO Time
   - Calculate Azimuth For Landing
   - Calculate The Heading For Landing
   - Save All The Input Data To "LAUNCH.DAT" File
Lunar Landing Site
Landing Site Latitude (-90 to +90)
Landing Site Longitude (0 to 360)

***** Vehicle Configuration *****

Payload Weight <lb>

Inert Weight <lb> | Propellant Weight <lb>
Thrust <lbf> | Specific Impulse <s>
Hover Time <s>

If the user chose to input the lander characteristics manually, then the inert weight, fuel weight, maximum thrust, specific impulse, and hover time of the lander must be provided. The program will then save the information on the output drive in a file called "LANDER.DAT". If the file entry method is adopted, then the program will search on the input drive for the "LANDER.DAT" file, and read the data in that file. Keep in mind that this option is only available for BASIC versions of LANDER.

The inert weight is the weight of the structure and equipment necessary for spacecraft operation. (All weights are Earth weights; the force measured by a scale on the surface of the Earth.) The propellant weight is supplied for ascent purposes. For ascent the weight of the propellant must be known in advance. It is added to the inert weight and the payload weight to obtain the spacecraft weight prior to lift-off. During descent simulations, the propellant weight is calculated and does not necessarily need to be input. The maximum thrust must be included since the thrust profile (discussed in section 6.2) is normalized to the maximum thrust. A constant propellant specific impulse is assumed throughout the flight, and the hover time can be of any length requested by the user.

The user is then queried for a payload weight. The payload is a constant mass element which is not an integral portion of the lander structure (i.e. not part of the inert weight).
The next input screen appears as follows:

Time to Main Engine Cut-off (MECO) ? _____ <s>
Holding Orbit (______ <nm> X ______ <nm>)

The spacecraft will perform a vertical rise (Flight Path Angle (Gamma) = 90 deg.) for the first few seconds of flight. At a relative velocity of 30 ft/s a pitch-over maneuver is executed; and the vehicle will momentarily thrust along a flight path defined by the user (Good Value = 70°)

Flight path angle at pitch-over ? ___

Holding orbit inclination ? (0° to 360°) ___
Do you wish to see the trajectory of each iteration ? ___

The user must provide an initial estimate to the simulation run time. The simulation runs until Main Engine Cut-off (MECO). 300 seconds is typical for an ascent simulation, while 450 to 500 seconds are good values for descent simulations. The holding orbit is the orbit from which the lander will begin its descent or to which the ascent spacecraft will inject after launching from the Moon.

The program requests that the user supply an initial value for the pitch-over flight path angle. If the flight path angle at the end of the simulation (MECO) is greater than zero (0), then the pitch-over angle is too high and the simulation is rerun with a lower pitch-over angle. The reverse is true if the flight path angle at MECO is less than zero (0). The process is iterative, and it requires several attempts to obtain the proper flight path angle at MECO. If the flight path angle at MECO is zero (0), then the final orbit is analyzed. If the resulting orbit is too high, then the simulation is terminated sooner (MECO time is reduced). Using the shorter simulation time the final flight path angle may not be zero (0), and must, therefore, be reiterated. If the resulting orbit is too low, then simulation is terminated later. Again, the process is iterative, and several modifications of the MECO time are necessary to obtain a solution.

The holding orbit inclination must be greater than the latitude of the landing site. If the landing site is at 45 degrees North or South latitude, then the true orbit inclination must be at least 45 degrees. From a mathematical point of view the orbit can never have a true inclination of more than 90 degrees. However, the latitude of the launch site and the true inclination of the orbit are not sufficient to define the direction from which the lander will make its approach. As Figure 4 demonstrates a lander attempting to land at site "A" from an orbit of true inclination "i" can be approaching from four different directions.
Posigrade orbits, those traveling in the direction of planetary rotation -- left to right, and retrograde orbits, traveling opposite the planetary rotation -- right to left, can approach a specified landing site from either the North or South. In order to show from which direction the lander is approaching the landing site, or the ascent spacecraft is heading from the launch site, this program allows the user to input an inclination that may be greater than 90°. If the input inclination is less than 90° (i.e. the input inclination equals the true inclination) then the spacecraft is flying from South to North in a posigrade orbit (Case I of figure 4). If the spacecraft is flying from South to North in a retrograde orbit, then the user supplies an inclination (pseudo-inclination) that is greater than 90° but less than 180° (See case II of figure 4). For a true inclination of "i" (between 0° and 90°), Table 1 shows how to calculate the pseudo-inclination which should be input to the program at the "inclination" prompt. When the flight is from North to South in a retrograde orbit, then the pseudo-inclination is between 180° and 270° (Case III -- figure 4). Finally for North to South flights in a posigrade orbit, the user should input a pseudo-inclination between 270° and 360° (Case IV -- figure 4). If the user inputs a pseudo-inclination which is less than 0° or greater than 360°, then the program will reprompt for the inclination.

Table 1: Calculation of the Pseudo-inclination

<table>
<thead>
<tr>
<th>Case</th>
<th>Orbit Type</th>
<th>Direction of Flight</th>
<th>Pseudo-inclination Equals</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Posigrade</td>
<td>From S. to N.</td>
<td>i</td>
</tr>
<tr>
<td>II</td>
<td>Retrograde</td>
<td>From S. to N.</td>
<td>180° - i</td>
</tr>
<tr>
<td>III</td>
<td>Retrograde</td>
<td>From N. to S.</td>
<td>180° + i</td>
</tr>
<tr>
<td>IV</td>
<td>Posigrade</td>
<td>From N. to S.</td>
<td>360° - i</td>
</tr>
</tbody>
</table>

The approach azimuth is calculated from the pseudo-inclination and the latitude of the landing/launch site through the use of right-spherical triangles (Figure 5).

The approach azimuth is an angle between -90° and 90°. Table 2 is used to relate the approach azimuth to the approach heading. The heading is measured from the North clockwise, and has a value between 0° and 360°. This is shown graphically in Figure 6.
Figure 4: Approach Paths for Landing
Table 2: Heading and Azimuth Relationship to Pseudo-inclination

<table>
<thead>
<tr>
<th>Pseudo-Inclination</th>
<th>Approach Azimuth (AZH)</th>
<th>Approach Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>0° to 90°</td>
<td>Positive</td>
<td>AZH</td>
</tr>
<tr>
<td>90° to 180°</td>
<td>Negative</td>
<td>360° + AZH</td>
</tr>
<tr>
<td>180° to 360°</td>
<td>Pos. or Neg.</td>
<td>180° + AZH</td>
</tr>
</tbody>
</table>

During each iteration of the trajectory, the program calculates numerous variables. The data stored in these variables can be sent to the screen if desired. However, if the program is running properly, then the extra output is unnecessary and time consuming. The trajectory data output can be turned off by answering "No" when asked, "Do you wish to see the trajectory of each iteration?"

**Figure 5: The Approach Azimuth**

\[
\text{Azimuth} = \sin^{-1} \left[ \frac{\cos (\text{inclination})}{\cos (\text{latitude})} \right]
\]
Figure 6: Heading, Azimuth, and Inclination
4.0 VARIABLE INITIALIZATION SUBROUTINE

A trajectory may be run numerous times during a simulation. It is important that each run start with the same initial conditions, and that stored data is not randomly retrieved during the simulation. During Variable Initialization all variables that are to be used during the simulation are set to their initial values. At the end of Variable Initialization the state vector, describing the initial conditions under which the vehicle is operating, is formulated.

The state vector describes characteristics of the vehicle which are not constant. These characteristics are referred to as parameters. Examples of "state" parameters include: the vehicle's position, velocity, and mass. The payload and inert weight are not considered "state" parameters because these vehicle characteristics are constant.

There are seven parameters which define the present "state" of the vehicle. The position is described by three parameters: the distance, the longitude angle, and the latitude angle. The mass of the vehicle represents the fourth state parameter. The vehicle's velocity is also described with three parameters: the radial range rate, the angular rate of longitude, and the angular rate of latitude.

All seven parameters form a vector or an array which, when integrated, creates a new state vector. The new state vector describes the conditions of the vehicle in the future; at the end of the time step. The ability to integrate the state vector is what makes it possible to determine the new position, velocity, and mass of the vehicle at the future time.

At the end of the Variable Initialization subroutine, a data storage file, called "LOUTPUT.PRN", is opened on the output drive. The file has a "PRN" extension so that it can be recognized by LOTUS (A spreadsheet programming language) as an input/output data file. In FORTRAN versions, this file is called "LOUTPUT.DAT".
Figure 7: Variable Initialization Flow Chart

Variable Initialization Subroutine

Define the Initial Values

Define the Initial Equations

Define the Secondary Equations

Open the Data Output Files
5.0 INTEGRATION SUBROUTINE

LANDER makes use of a Runge-Kutta fourth order integration routine. The derivation of this integrator is discussed in the second edition of Curtis F. Gerald's *Applied Numerical Analysis*, on page 259. A summary of this discussion and how it applies to the program is necessary in order to clarify the coding process.

The primary equation is obtained by substituting "t" for "X", and then "X" for "Y" in the equation presented by Gerald. In this equation the "K" values are estimates of the X. The weighted average of these estimates is the increment to \( X_{n+1} \) from \( X_n \).

\[
1) \quad X_{n+1} = X_n + \frac{(K1 + 2*K2 + 2*K3 + K4)}{6} \\
\text{Where:} \quad K1 = dt * f(X_n, t_n) \\
K2 = dt * f(X_n + K1/2, t_n + dt/2) \\
K3 = dt * f(X_n + K2/2, t_n + dt/2) \\
K4 = dt * f(X_n + K3, t_n + dt) \\
dt = Time \ step \\
t = Independent \ time \ variable \\
X = Time \ dependent \ state \ variable
\]

The function "f" is known as the Equation of Motion (EOM); and "\( X_n \)" is the state vector at time "t".

Example: In the spring (S), mass (M), damper (D) system shown,

\[
\text{the Equation of Motion is:} \quad M \frac{d^2x}{dt^2} + D \frac{dx}{dt} + Sx = 0
\]
Figure 8: Integration Flow Chart

Integration Subroutine

Save the State Vector at Time "t"

Calculate the Equations of Motion at the Initial State (I.S.)

Determine the First Estimate for the Change in State ($K_1$)

Propagate the Initial State Vector Forward by $K_1/2$ and Half the Time Step ($dt/2$)

Evaluate the Equations of Motion at the New State

Calculate the Second Estimate for the Change in State ($K_2$)

Propagate the Initial State Vector Forward by $K_2/2$

Evaluate the Equations of Motion at the New State

Calculate the Third Estimate for the Change in State ($K_3$)

Propagate the Initial State Vector Forward by $K_3$ and the Full Time Step ($dt$)

Evaluate the Equations of Motion at the New State

Calculate the Fourth Estimate for the Change in State

Calculate the State Vector at the Time "t+dt" Using a Weighted Average of the Four Change in State Estimates
The equation of motion is usually written in following form.

\[
f = \frac{d^2x}{dt^2} = -D \frac{dx}{dt} - Sx
\]

K1 is calculated by evaluating the EOM at state "Xn" and taking the product with the time step. Subsequently, K2 is determined after evaluating the EOM at the new state of "Xn + K1/2", and the new time "t_n + dt/2". K3 and K4 are similarly obtained after two more evaluations of the EOM. Once the "K" values have been determined, "X_{n+1}", the state vector at time "t + dt", is calculated using Equation 1.
6.0 EQUATIONS OF MOTION

Numerous calculations are necessary in order to set up the equations of motion. These preliminary calculations must be complete before integration can proceed.

The overall control parameter is the time variable. The time is what the program uses to determine when to stop the simulation. If the simulation time has not exceeded the user supplied stop time, then the main program continues to increment the time and loop back to the Integration subroutine. When the simulation time does exceed the stop time, the main program transfers control to the Orbit calculation subroutine, and then proceeds to the final output sequence.

The velocity is determined from the state vector in spherical coordinates \((R, \theta, \phi, M, \dot{R}, \dot{\theta}, \dot{\phi})\). "R" is the radial distance from the center of the Moon. "\(\theta\)" is the angle of longitude, measured East from the Prime Meridian. "\(\phi\)" is the angle of latitude, measured North from the Equator. "M" is the mass of the lander. "\(\dot{R}\)" is the radial range rate outward from the Moon's center. "\(\dot{\theta}\)" is the angular rate of change of longitude. And "\(\dot{\phi}\)" is the angular rate in latitude. The inertial velocity is calculated with the following equations:

**INERTIAL VELOCITY COMPONENTS**

\[
\begin{align*}
V_r &= \text{Radial Velocity} = \dot{R} \\
V_\theta &= \text{Longitudinal Velocity} = R \cdot \dot{\theta} \cdot \cos(\phi) \\
V_\phi &= \text{Lateral Velocity} = R \cdot \dot{\phi}
\end{align*}
\]

Inertial speed is the root sum square of the three velocity components shown above. The local velocity can be determined by reducing the longitudinal velocity component by the rotation rate of the moon.

**LOCAL VELOCITY COMPONENTS**

\[
\begin{align*}
V_{1r} &= V_r \\
V_{1\theta} &= V_\theta - R \cdot \Omega \cdot \cos(\phi) \\
V_{1\phi} &= V_\phi
\end{align*}
\]

Where: \(\Omega = \text{Rotation rate of the Moon} = 2.26622 \times 10^{-4} \text{ <rad/s>}\)
StarUEnd Subroutine

Increment the Time forward by the Time Step

Calculate the Inertial Velocity

Calculate Local Speed

Calculate the Heading

Calculate the Flight Path Angle

Determine the Control Procedure (Control Subroutine)

Determine the Thrust Profile (Profile Subroutine)

Calculate the Thrust

Calculate the Mass Flow

Calculate the Gravity

Formulate the Equations of Motion
The altitude above the surface of the Moon is calculated. Orientation of the thrust vector (GAMT) is determined in the Control subroutine. The level of thrust (PRF) is calculated in the Profile subroutine. The thrust (T1), the propellant mass flow (MDOT), the local acceleration of gravity (G), the heading (HEAD), weight (WEIGHT), and thrust to weight (TTOW) are also calculated in this section.

The program uses spherical coordinates to evaluate the motion of the lander. In spherical coordinates, "r" is the radial distance, "θ" is the longitude angle from the inertial X axis, and "φ" is the latitude angle measured from the equator.

In spherical coordinates, the equations of motion for a spacecraft of mass \( m \) under the influence of thrust \( T \) and gravity \( g \) are given in Equation 4.

\[
\begin{align*}
\ddot{r} &= r\dot{\theta}^2\cos(\phi)^2 + r\dot{\phi} + \frac{T\sin(\gamma)}{m} - g \\
\ddot{\theta} &= \frac{2\dot{\phi}\sin(\phi) - 2r\dot{\phi}}{\cos(\phi)} + \frac{T\cos(\gamma)\sin(h)}{mr\cos(\phi)} \\
\ddot{\phi} &= \frac{T\cos(\gamma)\cos(h)}{mr} - \dot{\theta}^2\cos(\phi)\sin(\phi) - \frac{2r\dot{\phi}}{r}
\end{align*}
\]

Where: \( \gamma = \) Flight Path Angle
\( h = \) Heading

The flight path angle is measured up from the local horizon, and the heading is measured clockwise from North.
Figure 10: Spherical Coordinates
6.1 CONTROL PROCEDURES

The Control subroutine provides the thrust orientation for the lander throughout the descent. The thrust vector is controlled through a pitch angle (GAMT). The thrust pitch angle can vary from 0° (tangential to the lunar surface) to 90° (normal to the surface).

The lander begins its descent from orbit using a gravity turn trajectory. As it slows the flight path angle gradually increases from 0°. Ten (10) seconds before the velocity reaches 30 ft/s, the lander initiates the pitch-over maneuver which is designed to reduce the horizontal velocity to zero. The thrust pitch angle is reoriented to the pitch-over angle (GAMP) during the next five (5) seconds. Then it orients to 90° (vertical) during the following five (5) seconds.

At the end of the pitch-over maneuver the lander is descending at 30 ft/s and has no horizontal velocity. The lander continues to decelerate until it is descending at 1.6 ft/s, basically hovering. A 1.6 ft descent/hover is then maintained until touchdown. The altitude at which the lander reaches 1.6 ft/s descent velocity is dependent upon the amount of time that the user wishes the vehicle to hover.

During ascent, the spacecraft launches at full thrust vertically until it reaches 30 ft/s local velocity. It performs a ten second pitch-over maneuver, and flies a gravity turn to orbit.

**Figure 11: Thrust Pitch Angle**
Figure 12: Control Model

Orbit Prior To Descent

Gravity Turn

Pitch Over Maneuver

Vertical Descent And Hover

Touch Down

Lunar Surface

Begin Descent
6.2 THRUST PROFILE

The thrust profile subroutine is accessed in the preliminary calculations of the Equations of Motion subroutine. This subroutine returns the level of thrust (PRF) as a percentage of the maximum thrust. The thrust level is dependent on time and local weight. From initiation of the descent to 35 seconds prior to hover the thrust level is set to maximum thrust. During the next 35 seconds, the thrust is linearly reduced to a level that is equal to the local weight of the lander. During this 35 seconds, the vertical descent velocity is reduced to 1.6 ft/s, and the horizontal velocity is nulled during pitch-over.

![Figure 13: Lander Thrust Profile](image)

The thrust profile for an ascent from the surface is a constant, and is held at maximum thrust. Thrust profile modifications can be accomplished by rewriting the Thrust Profile subroutine.
7.0 OUTPUT SUBROUTINE

The output subroutine is the portion of the program that controls when, where, and what information is to be presented to the screen and data storage files. The output data file is called LOUTPUT.PRN (LOUTPUT.DAT in FORTRAN version). This file has a ".PRN" extension in the BASIC version which allows it to be recognized as a data file by LOTUS. LOTUS is a spreadsheet program used to create graphical output from mass data. Other graphics programs may be used as long as they can read ASCII sequential data files.

Every 5 seconds of simulation time the output subroutine prints to the screen the time, altitude, range, velocity, flight path angle (Gamma), heading, thrust/weight, thrust, and the weight. Examples of typical ascent and descent screen output are provided in Appendix C. In addition to this data, longitude, latitude, angle of attack, rate of change of angle of attack, and the rate of change of flight path angle are saved to the LOUTPUT.*** file. The velocity and flight path angle are presented in local coordinates. When using local coordinates, the velocity and flight path angle are always given with respect to the launch site.

When the program is finished the output subroutine displays and saves the orbital parameters and the performance delta velocity. The orbital parameters consist of the apocynthion, pericynthion, inclination, longitude of the ascending node, argument of pericynthion, and the eccentricity (refer to Figure 2). The performance delta velocity ($\Delta V$) is the ideal velocity change that could be made with the fuel used if there are no gravity losses.

If the output interval needs to be changed, then it can be changed manually in the Initialization subroutine. The variable to be changed is called OUTINT. If the output interval is less than the integration step size (DT), then output will occur during each integration step.
Figure 14: Output Flow Chart

Output Subroutine

If The Interactions are Within Tolerance

Yes

If This is an Ascent Simulation

No

Reverse the Output Data

Yes

Print Orbit, Performance and Fuel Required

Print Heading & Flight Parameters

If This is a Descent Simulation

No

Print Orbit Performance and Fuel Required

Yes

If the Orbit has Been Cleared

No

Store the Trajectory Data

Yes

Store the Orbital Performance and Fuel Rqmt.

Print Iteration Data

If the Trajectory Run is Complete

No

Reset the Output Time

Yes
8.0 ORBIT SUBROUTINE

The orbit subroutine calculates the orbital elements of the orbit from which the lander is to descend. The orbital elements of interest are the apocynthion altitude, the pericynthion altitude, the inclination (i), the longitude of the ascending node (Ω), the argument of pericynthion (ω), and the eccentricity (e). Figure 2 is useful for visualizing these elements. The pericynthion altitude is the altitude of the spacecraft when it is at the perifocus of the orbit. Apocynthion altitude is the altitude when the spacecraft is opposite the perifocus. The eccentricity of the orbit is a measure of its ellipticity.

The orbital elements are calculated from the position and velocity vectors. The velocity vector must be in radial coordinates, and the position vector needs to be in rectangular inertial coordinates.

Radial coordinates are defined such that the X axis is aligned with the radial position vector from the center of the planet, the Y axis is parallel to the equatorial plane, and the Z axis is normal to the X-Y plane (Figure 16).

Since the velocity vector is normally in inertial coordinates (V_i), it must be converted to radial coordinates (V_r). This can be accomplished with Equation 5. The conversion to radial coordinates "r" from inertial coordinates "i" is achieved through vector multiplication of successive orthogonal rotation matrices for the longitude rotation (θ) and the latitude rotation (φ).

\[
5) \quad V_r = [M(\phi)] [M(\theta)] V_i
\]

Where:
\[
[M(\phi)] = \begin{bmatrix}
\cos(\phi) & 0 & -\sin(\phi) \\
0 & 1 & 0 \\
\sin(\phi) & 0 & \cos(\phi)
\end{bmatrix}
\]
\[
[M(\theta)] = \begin{bmatrix}
\cos(\theta) & \sin(\theta) & 0 \\
-\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
[C(\theta, \phi)] = [M(\phi)] [M(\theta)]
\]

\[
\begin{bmatrix}
\cos(\phi) \cos(\theta) & \cos(\phi) \sin(\theta) & -\sin(\phi) \\
-\sin(\theta) & \cos(\theta) & 0 \\
\sin(\phi) \cos(\theta) & \sin(\phi) \sin(\theta) & \cos(\phi)
\end{bmatrix}
\]
Figure 15: Orbit Flow Chart

1. Determine the Inertial to Radial Transformation Matrix
2. Convert Velocity in Inertial Coordinates to Velocity in Radial Coordinates
3. Convert Spherical Position Coordinates to Inertial Position
4. Calculate the Eccentricity
5. Determine the Inertial to Planer Transformation Matrix
6. Calculate the Inclination
7. Calculate the Longitude of the Ascending Node
8. Calculate the Argument of Perigee
9. Calculate the Apogee
10. Calculate the Perigee

Orbit Subroutine
Figure 16: Radial Coordinates

Prime Meridian

Equator

Latitude

Longitude

$Z^l$

$Y^l$

$X^l$

$Z^r$

$Y^r$

$X^r$
As mentioned earlier, the position vector must be in rectangular inertial coordinates, but it is given in spherical coordinates where "R" is the radial distance from the center of the planet, "θ" is the angle of longitude from the Prime Meridian (X axis), and "φ" is the angle of latitude from the equatorial plane. This conversion is shown in Equation 6.

\[
\begin{align*}
X &= R \times \cos(\phi) \times \cos(\theta) \\
Y &= R \times \cos(\phi) \times \sin(\theta) \\
Z &= R \times \sin(\phi)
\end{align*}
\]

Once the velocity and position vectors are in the proper coordinates, the orbital elements are calculated. A complete discussion of the calculation of orbital elements is beyond the scope of this report; but for those interested in the subject, a good treatment can be found in Chapter 17 "Satellite Photogrammetry" written by John L. Junkins from the Manual of Photogrammetry, 4th ed., American Society of Photogrammetry, Falls Church, Va., 1980.
APPENDIX A: VARIABLE DEFINITIONS

Table A1: Variable Arrays

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>C(3,3)</td>
<td>Coordinate System Transformation Matrix</td>
</tr>
<tr>
<td>K1(15)</td>
<td>1st Estimate for the Change of State</td>
</tr>
<tr>
<td>K2(15)</td>
<td>2nd Estimate for the Change of State</td>
</tr>
<tr>
<td>K3(15)</td>
<td>3rd Estimate for the Change of State</td>
</tr>
<tr>
<td>K4(15)</td>
<td>4th Estimate for the Change of State</td>
</tr>
<tr>
<td>M(4)</td>
<td>Mass Array</td>
</tr>
<tr>
<td>PSN(3)</td>
<td>Position Array</td>
</tr>
<tr>
<td>RKX(15)</td>
<td>Runge-Kutta State Vector</td>
</tr>
<tr>
<td>RKDX(15)</td>
<td>Runge-Kutta State Derivative</td>
</tr>
<tr>
<td>SDAT(5)</td>
<td>Stage Data Array</td>
</tr>
<tr>
<td>TRAJDAT(100,20)</td>
<td>Storage Array for Trajectory Data</td>
</tr>
<tr>
<td>VEL(3)</td>
<td>Relative Velocity Array</td>
</tr>
<tr>
<td>VSP(3)</td>
<td>Inertial Velocity Array</td>
</tr>
<tr>
<td>W(4)</td>
<td>Weight Array</td>
</tr>
<tr>
<td>X(15)</td>
<td>State Vector</td>
</tr>
</tbody>
</table>

Table A2: BASIC (FORTRAN) Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A$, B$, C$, D$, LOOP$</td>
<td>General Character Strings</td>
</tr>
<tr>
<td>ADOT</td>
<td>Rate of Change of Angle of Attack &lt;rad/s&gt;</td>
</tr>
<tr>
<td>ANGLE</td>
<td>Output Angle of the ArcTangent Function</td>
</tr>
<tr>
<td>ANS$ (BZ)</td>
<td>General Answer &lt;Character String&gt;</td>
</tr>
<tr>
<td>AOA</td>
<td>Angle of Attack &lt;rad&gt;</td>
</tr>
<tr>
<td>AOP</td>
<td>Argument of Pericynthion &lt;rad&gt;</td>
</tr>
<tr>
<td>APG</td>
<td>Apocynthion Radius &lt;ft&gt;</td>
</tr>
<tr>
<td>APGH</td>
<td>Former (Hold) Apocynthion Value &lt;ft&gt;</td>
</tr>
<tr>
<td>AZH</td>
<td>Heading Azimuth &lt;rad&gt;</td>
</tr>
<tr>
<td>COTG</td>
<td>CoTarget (Node Opposite Insertion)</td>
</tr>
<tr>
<td>COREL</td>
<td>Denominator for the ArcTangent Function &lt;n.d.&gt;</td>
</tr>
<tr>
<td>DG</td>
<td>Change in Flight Path Angle (Gamma) &lt;rad&gt;</td>
</tr>
<tr>
<td>DI$ (...)</td>
<td>Input Drive Letter &lt;Character&gt;</td>
</tr>
<tr>
<td>DR</td>
<td>Change in MECO Time (Runtime) &lt;s&gt;</td>
</tr>
<tr>
<td>DT0</td>
<td>Initial Step Size &lt;1 second&gt;</td>
</tr>
<tr>
<td>DT</td>
<td>Step Size &lt;1 second</td>
</tr>
<tr>
<td>DV</td>
<td>Performance Delta Velocity &lt;ft/s&gt;</td>
</tr>
<tr>
<td>DV2</td>
<td>Velocity Change for Insertion/Deorbit &lt;ft/s&gt;</td>
</tr>
<tr>
<td>DX$ (...)</td>
<td>Output Drive Letter &lt;Character&gt;</td>
</tr>
<tr>
<td>ECA</td>
<td>Eccentric Anomaly &lt;rad&gt;</td>
</tr>
<tr>
<td>ECC</td>
<td>Eccentricity &lt;n.d.&gt;</td>
</tr>
<tr>
<td>G0</td>
<td>Gravity at the Lunar Surface &lt;5.31 ft/s&gt;</td>
</tr>
<tr>
<td>GAM0</td>
<td>Former MECO Flight Path Angle &lt;rad&gt;</td>
</tr>
</tbody>
</table>
GAMFLAG (IGAMFLAG) - Flight Path Angle Iteration
Counter <Integer>

GAMH - Former Pitch-over Angle <rad>
GAMI - Inertial Flight Path Angle <rad>
GAML - Local Flight Path Angle <rad>
GAMP - Pitch-over Angle <rad>
GAMT - Thrust Elevation Angle <rad>
GE - Gravity at the Earth's Surface <32.2 ft/s>
H - Altitude <ft>
HEAD0 - Initial Heading <rad>
HEAD - Actual Heading <rad>
HEADD - Actual Heading <deg>
HEADT - Thrust Heading Angle <rad>

I, J, K, TEMP, TEMPl, TEMP2, TEMP3 - General Variables
IFLAG - Inertial Print Marker <0 - off, 1 - on>
INCL - Inclination <rad>
INCLN - Inclination <rad>
ITER - Iteration Print Counter <Integer>
LAN - Longitude of the Ascending Node <rad>
LATD - Latitude of the Spacecraft <deg>
LONGD - Longitude of the Spacecraft <deg>
(LNS) - Number of Output Data Lines <Integer>
MD1, MDOT - Rate of Fuel Use <slug/s>
MEA - Mean Eccentric Anomaly <rad>
MF - Final Mass after Insertion/before Deorbit <slugs>
MFU - Fuel Mass <slug>
MFUEL - Insertion/Deorbit Fuel Requirement <slugs>
MU - Lunar Gravitational Parameter <1.73x10^{11} ft^3/s^2>

NOFS (NOFZ) - Name Of File <Character String>
NUMOR - Numerator for the ArcTangent Function
OMEGA - Rotation Rate of the Moon <2.6622E-6 rad/s>

ORFLAG (IORFLAG) - Orbit Calculation Marker <1 - on, 0 - off>

OUTFLAG (IOUTFLAG) - Output Control Marker <Integer>
OUTINT - Time Between Outputs <s>
OUTTIME (OUTTIM) - Time of Output Printing <s>
PDOT - Angular Rate of Latitude <rad/s>
PFLAG (IPFLAG) - Pitch-over Angle <rad>
PHI0 - Angular rate of Latitude of Landing Site <rad/s>
PI - 3.14159 <rad/semicircle>
PPAD - Latitude of Landing Site <rad>
R0 - Radial Distance to Landing Site <ft>
RANGE - Groundtrack Distance to the Landing Site <ft>

33
RANGE2 - North Range Groundtrack Distance <ft>
RANGE3 - East Range Groundtrack Distance <ft>
RDA - Tangential Speed of the Atmosphere <ft/s>
RDOT - Rate of Radial Distance Change <ft/s>
PRF1, PRF - Normalized Thrust Level <n.d.>
Q1$ - Trajectory Print Marker <Character>
RTFLAG (IRTFLAG) - MECO Time Iteration Counter <Integer>
RTH - Former MECO Time <s>
RUNTIME (RUNTIM) - Simulation Stop Time <s>
SLR - Simi-latus Rectum <ft>
SMJ - Simi-major Axis <ft>
T1, T - Thrust <lbf>
TDOT - Angular Rate of Longitude <rad/s>
TGT - Target (Boost Orbit Apocynthion) Altitude <nm>
THETA - Output Angle for Cosine Function <rad>
THETA0 - Initial Longitude of Landing Site <rad>
TIME, TTEMP (TIM) - Simulation Time <s>
TPAD - Longitude of the Landing Pad <rad>
TTOW - Thrust to Weight <G’s>
TYP$ (TYPZ) - Simulation Type ('A' or 'D') <Character String>
V2 - Local Tangential Velocity Component <ft/s>
V - Local Speed of the Spacecraft <ft/s>
WEIGHT - Spacecraft Weight <lb>
WPLD - Payload Weight <lbf>
X - Temporary Variable for ArcCosine Function <n.d.>
APPENDIX B: Program Listings

BASIC Version
**TITLE**: Lunar Lander Trajectory Simulation

**NAME**: LANDER.BAS

**AUTHOR**: Chris Varner

**FOR**: Lunar Base Systems Study (LBSS)

**DATE**: 22 June, 1988

**PURPOSE**: The phase of flight between lunar orbit and the surface cannot be approximated using ideal free space equations. The lunar lander trajectory simulation is used to analyze the flight characteristics and the control requirements necessary for a descent to the lunar surface.

**NOTES**: Refer to the LANDER Program Manual for specific information on operation of this program.

**VARIABLES**: DTO = Initial Time step (1 second)

DX$ = Output Drive Letter <Character>

NOS$ = Name Of File <Character String>

ORFLAG = Orbit Calculation Flag ("O"-off; "1"-on)

OUTTIME = Time of next output <s>

TIME = Time of simulation <s>

-------------------------------

**User Defined Functions**

**PI** = 4 * ATN(1)

**DEF FNARCCOS (X)** = -ATN(X / SQR(-X * X + 1)) + PI / 2

**Dimension Arrays**

**DIM C(3, 3), K1(15), K2(15), K3(15), K4(15), M(4), P5(3), R(15)**

**DIM RKX(15), SDAT(5), TRAJDAT(100, 20), VEL(3), VSP(3), W(4), X(15)**

**Data Entry**

DX$ = "D"

GOSUB 10000

ITER = 0

RTFLAG = 1

**Begin Burn Time Iteration**
GAME'LAG = 1

\*

\* Begin Flight Path Angle Iteration \*
\*

\* Variable Initialization \*
\*

GOSUB 12000 'Variable Initialization
NOF$ = DX$ + ":OUTPUT.PRN"
OPEN "O", #3, NOF$

\* Start the Iteration/Integration Loop \*
\*

TIME = INT(TIME * 100) / 100

\* Integrate \*
\*

GOSUB 13000 'Runge-Kutta 4

\* Continue iteration sequence until the simulation time \*
*** exceeds the desired stop time (RUNTIME). ***

IF TIME < RUNTIME THEN 1340

\* Determine Orbital Parameters \*
\*

GOSUB 15000 'Orbit

\* Print the Orbital Parameters \*
\*

ORFLAG = 1
GOSUB 14000 'Output
CLOSE #3

\* Iterate the Pitch-over Flight Path Angle \*
\*

IF GAMFLAG > 1 THEN 1640
GAMH = GAMP
GAMP = GAMP + 2 * PI / 180
GOTO 1680
ELSE
TEMP = GAMP
DG = (GAMP - GAMH) * (0 - GAMI) / (GAMI - GAM0)
IF DG > 5 THEN DG = 5
IF DG < -5 THEN DG = -5
GAMP = GAMP + DG

37
1670          GAMH = TEMP
1680          ENDIF
1690          GAM0 = GAMI
1700          GAMFLAG = GAMFLAG + 1
1710          OUTFLAG = 1
1720          GOSUB 14000          'Output
1730          CLOSE #3
1740          IF ABS(GAMI) > .01 * PI / 180 THEN 1280
1742          '*******************************************************************************
1744          *** Iterate the MECO Time ***
1745          '*******************************************************************************
1750          IF RTFLAG > 1 THEN 1790
1760          RTH = RUNTIME
1770          RUNTIME = RUNTIME + 2
1780          GOTO 1830
1790          ELSE
1800          Tempo = RUNTIME
1810          DR = (RUNTIME - RTH) * (TGT - APG) / (APG - APGH)
1812          IF DR > 20 THEN DR = 20
1813          IF DR < -20 THEN DR = -20
1815          RUNTIME = RUNTIME + DR
1820          RTH = TEMPO
1830          ENDIF
1840          APGH = APG
1850          RTFLAG = RTFLAG + 1
1860          IF ABS(RTH - RUNTIME) > 1 AND X(4) > M(1) THEN 1210
1861          '*******************************************************************************
1862          *** Print Final Output ***
1863          '*******************************************************************************
1865          IF X(4) <= M(1) THEN 1985
1870          NOFS$ = DX$ + ":LOUTPUT.PRN"
1880          OPEN "I", #3, NOFS$
1890          TEMP = 0
1900          FOR I = 1 TO 18
1910          IF EOF(3) THEN 1940
1920          INPUT #3, TRAJDAT(TEMP, I)
1940          ENDIF
1950          NEXT I
1960          IF EOF(3) THEN 1970 ELSE 1900
1970          OUTFLAG = 2
1980          GOSUB 14000          'Output
1982          GOTO 1987
1985          ELSE
1986          PRINT "*** Not Enough Propellant ***"
1987          'ENDIF
1990          KEY ON
Data Entry Subroutine

Name: DE
Author: Chris Varner
Date: 30 December, 1986

*** Purpose: This routine is used to enter the data required for program operation.

CLS : KEY OFF
LOCATE 10, 1
INPUT "Drive for Input data files ------------------------ ", DI$
INPUT "Drive for Output data files ------------------------ ", DX$
LOOP$ = "ON"
INPUT "Choose 'F' for File Entry or 'M' for Manual Entry. ", ANS$
IF ANS$ = "M" OR ANS$ = "m" THEN LOOP$ = "OFF"
IF ANS$ = "F" OR ANS$ = "f" THEN LOOP$ = "OFF"
IF LOOP$ = "ON" THEN LOOP$ = "ON"
INPUT "Is this to be an Ascent or a Descent simulation? ", TYP$
IF LEFT$(TYP$, 1) = "A" THEN TYP$ = "A": LOOP$ = "OFF"
IF LEFT$(TYP$, 1) = "a" THEN TYP$ = "A": LOOP$ = "OFF"
IF LEFT$(TYP$, 1) = "D" THEN TYP$ = "D": LOOP$ = "OFF"
IF LEFT$(TYP$, 1) = "d" THEN TYP$ = "D": LOOP$ = "OFF"
IF LOOP$ = "ON" THEN CLS
CLS
PRINT "Lunar Landing Site"
PRINT "Landing Site Latitude (-90 to +90) 
PRINT "Landing Site Longitude (0 to 360) 
PRINT "***** Vehicle Configuration *****"
PRINT "Payload Weight <lb>"
PRINT
PRINT "Inert Weight <lb>
PRINT "Thrust <lbf>"
PRINT "Hover Time <s>"
LOCATE 5, 45: INPUT "", PHI0
IF PHI0 < -90 OR PHI0 > 90 THEN 10500
LOCATE 6, 45: INPUT "", THETA0
IF THETA0 < 0 OR THETA0 > 360 THEN 10520
IF ANS$ = "F" OR ANS$ = "f" THEN 10680
LOCATE 20, 30: INPUT "", SDAT(1)
LOCATE 20, 70: INPUT "", SDAT(2)
LOCATE 21, 30: INPUT "", SDAT(3)
LOCATE 21, 70: INPUT "", SDAT(4)
LOCATE 22, 30: INPUT "", SDAT(5)
NOF$ = DX$ + ":LANDER.DAT"
OPEN "O", #1, NOF$
PRINT #1, SDAT(1)
PRINT #1, SDAT(2)
PRINT #1, SDAT(3)
PRINT #1, SDAT(4)
PRINT #1, SDAT(5)
GOTO 10760
ELSE
NOF$ = DI$ + ":LANDER.DAT"
OPEN "I", #1, NOF$
INPUT #1, SDAT(1)
INPUT #1, SDAT(2)
INPUT #1, SDAT(3)
INPUT #1, SDAT(4)
INPUT #1, SDAT(5)
ENDIF
CLOSE #1
LOCATE 10, 48: INPUT "", WPLD
CLS
LOCATE 1, 45: PRINT ": LOCATE 1, 1
LOCATE 1, 17: INPUT ":", TGT
LOCATE 11, 20: PRINT ""
LOCATE 2, 29: INPUT "", COTG
IF TGT > COTG THEN TEMP = TGT: TGT = COTG: COTG = TEMP
IF TGT >= 15 THEN 10930
LOCATE 10, 20
PRINT "*** The Orbit’s Minimum Pericynthion Altitude ***"
LOCATE 11, 20
PRINT "***
is 15 nautical miles. ***"

' ENDF

IF TGT < 15 THEN 10820

PRINT " The spacecraft will perform a vertical rise (Flight Path Angle"
PRINT " [Gamma] = 90 deg.) for the first few seconds of flight. At"
PRINT " a relative velocity of 30 ft/s a pitch-over maneuver is"
PRINT " executed; and the vehicle will momentarily thrust along a"
PRINT " flight path defined by the user (Good Value = 70)."

PRINT
INPUT "Flight path angle at pitch-over? ", GAMP

GAMP = GAMP * PI / 180

PRINT
INPUT "Holding orbit inclination? (0 to 360) ", INCL

INPUT "Do you wish to see the trajectory of each iteration ", Q1$ IF LEFT$(Q1$, 1) = "y" OR LEFT$(Q1$, 1) = "Y" THEN Q1$ = "Y" ELSE Q1$ = "N"

IF INCLN > (180 - ABS(PHI0)) AND INCLN < (180 + ABS(PHI0)) THEN 11070

IF INCLN < ABS(PHI0) OR INCLN > (360 - ABS(PHI0)) THEN 11070

INCLN = INCLN * PI / 180

X = COS(INCLN) / COS(PHI0 * PI / 180)

GOSUB 22000 'Inverse Cosine

AZH = -(THETA - PI / 2) 'ArcSine

IF INCLN <= PI / 2 THEN HEAD0 = AZH

IF (INCLN > PI / 2) AND (INCLN <= PI) THEN HEAD0 = 2 * PI + AZH

IF INCLN > PI THEN HEAD0 = PI + AZH

IF TYP$ = "D" THEN 11163 ELSE 11169

IF HEAD0 < PI THEN 11164 ELSE 11166

HEAD0 = HEAD0 + PI

GOTO 11168

ELSE

HEAD0 = HEAD0 - PI

ENDIF

ENDIF

CLS

PRINT "*** Calculating ***"

NOFS$ = DI$ + ":LAUNCH.DAT"

OPEN "O", #1, NOFS$

PRINT #1, THETA0, PHI0

FOR J = 1 TO 5

PRINT #1, SDAT(J)

NEXT J

PRINT #1, RUNTIME, TGT, COTG

PRINT #1, HEAD0

PRINT #1, WPLD, GAMP

CLOSE #1
11300 RETURN

12000 ' Variable Initialization Subroutine
12010 '-----------------------------------------------------
12020 '-----------------------------------------------------
12030 NOF$ = DI$ + ":LAUNCH.DAT"
12040 OPEN "I", #1, NOF$
12060 INPUT #1, THETAO, PHI0
12070 FOR J = 1 TO 5
12080 INPUT #1, SDAT(J)
12090 NEXT J
12100 INPUT #1, TEMP, TGT, COTG
12120 INPUT #1, WPLD, TEMP
12130 CLOSE #1
12140 IF TYP$ = "D" THEN SDAT(4) = -SDAT(4)
12142 DTO = 1
12150 DT = DTO
12160 GAMT = PI / 2
12170 GO = 1.62 * 3.28084 'Lunar Surface Gravity <ft/s^2>
12180 GE = 9.810001 * 3.28084 'Terrian Surface Gravity <ft/s^2>
12190 HEAD = HEADO
12200 IFLAG = 0
12210 MFU = 0
12220 OMEGA = 2.26622E-06 'Rotation rate of the Moon <Rad/s>
12230 ORFLAG = 0
12235 OUTINT = 5
12240 OUTTIME = -.0001
12250 PDOT = 0
12260 IF TYP$ = "D" THEN RDOT = .5 * 3.28084 ELSE RDOT = 0 'Surface Speed
12270 RO = 1740000! * 3.28084 'Lunar Radius <ft>
12280 TDOT = OMEGA
12290 TIME = 0
12300 MU = GO * RO ^ 2
12310 PHI0 = PHI0 * PI / 180
12320 THETA0 = THETA0 * PI / 180
12330 W(1) = SDAT(1) + WPLD 'dry weight
12340 M(1) = W(1) / GE 'mass
12350 M(2) = SDAT(2) / GE 'prop mass
12360 X(8) = M(1)
12370 X(1) = RO
12380 X(2) = THETA0
12390 X(3) = PHI0
12400 X(4) = M(1)
12410 IF TYP$ = "A" THEN X(4) = X(4) + M(2)
12420 X(5) = RDOT
12430 X(6) = TDOT
12440 X(7) = PDOT
13000 ' Integration Subroutine (Runge-Kutta 4)
13010 ' T
13020 ' TIME = T
13030 TTEMP = TIME
13040 FOR I = 1 TO 7
13050 RKX(I) = X(I)
13060 NEXT I
13070 GOSUB 20000
13072 '****************************************************************************************************
13074 '*** If it is time to output data goto the output subroutine ***
13075 '****************************************************************************************************
13076 IF TIME > OUTTIME THEN 13077 ELSE 13078
13077 GOSUB 14000 'Output
13078 'ENDIF
13080 FOR I = 1 TO 7
13090 K1(I) = RKDX(I) * DT
13100 RKX(I) = X(I) + .5 * K1(I)
13110 NEXT I
13120 TIME = TIME + .5 * DT
13130 GOSUB 20000
13140 FOR I = 1 TO 7
13150 K2(I) = RKDX(I) * DT
13160 RKX(I) = X(I) + .5 * K2(I)
13170 NEXT I
13180 GOSUB 20000
13190 FOR I = 1 TO 7
13200 K3(I) = RKDX(I) * DT
13210 RKX(I) = X(I) + K3(I)
13220 NEXT I
13230 TIME = TTEMP + DT
13240 GOSUB 20000
13250 FOR I = 1 TO 7
13260 K4(I) = DT * RKDX(I)
13270 X(I) = X(I) + (K1(I) + 2 * K2(I) + 2 * K3(I) + K4(I)) / 6
13280 NEXT I
13290 RETURN
14000 ' Output Subroutine
14010 ' Output Subroutine
14020 ' Output Subroutine
14030 IF OUTFLAG <> 2 THEN 14160
14040 IF TYP$ = "A" THEN 14104
14050 J = TEMP
14060 FOR I = 1 TO INT((TEMP - 1) / 2)
14070 FOR K = 2 TO 18
14080 TEMP1 = TRAJDAT(J - I, K)
14090 TRAJDAT(J - I, K) = TRAJDAT(I, K)
14100 NEXT K
14110 NEXT I
14120 GOSUB 14000
14130 RETURN
TRAJDAT(I, K) = TEMP1

NEXT K

NEXT I

PRINT " Weight Prior to Deorbit Burn <lb>: "; TRAJDAT(TEMP, 11)

PRINT " Delta Velocity Required to Deorbit" 

PRINT " to the Initial Descent Orbit <ft/s>: "; TRAJDAT(TEMP, 9)

PRINT " Fuel Required for the Deorbit Burn <lbf>: "; TRAJDAT(TEMP, 10)

PRINT " Initial Descent Orbit:" 

PRINT " Apocynthion <nm> -- "; TRAJDAT(TEMP, 2)

PRINT " Pericynthion <nm> -- "; TRAJDAT(TEMP, 3)

TRAJDAT(TEMP, 4) = 180 - TRAJDAT(TEMP, 4)

IF TRAJDAT(TEMP, 5) < 180 THEN 14089 ELSE 14091

TRAJDAT(TEMP, 5) = TRAJDAT(TEMP, 5) + 180

GOTO 14093

ELSE

TRAJDAT(TEMP, 5) = TRAJDAT(TEMP, 5) - 180

END IF

PRINT " Longitude of the Ascending Node <deg> -- "; TRAJDAT(TEMP, 5)

IF TRAJDAT(TEMP, 6) < 180 THEN 14096 ELSE 14098

TRAJDAT(TEMP, 6) = 180 - TRAJDAT(TEMP, 6)

GOTO 14100

ELSE

TRAJDAT(TEMP, 6) = 540 - TRAJDAT(TEMP, 6)

END IF

PRINT " Argument of Pericynthion <deg> -- "; TRAJDAT(TEMP, 6)

PRINT " Eccentricity <n.d.> -- "; TRAJDAT(TEMP, 7)

END IF

PRINT "Ideal Performance Delta Velocity is "; TRAJDAT(TEMP, 8)

IF TYP$ = "D" THEN 14153

PRINT " Orbit Attained:"
114135 PRINT " Apocynthion <nm> -- "; TRAJDAT(TEMP, 2)
114136 PRINT " Pericynthion <nm> -- "; TRAJDAT(TEMP, 3)
114137 PRINT " Inclination <deg> -- "; TRAJDAT(TEMP, 4)
114138 PRINT " Longitude of the Ascending Node <deg> -- "; TRAJDAT(TEMP, 5)
114139 PRINT " Argument of Pericynthion <deg> -- "; TRAJDAT(TEMP, 6)
114140 PRINT " Eccentricity <n.d.> -- "; TRAJDAT(TEMP, 7)
114141 PRINT
114142 PRINT " Velocity Required at Apocynthion to"
114143 PRINT " Achieve the Holding Orbit <ft/s>: "; TRAJDAT(TEMP, 10)
114144 PRINT
114149 PRINT " Fuel Required for the Apocynthion Burn<lb>: "; TRAJDAT(TEMP, 11)
114150 PRINT
114151 PRINT " Weight After Apocynthion Burn <lb>: "; TRAJDAT(TEMP, 12)
114152 PRINT " Weight of the Payload Placed in Orbit <lb>: "; TRAJDAT(TEMP, 9)
114153 ' ENDIF
114154 PRINT
114155 PRINT "********** Simulation Complete **********
114156 OUTFLAG = 0
114157 GOTO 14760
114160 'ELSE
114170 IF OUTFLAG = 0 THEN 14250
114180 ♦ ITER ♦ = ♦ ITER ♦ + 1
114190 PRINT "Iteration # "; ITER;
114200 PRINT " Apocynthion = "; APG;
114210 PRINT " <nm> Pericynthion = "; PEG;
114220 PRINT " <nm>
114230 OUTFLAG = 0
114240 GOTO 14750
114250 ' ELSE
114260 IF ORFLAG = 1 THEN 14270 ELSE 14550
114270 PRINT #3,
114280 PRINT #3, APG
114290 PRINT #3, PEG
114300 PRINT #3, INCL * 180 / PI
114310 PRINT #3, LAN
114320 PRINT #3, AOF
114330 PRINT #3, ECC
114340 IF TYP$ = "A" THEN 14370
114350 ♦ DV ♦ = ♦ SDAT ♦ (4) * ♦ GE ♦ * ♦ LOG ♦ (X(4) / M(1))
114360 GOTO 14390
114370 ' ELSE
114380 ♦ DV ♦ = ♦ SDAT ♦ (4) * ♦ GE ♦ * ♦ LOG ♦ ((M(1) + M(2)) / X(4))
114390 ' ENDIF
114400 PRINT #3, DV
114410 ' PRINT
114420 TEMP = 2 / (R0 + APG * 6076.1) - 2 / (2 * R0 + (APG + COTG) * 6076.1)
114430 DV2 = SQR(MU * TEMP)
45
14432 TEMP = 2 / (R0 + APG * 6076.1) - 2 / (2 * R0 + (APG + PEG) * 6076.1)
14434 DV2 = DV2 - SQR(MU * TEMP)
14440 MF = X(4) * EXP(-DV2 / SDAT(4) / GE)
14450 MFUEL = MF - X(4)
14460 IF TYP$ = "D" THEN 14490
14470 WPLD = MF * GE - SDAT(1)
14480 PRINT #3, WPLD
14490 ' ENDIF
14500 PRINT #3, DV2
14510 PRINT #3, MFUEL * GE
14520 PRINT #3, MF * GE
14530 ORFLAG = 0
14540 GOTO 14740
14550 ' ELSE
14560 TPAD = THETA0 + OMEGA * TIME
14570 PPAD = PHI0
14590 RANGE2 = 940 * (X(2) - TPAD) ' in nautical miles
14600 RANGE3 = 940 * (X(3) - PPAD) ' in nautical miles
14610 X = COS(X(2) - TPAD) * COS(X(3) - PPAD)
14620 GOSUB 22000 ' Arccosine
14630 RANGE = THETA * 940
14640 LONGD = (X(2) - OMEGA * TIME) * 180 / PI: LATD = X(3) * 180 / PI
14641 IF Q1$ = "N" THEN 14644
14642 U$ = "####.## ###### ###### ###### ####### #######
14643 PRINT USING U$; TIME, H, V, GAML * 180 / PI, T, WEIGHT
14644 ' ENDIF
14645 IF TYP$ = "D" THEN 14646 ELSE 14652
14646 IF HEADD < 180 THEN 14647 ELSE 14649
14647 HEADD = HEADD + 180
14648 GOTO 14651
14649 ' ELSE
14650 HEADD = HEADD - 180
14651 ' ENDIF
14652 ' ENDIF
14653 PRINT #3, TIME, H, RANGE, LONGD;
14660 PRINT #3, USING " ####.## ###### ######"; LATD, V, VSP(0);
14670 PRINT #3, USING " ####.## ###### ######"; GAML * 180 / PI, GAMI * 180 PI, AOA;
14680 PRINT #3, USING " ###### ######"; T, WEIGHT;
14690 PRINT #3, USING " ###### ###### ######"; ADOT, RANGE, RANGE2;
14700 PRINT #3, USING " ###### ###### ######"; RANGE3, TTOW, HEADD
14710 OUTTIME = OUTTIME + OUTINT
14720 OUTTIME = INT(OUTTIME)
14730 IF OUTTIME < TIME THEN 14710
14740 ' ENDIF
14750 ' ENDIF
14760 ' ENDIF

46
Orbital Parameters Subroutine

\[
\begin{align*}
C(1, 1) &= \cos(X(3)) \cdot \cos(X(2)) : C(1, 2) = \cos(X(3)) \cdot \sin(X(2)) \\
C(1, 3) &= \sin(X(3)) \\
C(2, 1) &= -\sin(X(2)) : C(2, 2) = \cos(X(2)) : C(2, 3) = 0 \\
C(3, 1) &= -\sin(X(3)) \cdot \cos(X(2)) : C(3, 2) = -\sin(X(3)) \cdot \sin(X(2)) \\
C(3, 3) &= \cos(X(3)) \\
\end{align*}
\]

FOR I = 1 TO 3

\[
\text{TEMP} = 0 \\
\text{TEMP} = \text{TEMP} + VSP(J) \cdot C(J, I) \\
\text{NEXT J} \\
\text{VEL}(1) = \text{TEMP} \\
\text{NEXT I}
\]

PSN(1) = X(1) \cdot \cos(X(3)) \cdot \cos(X(2))

PSN(2) = X(1) \cdot \cos(X(3)) \cdot \sin(X(2))

PSN(3) = X(1) \cdot \sin(X(3))

PSN(0) = \sqrt{PSN(1)^2 + PSN(2)^2 + PSN(3)^2)

VEL(0) = \sqrt{VEL(1)^2 + VEL(2)^2 + VEL(3)^2)

SMJ = 1 / (2 / PSN(0) - VEL(0)^2 / MU)

I = PSN(1) \cdot VEL(1) + PSN(2) \cdot VEL(2) + PSN(3) \cdot VEL(3)

J = 1 - PSN(0) / SMJ

K = I / SQR(MU * SMJ)

ECC = SQR(J^2 + K^2)

NUMOR = K: DENOM = J: GOSUB 21000 'ArcTan360

ECA = ANGLE

MEA = ECA - ECC \cdot \sin(ECA)

K = MU / J / PSN(0)

C(1, 1) = (K \cdot PSN(1) - I \cdot VEL(1)) / MU / ECC

C(1, 2) = (K \cdot PSN(2) - I \cdot VEL(2)) / MU / ECC

C(1, 3) = (K \cdot PSN(3) - I \cdot VEL(3)) / MU / ECC

SLR = SMJ * (1 - ECC^2)

J = PSN(0) - SLR

K = I / PSN(0)

C(2, 1) = (K \cdot PSN(1) - J \cdot VEL(1)) / ECC / SQR(MU * SLR)

C(2, 2) = (K \cdot PSN(2) - J \cdot VEL(2)) / ECC / SQR(MU * SLR)

C(2, 3) = (K \cdot PSN(3) - J \cdot VEL(3)) / ECC / SQR(MU * SLR)

C(3, 1) = C(1, 2) \cdot C(2, 3) - C(1, 3) \cdot C(2, 2)

C(3, 2) = C(1, 3) \cdot C(2, 1) - C(1, 1) \cdot C(2, 3)

C(3, 3) = C(1, 1) \cdot C(2, 2) - C(1, 2) \cdot C(2, 1)

X = C(3, 3)

GOSUB 22000 'ARCCOSIGN

INCL = THETA

NUMOR = C(3, 1): DENOM = -C(3, 2): GOSUB 21000 'ArcTan360

LAN = ANGLE
15460  LAN = LAN * 180 / 3.141592654#
15470  NUMOR = C(1, 3): DENOM = C(2, 3): GOSUB 21000 'ArcTan360
15480  AOP = ANGLE
15490  AOP = AOP * 180 / 3.141592654#
15500  APG = (SMJ * (1 + ECC) - R0) / 6078
15510  PEG = (SMJ * (1 - ECC) - R0) / 6078
15520  RETURN

20000 ' ---------------------------------------------
20010 ' |                                      |
20020 ' | Equations of Motion                    |
20025 ' | --------------------------------------- |
20030 '*******************************************************************************
20040 '**** Preliminary Calculations ***
20050 '*******************************************************************************
20060  DT = DT0
20080  RKK(8) = RKK(4)
20090  VSP(1) = RKK(5)
20100  VSP(2) = RKK(1) * RKK(6) * COS(RKK(3))
20110  VSP(3) = RKK(1) * RKK(7)
20120  VSP(0) = SQR(VSP(1) ^ 2 + VSP(2) ^ 2 + VSP(3) ^ 2)
20130  RDA = R0 * OMEGA * COS(PHI0)
20140  V2 = VSP(2) - RDA
20150  V = SQR(VSP(1) ^ 2 + V2 ^ 2 + VSP(3) ^ 2)
20160  H = RKK(1) - R0
20170  GAM1 = ATN(VSP(1) / SQR(VSP(2) ^ 2 + VSP(3) ^ 2))
20180  IF V2 = 0 AND VSP(3) = 0 THEN 20190 ELSE 20220
20190  GAML = 90 * PI / 180
20200  GOTO 20230
20210  'ELSE
20220  GAML = ATN(VSP(1) / SQR(V2 ^ 2 + VSP(3) ^ 2))
20230  'ENDIF
20240  NUMOR = RKK(7)
20250  DENOM = RKK(6) - OMEGA
20260  GOSUB 21000 'ArcTan 360
20270  IF ANGLE <= PI / 2 THEN HEAD = PI / 2 - ANGLE
20280  IF ANGLE > PI / 2 THEN HEAD = 5 * PI / 2 - ANGLE
20290  GOSUB 24000 'Control
20300  GOSUB 23000 'PROFILE
20310  T1 = SDAT(3) * PRF
20320  MD1 = -T1 / GE / SDAT(4)
20330  T = T1
20340  HEADD = HEAD * 180 / PI
20350  WEIGHT = RKK(4) * GE
20360  TTOW = T / WEIGHT
20370  MDOT = MD1
20380  FOR I = 1 TO 7
20390  RKK(I + 8) = RKK(I)
20400  NEXT I

48
Equations of Motion for Spherical Coordinates

\[
\begin{align*}
\text{RKDX}(1) &= \text{RKX}(5) \\
\text{RKDX}(2) &= \text{RKX}(6) \\
\text{RKDX}(3) &= \text{RKX}(7) \\
\text{RKDX}(4) &= \text{MDOT} \\
\text{TEMP1} &= 0; \text{TEMP2} = 0; \text{TEMP3} = 0 \\
\text{TEMP1} &= T \cdot \sin(\text{GAMT}) \\
\text{TEMP2} &= \left( T \cdot \cos(\text{GAMT}) \cdot \sin(\text{HEADT}) \right) / \left( \text{RKX}(4) \cdot \text{RKX}(1) \cdot \cos(\text{RKX}(3)) \right) \\
\text{TEMP3} &= \left( T \cdot \cos(\text{GAMT}) \cdot \cos(\text{HEADT}) \right) / \left( \text{RKX}(4) \cdot \text{RKX}(1) \right) \\
\text{RKDX}(5) &= \text{RKX}(1) \cdot \text{RKX}(6) \cdot 2 \cdot \cos(\text{RKX}(3)) \cdot 2 + \text{RKX}(1) \cdot \text{RKX}(7) \cdot 2 \\
\text{RKDX}(6) &= \text{RKDX}(5) - \text{MU} / \text{RKX}(1) \cdot 2 + \text{TEMP1} \\
\text{RKDX}(6) &= \text{RKDX}(6) + \text{TEMP2} \\
\text{RKDX}(7) &= -2 \cdot \text{RKX}(5) \cdot \text{RKX}(7) / \text{RKX}(1) \\
\text{RKDX}(7) &= \text{RKDX}(7) - \text{RKX}(6) \cdot 2 \cdot \sin(\text{RKX}(3)) \cdot \cos(\text{RKX}(3)) + \text{TEMP3} \\
\text{RETURN} \\
\end{align*}
\]

ArcTan360 Function

\[
\begin{align*}
\text{IF NUMOR} > 0 \text{ AND DENOM} = 0 \text{ THEN 21050 ELSE 21070} \\
\text{ANGLE} &= 3.141592654\# / 2 \\
\text{GOTO 21240} \\
\end{align*}
\]

ArcCosine Function

\[
\begin{align*}
\text{IF NUMOR} >= 0 \text{ AND DENOM} > 0 \text{ THEN 21130 ELSE 21150} \\
\text{ANGLE} &= \text{ATN(} \text{NUMOR} \text{ / DENOM)} \\
\text{GOTO 21220} \\
\end{align*}
\]
```plaintext
** PURPOSE: Outputs the ArcCosine of X as THETA.
** NOTES: Define the function:
  \[ \text{FNARCCOS}(X) = -\arctan\left(\frac{X}{\sqrt{-X^2 + 1}}\right) + \frac{3.141592654}{2} \]
  at the beginning of the main program.
** VARIABLES: THETA - The ArcCosine of X <rad>
  X - The adjacent/hypotenuse <n.d.>
** RESERVED VARIABLES: PI

** Define PI
  \[ \pi = 3.141592654 \]

** Test for singularities in the derived function.
  IF X > 1 THEN X = 1
  IF X < -1 THEN X = -1
  IF X = 1 THEN 22280 ELSE 22300
  THETA = 0
  GOTO 22350
  IF X = -1 THEN 22310 ELSE 22340
  THETA = PI
  GOTO 22350
  ** If there are no singularities, calculate arccosine of X.
  \[ \text{THETA} = \text{FNARCCOS}(X) \]
  ** END ARCCOSINE
  RETURN

---

Thrust Profile Subroutine

IF TYP$ = "A" THEN 23170
  IF TIME <= SDAT(5) THEN 23120
    IF TIME > SDAT(5) + 35 THEN 23080
    PRF = PRF1 + (1 - PRF1) / 35 * (TIME - SDAT(5))
  GOTO 23100
  ELSE
    PRF = 1
  ENDIF
GOTO 23150
```
Control Procedures Subroutine

**TITLE**: Control Procedures

**NAME**: LCONTROL

**AUTHOR**: Chris Varner

**FOR**: LAUNCH PROGRAM

**DATE**: 15 June, 1987

**PURPOSE**: Provides control and guidance for Eagle’s Ascent Program. The method of control is that of a zero angle of attack trajectory turn. (Gravity Turn).

1. IF V < 30 THEN 24150 ELSE 24210
   - GAMT = PI / 2
   - GAML = PI / 2
   - HEADT = HEAD0
   - HEAD = HEAD0
   - PFLAG = 0
   - GOTO 24470
2. ELSE
   - IF PFLAG < 20 THEN 24230 ELSE 24350
     - IF PFLAG > 10 THEN 24300
       - GAMT = GAMP - (90 * PI / 180 - GAMP) / 10 * PFLAG
       - GAML = GAMT
       - HEAD = HEAD0
       - HEADT = HEAD0
       - PFLAG = PFLAG + 1
       - GOTO 24330
     - ELSE
       - GAMT = GAMP + (GAML - GAMP) / 10 * (PFLAG - 10)
       - PFLAG = PFLAG + 1
     - ENDIF
     - GOTO 24460
   - ELSE
     - GAMT = GAML
   - END

51
24370 IF GAMT < 0 THEN GAMT = 0
24380 IF GAML > 80 * PI / 180 THEN 24390 ELSE 24430
24390 HEADT = HEAD0
24400 HEAD = HEAD0
24410 GOTO 24440
24420 ' ELSE
24430 ' HEADT = HEAD0
24440 ' ENDIF
24450 ' ENDIF
24460 ' ENDIF
24470 ' ENDIF
24480 ' ENDIF
24490 RETURN
FORTRAN Version
*** TITLE : Lunar Lander Trajectory Simulation
NAME : LANDER.BAS
AUTHOR : Chris Varner
TRANSLATOR : Mike D'Onofrio
FOR : Lunar Base Systems Study (LBSS)
DATE : 15 August 1988

*** PURPOSE: The phase of flight between lunar orbit and surface cannot be approximated using ideal free space equations. The lunar lander trajectory simulation is used to analyze the flight characteristics and the control requirements necessary for a descent to the lunar surface.

*** NOTES: Refer to the LANDER Program Manual for specific information on operation of this program.

*** Declare Variables **
IMPLICIT REAL *16 (A-Z)
INTEGER IGAME FLAG, I, IFLAG, ITER, J, K, LNS, IORFLAG, IOUTFLAG
INTEGER IPFLAG, IRTFLAG
CHARACTER LOOPZ*3, NOFZ*72, BZ*1, TYPZ*1

*** Dimension Arrays ***
DIMENSION C(3, 3), D6(5), K1(15), K2(15), K3(15), K4(15)
DIMENSION PSN(3), RKX(15), RKDX(15)
DIMENSION VEL(4), W(4)
COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
COMMON/TOTF/TRAJDAT (100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
COMMON/TOTG/WPLD, X(15)
A = 1.
PI = 4.* QATAN(A)
CALL DE (GAMP, RUNTIM)
ITER = 0
IRTFLAG = 1
X(4) = M(1) + 1
GAMP = GAMP * PI / 180.

*** Begin Burn TIM Iteration ***
DO 1890 WHILE (QABS(RTH - RUNTIM ).GT.1. .AND. X(4).GT.M(1))
IGAMFLAG = 1
GAMI = .2

*** Begin Flight Path Angle Iteration ***
DO 1880 WHILE (QABS(GAMI) .GT. 0.1 * PI / 180.)

*** Variable Initialization ***
CALL INITIALIZE (GAMP, X, OUTTIM, OUTINT)
LNS = 0
NOFZ = 'LOUTPUT.DAT'
OPEN (UNIT=10, FILE=NOFZ, STATUS='OLD', ERR=1335, +DISPOSE='DELETE')
CONTINUE
CLOSE (UNIT=10)
OPEN (UNIT=10, STATUS='NEW', FILE=NOFZ)

*** Start the Iteration/Integration Loop ***
DO 1340 WHILE (TIM .LT. RUNTIM)

TIM = QFLOAT ( INT( TIM * 100. )) / 100.
*** Integrate ***
CALL RK4 (DT, GAMP, LNS, OUTTIM, OUTINT, TIM, X)

*** Continue iteration sequence until the simulation **
*** time exceeds the desired stop time (RUNTIM). **

CONTINUE

*** Determine Orbital Parameters ***
CALL ORBIT (VSP, X, AOP, APG, ECC, AINCL, ALAN, PEG, SMJ)

*** Print the Orbital Parameters ***
IORFLAG = 1
CALL OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)

*** Print Final Output Sequence ***
IOUTFLAG = 1
CALL OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)
CLOSE (UNIT=10)

*** Modify the Pitch-over Angle ***
*** Newton-Raphson Iteration ***

IF ( IGAMFLAG .GT. 1 ) THEN

TEMP = GAMP
DG = (GAMP - GAMH) * ( -GAMI ) / ( GAMI - GAM0 )
IF ( DG .GT. 5.0 ) DG = 5.0
IF ( DG .LT. -5.0 ) DG = -5.0
GAMP = GAMP + DG
GAMH = TEMP
ELSE
GAMH = GAMP
GAMP = GAMP + 2.0 * PI / 180.
ENDIF
GAM0 = GAMI
IGAMFLAG = IGAMFLAG + 1
1880 CONTINUE

********************************************************************
C *** Modify the MECO Time ***
C *** Newton-Raphson Iteration ***
C ********************************************************************
IF (IRTFLAG .GT. 1) THEN
   TEMP = RUNTIM
   DR = (RUNTIM - RTH) * (TGT - APG) / (APG - APGH)
   IF (DR .GT. 50.) DR = 50.
   IF (DR .LT. -50.) DR = -50.
   RUNTIM = RUNTIM + DR
   RTH = TEMP
ELSE
   RTH = RUNTIM
   RUNTIM = RUNTIM + 2.
ENDIF
APGH = APG
IRTFLAG = IRTFLAG + 1
1890 CONTINUE
IF (X(4) .LE. M(1)) THEN
   PRINT *, '*** Not Enough Propellant ***'
ELSE
   NOFZ = 'LOUTPUT.DAT'
   OPEN (UNIT=10, STATUS='OLD', FILE=NOFZ)
   DO I=1, LNS
      READ (10, 1925), (TRAJDAT(1, J), J=1, 17)
   1925 FORMAT (1X, F5.0, F8.0, F7.0, 2F7.2, 2F7.0, 3F6.2, 2F9.0, +F6.2, 2F7.0, F5.2, F6.2)
   END DO
   READ (10, 1930), (TRAJDAT(LNS+1, J), J=1, 6)
   1930 FORMAT (1X, F7.1, F7.1, F8.2, F8.2, F8.2, F7.4)
   READ (10, 1932), TRAJDAT(LNS+1, 7)
   1932 FORMAT (1X, F6.1)
   IF (TYPZ .EQ. 'A') THEN
      READ (10, 1938), TRAJDAT (LNS+1, 8)
   1938 FORMAT (1X, F6.0)
      READ (10, 1940), (TRAJDAT(LNS+1, J), J=9, 11)
   1940 FORMAT (1X, F8.1, F8.2, F9.0)
   ELSE
      READ (10, 1945), (TRAJDAT(LNS+1, J), J=8, 10)
   1945 FORMAT (1X, F8.1, F8.2, F9.0)
   ENDIF
   CLOSE (UNIT=10)
   NOFZ = 'LRUN.DAT'
   OPEN (UNIT=10, STATUS='NEW', FILE=NOFZ)
IOUTFLAG = 2
CALL OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)
CLOSE (UNIT=10)
OPEN (UNIT=10, STATUS='OLD', FILE='LAUNCH.DAT',
+DISPOSE='DELETE')
CLOSE (UNIT=10)
ENDIF
STOP
END

----------------------------------------
Data Entry Subroutine
----------------------------------------

| Name  : DE            |
| Author: Chris Varner |
| Date  : 3 August, 1986 |

*** Purpose: This routine is used to enter the data required for program operation.

SUBROUTINE DE (GAMP, RUNTIM)

*** Declare Variables **
IMPLICIT REAL *16 (A-Z)
INTEGER I, J, K
CHARACTER LOOPZ*3, NOFZ*72, BZ*1, TYPZ*1

*** Dimension Arrays ***
DIMENSION SDAT(5)
LOOPZ = 'ON'
DO 2010 WHILE (LOOPZ .EQ. 'ON')
  WRITE (5, 1999)
  READ (6, 2000), TYPZ
  1999 FORMAT (' Is this to be an Ascent or a Descent Simulation ?')
  2000 FORMAT (A1)
  IF (TYPZ .EQ. 'A') LOOPZ = 'OFF'
  IF (TYPZ .EQ. 'a') THEN
    TYPZ = 'A'
    LOOPZ = 'OFF'
  ENDIF
  IF (TYPZ .EQ. 'D') LOOPZ = 'OFF'
  IF (TYPZ .EQ. 'd') THEN
    TYPZ = 'D'
    LOOPZ = 'OFF'
  ENDIF
2010 CONTINUE
WRITE (5, 2020)
2020  FORMAT (' Lunar Landing Site Latitude (-90 to 90) ') 
      PHI0 = 100. 
      DO 2035 WHILE (PHI0 .LT. -90. .OR. PHI0 .GT. 90.) 
         READ (6, *), PHI0 
      2035 CONTINUE 
      WRITE (5, 2038) 
      2038  FORMAT (' Landing Site Longitude (0 to 360) ') 
      THETA0 = 400. 
      DO 2045 WHILE (THETA0 .LT. 0. .OR. THETA0 .GT. 360.) 
         READ (6, *), THETA0 
      2045 CONTINUE 
      WRITE (5, 2050) 
      2050  FORMAT (115x ' Vehicle', 
               ' Configuration '14x ' ----' , 
               ' ---------------------- ', 
               ' ---'14x ' /') 
      WRITE (5, 2059) 
      READ (6, *), SDAT(1) 
      2059  FORMAT (' Inert Weight <lb>') 
      WRITE (5, 2064) 
      READ (6, *), SDAT(2) 
      2064  FORMAT (' Propellant Weight <lb>') 
      WRITE (5, 2069) 
      READ (6, *), SDAT(3) 
      2069  FORMAT (' Thrust <lbf>') 
      WRITE (5, 2074) 
      READ (6, *), SDAT(4) 
      2074  FORMAT (' Specific Impulse <s>') 
      WRITE (5, 2079) 
      READ (6, *), SDAT(5) 
      2079  FORMAT (' Hover Time <s>') 
      WRITE (5, 2084) 
      READ (6, *), WPLD 
      2084  FORMAT (' Payload Weight <lb>') 
      WRITE (5, 2089) 
      READ (6, *), RUNTIM 
      2089  FORMAT (115x ' Time to Main Engine Cut-off (MECO) ? <s> ') 
      TGT = 0. 
      DO 2120 WHILE (TGT .LT. 15.) 
         WRITE (5, 2099) 
         READ (6, *), TGT 
      2099  FORMAT (' Holding Orbit Pericynthion <nm>') 
      WRITE (5, 2104) 
      READ (6, *), COTG 
      2104  FORMAT (' Holding Orbit Apocynthion <nm>') 
      IF (TGT .GT. COTG) THEN 
         TEMP = TGT 
      END IF 
      END
TGT = COTG
COTG = TEMP
ENDIF
IF (TGT .LT. 15.) THEN
WRITE (5, 2110)
2110 FORMAT (/'' *** The Orbit's Minimum Altitude ***''/
       + '' *** is 15 nautical miles. ***''/)
ENDIF
2120 CONTINUE
WRITE (5, 2130)
2130 FORMAT (/'' The spacecraft will perform a vertical rise'','
       +' (Flight Path Angle'' (Gamma) = 90 deg.) for the first'','
       +' few seconds of flight. At a''
WRITE (5, 2140)
2140 FORMAT ('' relative velocity of 30 ft/s a pitch-over'','
       +' maneuver is executed;'' and the vehicle will'','
       +' momentarily thrust along a flight path'')
WRITE (5, 2144)
2144 FORMAT ('' defined by the user (Good Value = 70).''/
WRITE (5, 2149)
READ (6, *), GAMP
2149 FORMAT ('' Flight path angle at pitch-over? ''
AINCTN = 361
DO 2170 WHILE (AINCTN .LT. QABS(PHI0) .OR. AINCTN .GT. 360.
+ - QABS(PHI0) .OR. (AINCTN .GT. (180. - QABS(PHI0)) .AND.
+ AINCTN .LT. (180. + QABS(PHI0))))
IF (TYPZ .EQ. 'D') THEN
WRITE (5, 2150)
ELSE
WRITE (5, 2159)
ENDIF
READ (6, *), AINCTN
2150 FORMAT ('' Holding orbit inclination ? (0 to 360) ''
2159 FORMAT ('' Desired orbit inclination ? (0 to 360) ''
2170 CONTINUE
WRITE (5, 2179)
READ (6, 2184), BZ
2179 FORMAT ('' Do you wish to see the trajectory of each'','
       +' iteration?''
2184 FORMAT (A1)
PRINT *, *** Calculating ***
NOFZ = 'LAUNCH.DAT'
OPEN (UNIT=10, STATUS='NEW', FILE=NOFZ)
WRITE (10, 2190), AINCTN, THETAO, PHI0, TGT, COTG, WPLD,
       +(SDAT(J), J=1, 5), BZ, TYPZ
2190 FORMAT (1X, F6.2, F7.2, F6.2, 2F5.0, F8.2, 3F9.2, F7.2,
       +F5.1, ' ', A1, ' ', A1)
Variable Initialization Subroutine

<table>
<thead>
<tr>
<th>Name</th>
<th>INITIALIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Chris Varner</td>
</tr>
</tbody>
</table>

*** Purpose: This routine is used to set the variables to their initial values prior to entering the integration loop.

SUBROUTINE INITIALIZE (GAMP, X, OUTTIM, OUTINT)
*** Declare Variables **
IMPLICIT REAL *16 (A-Z)
INTEGER IFLAG, ITER, LNS, IORFLAG, IOUTFLAG
CHARACTER NOFZ*72, BZ*1, TYPZ*1
*** Dimension Arrays ***
DIMENSION W(4), X(15)
COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
COMMON/TOTG/WPLD, DI (15)

A = 1.
PI = 4. * QATAN(A)
NOFZ = 'LAUNCH.DAT'
OPEN (UNIT=10, STATUS='OLD', FILE=NOFZ)
READ (10, 2500), AINCLN, THETA0, PHI0, TGT, COTG, WPLD,
+ (SDAT(J), J=1, 5), BZ, TYPZ
2500 FORMAT (1X, F6.2, F7.2, F6.2, 2F5.0, F8.2, 3F9.2, F7.2,
+F5.1, ' ', A1, ' ', A1)
CLOSE (UNIT=10)
IF (BZ .EQ. 'Y') BZ = 'Y'
AINCLN = AINCLN * PI / 180.
A = QCOS(AINCLN) / QCOS(PHI0 * PI / 180.)
AZH = QASIN(A)
IF (AINCLN .LE. PI / 2.) HEAD0 = AZH
IF (AINCLN.GT.PI / 2. .AND. AINCLN.LE.PI) HEAD0 = 2.*PI + AZH
IF (AINCLN .GT. PI) HEAD0 = PI + AZH
IF (TYPZ .EQ. 'D') THEN
IF (HEADO .LT. PI) THEN
    HEADO = HEADO + PI
ELSE
    HEADO = HEADO - PI
ENDIF
SDAT(4) = -SDAT(4)
ENDIF
DTO = 1.
DT = DTO
FTPNM = 1852./0.3048
GAMT = PI / 2.
G0 = 1.7314E14 / 5710000. ** 2.
GE = 1.407646882E16 / 2.092567257E7 ** 2.
HEAD = HEADO
IFLAG = 0
OMEGA = 2.6622E-06
IORFLAG = 0
OUTINT = 5.
OUTTIM = 0
PDOT = 0.
RDOT = .5 * 3.28084
R0 = 5710000.
TDOT = OMEGA
TIM = 0.
MU = G0 * R0 ** 2.
PHI0 = PHI0 * PI / 180.
THETA0 = THETA0 * PI / 180.
W(1) = SDAT(1) + WPLD
M(1) = W(1) / GE
M(2) = SDAT(2) / GE
X(8) = M(1)
X(1) = R0
X(2) = THETA0
X(3) = PHI0
X(4) = M(1)
IF (TYPZ .EQ. 'A') X(4) = X(4) + M(2)
X(5) = RDOT
X(6) = TDOT
X(7) = PDOT
RETURN
END
SUBROUTINE RK4 (DT, GAMP, LNS, OUTTIM, OUTINT, TIM, X)

*** Declare Variables ***

IMPLICIT REAL *16 (A-Z)
INTEGER I, ID1, LNS

*** Dimension Arrays ***

DIMENSION K1(15), K2(15), K3(15), K4(15), RKX(15), RKDX(15)
COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, ID1, M(4), MU
COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0

TTEMP = TIM
DO 10 I=1, 7
     RKX(I) = X(I)
10    CONTINUE
CALL EOM (GAMP, RKX, RKDX)
IF (TIM .GE. OUTTIM) THEN
    CALL OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)
ENDIF

DO I= 1, 7
     K1(I) = RKDX(I) * DT
     RKX(I) = X(I) + .5 * K1(I)
END DO
TIM = TIM + 0.5 * DT
CALL EOM (GAMP, RKX, RKDX)
DO I=1, 7
     K2(I) = RKDX(I) * DT
     RKX(I) = X(I) + 0.5 * K2(I)
END DO
CALL EOM (GAMP, RKX, RKDX)
DO I=1, 7
     K3(I) = RKDX(I) * DT
     RKX(I) = X(I) + K3(I)
END DO
CALL EOM (GAMP, RKX, RKDX)
DO I=1, 7
     K4(I) = DT * RKDX(I)
END DO
RETURN
END
Output Subroutine

SUBROUTINE OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)
*** Declare Variables **
IMPLICIT REAL *16 (A-Z)
INTEGER ID1, ID2, ID3, I, IFLAG, ITER, J, K, LNS, IORFLAG
INTEGER IOUTFLAG
CHARACTER BZ*1, TYPZ*1
*** Dimension Arrays ***
COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, GO
COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, ID1, M(4), MU
COMMON/TOTD/OMEGA, ID2, ID3, PEG, PHI0, PI, BZ, R0
COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
COMMON/TOTF/TRAJDAT (100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
COMMON/TOTG/WPLD, X(15)

IF (IOUTFLAG .EQ. 2) THEN
  J = INT(LNS + 1)
  IF (TYPZ .EQ. 'D') THEN
    DO 3510 I=1, (J - 1) / 2
      DO 3500 K=2, 18
        TEMP1 = TRAJDAT(J - I, K)
        TRAJDAT(J - I, K) = TRAJDAT(I, K)
        TRAJDAT(I, K) = TEMP1
      3500 CONTINUE
    3510 CONTINUE
  WRITE (10, 3700), TRAJDAT(J, 10)
  WRITE (10, 3710), TRAJDAT(J, 1), COTG, TRAJDAT(J, 8)
  WRITE (10, 3720), TRAJDAT(J, 9)
  WRITE (10, 3515)
  3515 FORMAT (' Initial Descent Orbit:/'
    WRITE (10, 3730), TRAJDAT(J, 1)
    WRITE (10, 3740), TRAJDAT(J, 2)
    TRAJDAT(J, 3) = 180 - TRAJDAT(J, 3)
    WRITE (10, 3750), TRAJDAT(J, 3)
    IF (TRAJDAT(J, 4) .LT. 180.) THEN

63
TRAJDAT(J, 4) = TRAJDAT(J, 4) + 180.
ELSE
TRAJDAT(J, 4) = TRAJDAT(J, 4) - 180.
ENDIF
WRITE (10, 3760), TRAJDAT(J, 4)
IF (TRAJDAT(J, 5) .LT. 180.) THEN
TRAJDAT(J, 5) = 180. - TRAJDAT(J, 5)
ELSE
TRAJDAT(J, 5) = 540. - TRAJDAT(J, 5)
ENDIF
WRITE (10, 3770), TRAJDAT(J, 5)
WRITE (10, 3780), TRAJDAT(J, 6)
ENDIF
WRITE (10, 3520)
3520 FORMAT (' Time Altitude Range Velocity Gamma Heading',
+ ' Thrust Weight/', '<s> <ft> <nm> <ft/s>',
+ '<deg> <deg> <lbf> <lbm>')</n
DO I=1, J-1
IF (TYPZ .EQ. 'D') THEN
IF (TRAJDAT(I, 17) .LT. 180.) THEN
TRAJDAT(I, 17) = TRAJDAT(I, 17) + 180.
ELSE
TRAJDAT(I, 17) = TRAJDAT(I, 17) - 180.
ENDIF
ENDIF
WRITE (10, 3530), TRAJDAT(I, 1), TRAJDAT(I, 2),
+ TRAJDAT(I, 3), TRAJDAT(I, 6), TRAJDAT(I, 8), TRAJDAT(I, 17),
+ TRAJDAT(I, 11), TRAJDAT(I, 12)
3530 FORMAT (1X,F4.0,F10.0,F7.0,F10.0,F7.2,F8.2,F9.0,F8.0)
END DO
WRITE (10, 3550), TRAJDAT(J, 7)
3550 FORMAT (' Ideal Performance Delta Velocity is: ',F8.2,
+ '<ft/s>/)
IF (TYPZ .EQ. 'A') THEN
WRITE (10, 3555)
3555 FORMAT (' Boost Orbit:/)
WRITE (10, 3730), TRAJDAT(J, 1)
WRITE (10, 3740), TRAJDAT(J, 2)
WRITE (10, 3750), TRAJDAT(J, 3)
WRITE (10, 3760), TRAJDAT(J, 4)
WRITE (10, 3770), TRAJDAT(J, 5)
WRITE (10, 3780), TRAJDAT(J, 6)
WRITE (10, 3560), TRAJDAT(J, 1), COTG, TRAJDAT(J, 9)
3560 FORMAT (' Velocity Required at Apocynthion to Achieve:/
+ the Holding Orbit ( ',F4.0, ' X ',F4.0, ') :'
+ F9.2, '<ft/s>/)
TRAJDAT(J, 10) = -TRAJDAT(J, 10)
WRITE (10, 3570), TRAJDAT(J, 10)
3570 FORMAT (' Fuel Required for the Apocynthion Burn :', F9.2, +' <lbm>')
WRITE (10, 3580), TRAJDAT(J, 11)
3580 FORMAT (' Weight After Apocynthion Burn :', F9.2, +' <lbm>')
WRITE (10, 3590), TRAJDAT(J, 8)
3590 FORMAT (' Weight of the Payload Placed in Orbit:', F9.2, +' <lbm>')
ENDIF
WRITE (10, 3600)
3600 FORMAT ('*************** SIMULATION COMPLETE ***************')
IOUTFLAG = 0
ELSE
IF (IOUTFLAG .EQ. 1) THEN
ITER = ITER + 1
WRITE (5, 3610), ITER, APG, PEG
3610 FORMAT ('Iteration #', I3, ' Apocynthion = ', F7.1, '<nm>', +' Pericynthion = ', F7.1, '<nm>')
IOUTFLAG = 0
ELSE
IF (IORFLAG .EQ. 1) THEN
AINCLD = AINCL * 180. / PI
WRITE (10, 3620), APG, PEG, AINCLD, ALAN, AOP, ECC
3620 FORMAT (1X, F7.1, F7.1, F8.2, F8.2, F8.2, F7.4)
IF (TYPZ .EQ. 'D') THEN
DV = -SDAT(4) * GE * LOG(X(4) / M(1))
ELSE
DV = SDAT(4) * GE * LOG((M(1) + M(2)) / X(4))
ENDIF
WRITE (10, 3630), DV
3630 FORMAT (1X, F6.1)
TEMP = 2. / (R0 + APG * FTPNM) - 2. / (2. * R0 + (APG + COTG) * FTPNM)
DV2 = QSQRRT(MU * TEMP)
TEMP = 2. / (R0 + APG * FTPNM) - 2. / (2. * R0 + (APG + PEG) * FTPNM)
DV2 = DV2 - QSQRRT(MU * TEMP)
MF = X(4) * EXP(-DV2 / SDAT(4) / GE)
MFUEL = MF - X(4)
ENDIF
WRITE (10, 3640), WPLD
3640 FORMAT (1X, F6.0)
WFUEL = MFUEL * GE
WF = MF * GE
WRITE (10, 3650), DV2, WFCAL, WF
3650 FORMAT (1X, F8.1, F8.2, F9.0)
IORFLAG = 0

ELSE
TPAD = PHIO
PPAD = PHIO
RANGE2 = R0 / FTPNM * (X(2) - TPAD)
RANGE3 = R0 / FTPNM * (X(3) - PPAD)
A = QCOS(X(2) - TPAD) * QCOS(X(3) - PPAD)
RANGE = QACOS(A) * R0 / FTPNM
ALONGD = (X(2) - OMEGA * TIM) * 180. / PI
ALATD = X(3) * 180. / PI
GAMLD = GAML*180./PI
GAMID = GAMI*180./PI
LNS = LNS + 1
IF (BZ .EQ. 'Y') THEN
WRITE (5, 3660), TIM, H, V, GAMLD, T,
WEIGHT
3660 FORMAT (1X, F5.0, F9.0, F7.0, F6.2, 2F9.0)
ENDIF
WRITE (10, 3670), TIM, H, RANGE, ALONGD, ALATD, V,
VSP(4), GAMLD, GAMID, AOA, T, WEIGHT,
+ADOT, RANGE2, RANGE3, TTOW, HEADD
3670 FORMAT (1X, F5.0, F8.0, F7.0, 2F7.2, 2F7.0, 3F6.2, 2F9.0,
+F6.2, 2F7.0, F5.2, F6.2)
DO WHILE (OUTTIM .LE. TIM)
OUTTIM = OUTTIM + OUTINT
OUTTIM = QFLOAT(INT(OUTTIM ))
END DO
ENDIF
ENDIF
ENDIF
3700 FORMAT (/*
Weight Prior to Deorbit Burn :',F9.2,
+'<lbm>/')
3710 FORMAT ('Delta Velocity Required to Deorbit'/*
+'from the Holding Orbit (',F4.0,' X ',F4.0,)'/
+'the Initial Descent Orbit :',F9.2,'<ft/s>/')
3720 FORMAT ('Weight of Fuel Required to Deorbit:',F9.2,
+'<lbm>/')
3730 FORMAT ('Apocynthion -- ', F9.4,
+'<nm>/')
3740 FORMAT ('Pericynthion -- ', F9.4,
+'<nm>/')
3750 FORMAT ('Inclination -- ', F9.4,
+'<deg>/')
3760 FORMAT ('Longitude of the Ascending Node -- ', F9.4,
+'<deg>/')
3770 FORMAT ('Argument of Pericynthion -- ', F9.4,
+'<deg>/')
**Orbit Calculation Subroutine**

**Name**: ORBIT

**Author**: Chris Varner

**Date**: 18 March, 1986

***Purpose:*** This routine calculates the orbital parameters based on the position and velocity of the spacecraft with respect to the planet about which the orbit is to be determined.

**SUBROUTINE ORBIT (VSP, X, AOP, APG, ECC, AINCL, ALAN, PEG, SMJ)**

***Declare Variables***

IMPLICIT REAL *16 (A-Z)

INTEGER I, IFLAG, ITER, J, K, LNS, IORFLAG, IOUTFLAG

CHARACTER BZ*1, TYPZ*1

***Dimension Arrays***

DIMENSION C(3, 3), PSN(4), VEL(4), VSP(4), X(15)

COMMON/TOTA/AOA, ADOT, D1, D2, COTG, DT, D3, FTPNM, G0

COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD

COMMON/TOTC/IFLAG, ITER, D5, LNS, M(4), MU

COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, D55, PHI0, PI, BZ, R0

COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETAO, TIM

COMMON/TOTF/TRAJDAT(100, 201, TTOW, TYPZ, V, D6(4), WEIGHT

COMMON/TOTG/WPLD, D7(15)

C(1, 1) = QCOS(X(3)) * QCOS(X(2))
C(1, 2) = QCOS(X(3)) * QSIN(X(2))
C(1, 3) = QSIN(X(3))
C(2, 1) = -QSIN(X(2))
C(2, 2) = QCOS(X(2))
C(2, 3) = 0.
C(3, 1) = -QSIN(X(3)) * QCOS(X(2))
C(3, 2) = -QSIN(X(3)) * QSIN(X(2))
C(3, 3) = QCOS(X(3))

DO I=1, 3
    TEMP = 0
    DO J=1, 3
        TEMP = TEMP + VSP(J) * C(J, I)
    END DO

END
VEL(I) = TEMP
END DO
PSN(1) = X(1) * QCOS(X(3)) * QCOS(X(2))
PSN(2) = X(1) * QCOS(X(3)) * QSIN(X(2))
PSN(3) = X(1) * QSIN(X(3))
PSN(4) = QSQRT(PSN(1) ** 2. + PSN(2) ** 2. + PSN(3) ** 2.)
VEL(4) = QSQRT(VEL(1) ** 2. + VEL(2) ** 2. + VEL(3) ** 2.)
SMJ = 1. / (2. / PSN(4) - VEL(4) ** 2. / MU)
AI = PSN(1) * VEL(1) + PSN(2) * VEL(2) + PSN(3) * VEL(3)
IF (SMJ .LT. 0) THEN
  PRINT *, '***** Hyperbolic Orbit *****'
  PRINT *, 'Try again with a shorter MECO time'
  STOP
ENDIF
DENOM = 1. - PSN(4) / SMJ
NUMOR = AI / QSQRT(MU * SMJ)
ECC = QSQRT(DENOM ** 2. + NUMOR ** 2.)
ECA = QATAN2(NUMOR, DENOM)
MEA = ECA - ECC * QSIN(ECA)
AK = MU * DENOM / PSN(4)
SLR = SMJ * (1. - ECC ** 2.)
AJ = PSN(4) - SLR
AL = AI / PSN(4)
DO I=1, 3
  C(1, I) = (AK * PSN(I) - AI * VEL(I)) / MU / ECC
  C(2, I) = (AL * PSN(I) - AJ * VEL(I)) / ECC / QSQRT(MU * SLR)
END DO
C(3, 1) = C(1, 2) * C(2, 3) - C(1, 3) * C(2, 2)
C(3, 2) = C(1, 3) * C(2, 1) - C(1, 1) * C(2, 3)
C(3, 3) = C(1, 1) * C(2, 2) - C(1, 2) * C(2, 1)
IF (C(3, 3) .GT. 1.) C(3, 3) = 1.
AINCL = QACOS(C(3, 3))
ALAN = QATAN2(C(3, 1), -C(3, 2)) * 180. / PI
IF (ALAN .LT. 0) ALAN = 360 + ALAN
AOP = QATAN2(C(1, 3), C(2, 3)) * 180. / PI
IF (AOP .LT. 0) AOP = 360 + AOP
APG = (SMJ * (1 + ECC) - RO) / FTPNM
PEG = (SMJ * (1 - ECC) - RO) / FTPNM
RETURN
END

<table>
<thead>
<tr>
<th>Equations of Motion Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name   : EOM</td>
</tr>
<tr>
<td>Author : Chris Varner</td>
</tr>
<tr>
<td>Date   : 25 June, 1988</td>
</tr>
</tbody>
</table>
*** Purpose: This routine is used to evaluate the equations of motion. ***

SUBROUTINE EOM (GAMP, RKX, RKDX)

*** Declare Variables ***
IMPLICIT REAL *16 (A-Z)
INTEGER I, IFLAG, ITER, J, K, LNS, IORFLAG, IOUTFLAG
CHARACTER BZ*1, TYPZ*1

*** Dimension Arrays ***
DIMENSION RKX(15), RKDX(15)
COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
COMMON/TOTD/OMEGA, IORELAG, IOUTFLAG, PEG, PHIO, PI, BZ, R0
COMMON/TOTF/RUNITM, SDAT(5), T, TGT, THETAO, TIM
COMMON/TOTG/WPLD, x(15)

*** Preliminary Calculations ***
RKX(8) = RKX(4)
VSP(1) = RKX(5)
VSP(2) = RKX(1) * RKX(6) * QCOS(RKX(3))
VSP(3) = RKX(1) * RKX(7)
VSP(4) = QSQRT(VSP(1) ** 2. + VSP(2) ** 2. + VSP(3) ** 2.)
RDA = R0 * OMEGA * QCOS(PHI0)
V2 = VSP(2) - RDA
V = QSQRT(VSP(1) ** 2. + V2 ** 2. + VSP(3) ** 2.)
H = RKX(1) - R0
GAMI = QATAN(VSP(1) / QSQRT(VSP(2) ** 2. + VSP(3) ** 2.))
IF (V2 .EQ. 0. .AND. VSP(3) .EQ. 0.) THEN
    GAML = 90. * PI / 180.
ELSE
    GAML = QATAN(VSP(1) / QSQRT(V2 ** 2. + VSP(3) ** 2.))
ENDIF
NUMOR = RKX(7)
DENOM = RKX(6) - OMEGA
IF (DENOM .EQ. 0.) THEN
    IF (NUMOR .GE. 0.) THEN
        ANGLE = PI / 2.
    ELSE
        ANGLE = -PI / 2.
    ENDIF
ELSE
    ANGLE = QATAN2(NUMOR, DENOM)
ENDIF
IF (ANGLE .LE. PI / 2.) THEN HEAD = PI / 2. - ANGLE
IF (ANGLE .GT. PI / 2.) THEN HEAD = 5 * PI / 2. - ANGLE
CALL CONTROL (GAML, GAMP, V, GAMT, HEAD, HEADT)
CALL PROFILE (GAMP, RXX, LEVEL)
T = SDAT(3) * LEVEL
MDOT = -T / GE / SDAT(4)
T = T
HEADD = HEAD * 180. / PI
WEIGHT = RXX(4) * GE
TTOW = T / WEIGHT
DO I=1, 7
    RXX(I + 8) = RXX(I)
END DO

********************************************************************************

*** Equations of Motion for Spherical Coordinates ***
********************************************************************************

  RKDX(1) = RXX(5)
  RKDX(2) = RXX(6)
  RKDX(3) = RXX(7)
  RKDX(4) = MDOT
  TEMP1 = 0
  TEMP2 = 0
  TEMP3 = 0
  TEMP1 = T * QSIN(GAMT)
  TEMP1 = TEMP1 / RXX(4)
  TEMP2 = (T * QCOS(GAMT) * QSIN(HEADT)) / (RXX(4) * RXX(1) * QCOS(RXX(3)))
  TEMP3 = (T * QCOS(GAMT) * QCOS(HEADT)) / (RXX(4) * RXX(1))
  RKDX(5) = RXX(1) * RXX(6) ** 2. * QCOS(RXX(3)) ** 2. + RXX(1) * RXX(7) ** 2. + TEMP1
  RKDX(5) = RKDX(5) - MU / RXX(1) ** 2. + TEMP1
  RKDX(6) = 2. * (-RXX(5) * RXX(6) / RXX(1) + RXX(6) * RXX(7) * TAN(RXX(3)))
  RKDX(6) = RKDX(6) + TEMP2
  RKDX(7) = -2. * RXX(5) * RXX(7) / RXX(1)
  RKDX(7) = RKDX(7) - RXX(6) ** 2. * QSIN(RXX(3)) * QCOS(RXX(3)) + TEMP3
RETURN
END

Thrust Profile Subroutine

Name : THRUST
Author : Chris Varner
Date : 3 July, 1988

*** Purpose: The Thrust Profile subroutine provides the equations of motion with the level of thrust supplied by the engines.

70
SUBROUTINE PROFILE (GAMP, RKX, LEVEL)
*** Declare Variables ***
IMPLICIT REAL *16 (A-Z)
INTEGER IFLAG, ITER, LNS, IORFLAG, IOUTFLAG
CHARACTER BZ*1, TYPZ*1
*** Dimension Arrays ***
DIMENSION RKX(15)
COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
COMMON/TOTG/WPLD, x (15)
IF (TYPZ .EQ. 'D') THEN
  IF (TIM .GT. SDAT(5)) THEN
    IF (TIM .LE. SDAT(5) + 35.) THEN
      LEVEL = PRF1 + (1. - PRF1) / 35. * (TIM - SDAT(5))
    ELSE
      LEVEL = 1.
    ENDIF
  ELSE
    LEVEL = RKX(4) * G0 / SDAT(3)
    PRF1 = LEVEL
  ENDIF
ELSE
  LEVEL = 1.
  PRF1 = 1.
ENDIF
RETURN
END

Control Procedures Subroutine

Name : CONTROL
Author : Chris Varner
Date : 15 June, 1988

*** Purpose: This subroutine supplies the thrust control procedures required for the ascent and descent launches and landings.

SUBROUTINE CONTROL (GAML, GAMP, V, GAMT, HEAD, HEADT)
*** Declare Variables **
IMPLICIT REAL *16 (A-Z)
INTEGER IFLAG, ITER, LNS, IORFLAG, IOUTFLAG, IPFLAG
CHARACTER BZ*1, TYPZ*l
COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
COMMON/TOTB/GAMI, D1, D2, GE, H, HEAD0, D3, HEADD
COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, D4, VSP(4), WEIGHT
COMMON/TOTG/WPLD, X(15)

IF (V .LT. 30.) THEN
   GAMT = PI / 2.
   HEADT = HEAD0
   IPFLAG = 0
ELSE
   IF (IPFLAG .LT. 20) THEN
      IF (IPFLAG .LE. 10) THEN
         GAMT = GAMP - (90.*PI/180. - GAMP)/10.*QFLOAT(IPFLAG)
         HEADT = HEAD0
         IPFLAG = IPFLAG + 1
      ELSE
         GAMT = GAMP + (GAML - GAMP)/10.*QFLOAT(IPFLAG - 10)
         IPFLAG = IPFLAG + 1
      ENDIF
   ELSE
      GAMT = GAML
      IF (GAMT .LT. 0.) THEN GAMT = 0
      IF (GAML .GT. 80.* PI/180.) THEN
         HEADT = HEAD0
         HEAD = HEAD0
      ELSE
         HEADT = HEAD0
      ENDIF
   ENDIF
ENDIF
RETURN
END
APPENDIX C: Input/Output Examples

The following examples are simulations of the Apollo 15 descent to and ascent from the lunar surface. The spacecraft characteristics and the trajectory data are taken from the Apollo 15 Mission Report, document MSC-05161, written by the National Aeronautics and Space Administration’s Manned Spacecraft Center (Houston, Texas) in December of 1971. All of the pertinent data is condensed into Table A3.

Table A3: Apollo 15 Mission Characteristics

<table>
<thead>
<tr>
<th>Holding Orbit</th>
<th>50.1 &lt;nm&gt;</th>
<th>X</th>
<th>56.4 &lt;nm&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orbit Inclination</td>
<td>26.2°</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Landing Site Location</td>
<td>Hadley Rille-Apennine Mountains Region</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26.101° North Latitude</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.6528° East Longitude</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mass of Lunar Module at:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separation</td>
<td>35,718 &lt;lb&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lunar Landing</td>
<td>18,175 &lt;lb&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lunar Lift-off</td>
<td>10,915 &lt;lb&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Docking</td>
<td>5,826 &lt;lb&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engine Performance:</td>
<td>Thrust</td>
<td>Specific Impulse</td>
<td></td>
</tr>
<tr>
<td>Descent Engine</td>
<td>9,750 &lt;lbf&gt;</td>
<td>303 &lt;s&gt;</td>
<td></td>
</tr>
<tr>
<td>Ascent Engine</td>
<td>3,500 &lt;lbf&gt;</td>
<td>306 &lt;s&gt;</td>
<td></td>
</tr>
</tbody>
</table>

78 seconds elapsed between commencing Attitude Hold and Touchdown on the lunar surface. It is estimated that approximately 60 seconds were spent hovering.

The results of these simulations compare favorably with those of the actual Apollo 15 flight data shown in Table A3. The descent simulation predicts a weight prior to the deorbit burn of 35,642 lb; the actual value for Apollo 15 was 35,718 lb. The ascent simulation is of equivalent accuracy. Predicting a post apocynthion burn weight of 5,754 lb, the simulation is only 72 lb less than the actual weight recorded in the Mission Report.
LANDER Descent Simulation of the Apollo 15 Lunar Descent Module

The following inputs are supplied at the program prompts:

IS THIS TO BE AN ASCENT OR DESCENT SIMULATION ?
Answer: D

LANDING SITE LATITUDE (-90 TO +90)
Answer: 26.101

LANDING SITE LONGITUDE (0 TO 360)
Answer: 3.6527

INERT WEIGHT <LB>
Answer: 18175

PROPELLANT WEIGHT <LB>
Answer: 17543

THRUST <LBF>
Answer: 9750

SPECIFIC IMPULSE <S>
Answer: 303

HOVER TIME <S>
Answer: 60

PAYLOAD WEIGHT <LB>
Answer: 0

TIME TO MAIN ENGINE CUT-OFF (MECO)? <S>
Answer: 440

HOLDING ORBIT PERICYCTHON <NM>
Answer: 50

HOLDING ORBIT APOCYCTHON <NM>
Answer: 50

FLIGHT PATH ANGLE AT PITCH-OVER ?
Answer: 70

HOLDING ORBIT INCLINATION ? (0 TO 360)
Answer: 26.2
DO YOU WISH TO SEE THE TRAJECTORY OF EACH ITERATION?
Answer: N

OUTPUT:

Weight Prior to Deorbit Burn : 35642.00 <lbm>

Delta Velocity Required to Deorbit from the Holding Orbit ( 48. X 50. ) to the Initial Descent Orbit : 34.90 <ft/s>

Weight of Fuel Required to Deorbit: 127.47 <lbm>

Initial Descent Orbit:

| Apocynthon  | 48.5000 <nm> |
| Pericynthion | 24.7000 <nm> |
| Inclination  | 25.8400 <deg> |
| Longitude of the Ascending Node | 274.2300 <deg> |
| Argument of Pericynthion | 75.1600 <deg> |
| Eccentricity | 0.0122 <nd> |

<table>
<thead>
<tr>
<th>Time &lt;s&gt;</th>
<th>Altitude &lt;ft&gt;</th>
<th>Range &lt;nm&gt;</th>
<th>Velocity &lt;ft/s&gt;</th>
<th>Gamma &lt;deg&gt;</th>
<th>Heading &lt;deg&gt;</th>
<th>Thrust &lt;lbf&gt;</th>
<th>Weight &lt;lbm&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>149917.</td>
<td>258.</td>
<td>5465.</td>
<td>0.00</td>
<td>87.63</td>
<td>9750.</td>
<td>35450.</td>
</tr>
<tr>
<td>5.</td>
<td>149918.</td>
<td>253.</td>
<td>5421.</td>
<td>0.00</td>
<td>87.63</td>
<td>9750.</td>
<td>35289.</td>
</tr>
<tr>
<td>10.</td>
<td>149918.</td>
<td>248.</td>
<td>5376.</td>
<td>0.00</td>
<td>87.63</td>
<td>9750.</td>
<td>35128.</td>
</tr>
<tr>
<td>15.</td>
<td>149914.</td>
<td>243.</td>
<td>5332.</td>
<td>0.01</td>
<td>87.63</td>
<td>9750.</td>
<td>34967.</td>
</tr>
<tr>
<td>20.</td>
<td>149905.</td>
<td>239.</td>
<td>5287.</td>
<td>0.03</td>
<td>87.63</td>
<td>9750.</td>
<td>34806.</td>
</tr>
<tr>
<td>25.</td>
<td>149889.</td>
<td>234.</td>
<td>5242.</td>
<td>0.04</td>
<td>87.63</td>
<td>9750.</td>
<td>34645.</td>
</tr>
<tr>
<td>30.</td>
<td>149864.</td>
<td>229.</td>
<td>5197.</td>
<td>0.07</td>
<td>87.63</td>
<td>9750.</td>
<td>34485.</td>
</tr>
<tr>
<td>35.</td>
<td>149827.</td>
<td>225.</td>
<td>5151.</td>
<td>0.10</td>
<td>87.63</td>
<td>9750.</td>
<td>34324.</td>
</tr>
<tr>
<td>40.</td>
<td>149777.</td>
<td>220.</td>
<td>5106.</td>
<td>0.13</td>
<td>87.63</td>
<td>9750.</td>
<td>34163.</td>
</tr>
<tr>
<td>45.</td>
<td>149712.</td>
<td>216.</td>
<td>5060.</td>
<td>0.16</td>
<td>87.63</td>
<td>9750.</td>
<td>34002.</td>
</tr>
<tr>
<td>50.</td>
<td>149631.</td>
<td>211.</td>
<td>5014.</td>
<td>0.21</td>
<td>87.63</td>
<td>9750.</td>
<td>33841.</td>
</tr>
<tr>
<td>55.</td>
<td>149531.</td>
<td>207.</td>
<td>4968.</td>
<td>0.25</td>
<td>87.63</td>
<td>9750.</td>
<td>33680.</td>
</tr>
<tr>
<td>60.</td>
<td>149411.</td>
<td>203.</td>
<td>4921.</td>
<td>0.31</td>
<td>87.63</td>
<td>9750.</td>
<td>33519.</td>
</tr>
<tr>
<td>65.</td>
<td>149268.</td>
<td>198.</td>
<td>4875.</td>
<td>0.36</td>
<td>87.63</td>
<td>9750.</td>
<td>33358.</td>
</tr>
<tr>
<td>70.</td>
<td>149102.</td>
<td>194.</td>
<td>4828.</td>
<td>0.42</td>
<td>87.63</td>
<td>9750.</td>
<td>33197.</td>
</tr>
<tr>
<td>75.</td>
<td>148911.</td>
<td>190.</td>
<td>4781.</td>
<td>0.49</td>
<td>87.63</td>
<td>9750.</td>
<td>33037.</td>
</tr>
<tr>
<td>80.</td>
<td>148692.</td>
<td>186.</td>
<td>4733.</td>
<td>0.56</td>
<td>87.63</td>
<td>9750.</td>
<td>32876.</td>
</tr>
<tr>
<td>85.</td>
<td>148445.</td>
<td>181.</td>
<td>4686.</td>
<td>0.64</td>
<td>87.63</td>
<td>9750.</td>
<td>32715.</td>
</tr>
<tr>
<td>90.</td>
<td>148167.</td>
<td>177.</td>
<td>4638.</td>
<td>0.72</td>
<td>87.63</td>
<td>9750.</td>
<td>32554.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>95</td>
<td>147857</td>
<td>173</td>
<td>4591</td>
<td>0.81</td>
<td>87.63</td>
<td>9750</td>
<td>32393</td>
</tr>
<tr>
<td>100.</td>
<td>147515</td>
<td>169</td>
<td>4542</td>
<td>0.91</td>
<td>87.63</td>
<td>9750</td>
<td>32232</td>
</tr>
<tr>
<td>105.</td>
<td>147137</td>
<td>165</td>
<td>4494</td>
<td>0.10</td>
<td>87.63</td>
<td>9750</td>
<td>32071</td>
</tr>
<tr>
<td>110.</td>
<td>146723</td>
<td>161</td>
<td>4446</td>
<td>1.11</td>
<td>87.63</td>
<td>9750</td>
<td>31910</td>
</tr>
<tr>
<td>115.</td>
<td>146272</td>
<td>157</td>
<td>4397</td>
<td>1.23</td>
<td>87.63</td>
<td>9750</td>
<td>31749</td>
</tr>
<tr>
<td>120.</td>
<td>145781</td>
<td>153</td>
<td>4348</td>
<td>1.34</td>
<td>87.63</td>
<td>9750</td>
<td>31588</td>
</tr>
<tr>
<td>125.</td>
<td>145251</td>
<td>149</td>
<td>4299</td>
<td>1.47</td>
<td>87.63</td>
<td>9750</td>
<td>31428</td>
</tr>
<tr>
<td>130.</td>
<td>144679</td>
<td>146</td>
<td>4250</td>
<td>1.60</td>
<td>87.63</td>
<td>9750</td>
<td>31267</td>
</tr>
<tr>
<td>135.</td>
<td>144064</td>
<td>142</td>
<td>4200</td>
<td>1.74</td>
<td>87.63</td>
<td>9750</td>
<td>31106</td>
</tr>
<tr>
<td>140.</td>
<td>143405</td>
<td>138</td>
<td>4151</td>
<td>1.88</td>
<td>87.63</td>
<td>9750</td>
<td>30945</td>
</tr>
<tr>
<td>145.</td>
<td>142702</td>
<td>134</td>
<td>4101</td>
<td>2.03</td>
<td>87.63</td>
<td>9750</td>
<td>30784</td>
</tr>
<tr>
<td>150.</td>
<td>141952</td>
<td>131</td>
<td>4051</td>
<td>2.19</td>
<td>87.63</td>
<td>9750</td>
<td>30623</td>
</tr>
<tr>
<td>155.</td>
<td>141155</td>
<td>127</td>
<td>4001</td>
<td>2.35</td>
<td>87.63</td>
<td>9750</td>
<td>30462</td>
</tr>
<tr>
<td>160.</td>
<td>140310</td>
<td>124</td>
<td>3950</td>
<td>2.52</td>
<td>87.63</td>
<td>9750</td>
<td>30301</td>
</tr>
<tr>
<td>165.</td>
<td>139416</td>
<td>120</td>
<td>3899</td>
<td>2.70</td>
<td>87.63</td>
<td>9750</td>
<td>30140</td>
</tr>
<tr>
<td>170.</td>
<td>138473</td>
<td>117</td>
<td>3849</td>
<td>2.89</td>
<td>87.63</td>
<td>9750</td>
<td>29980</td>
</tr>
<tr>
<td>175.</td>
<td>137478</td>
<td>113</td>
<td>3798</td>
<td>3.08</td>
<td>87.63</td>
<td>9750</td>
<td>29819</td>
</tr>
<tr>
<td>180.</td>
<td>136432</td>
<td>110</td>
<td>3746</td>
<td>3.28</td>
<td>87.63</td>
<td>9750</td>
<td>29658</td>
</tr>
<tr>
<td>185.</td>
<td>135334</td>
<td>107</td>
<td>3695</td>
<td>3.49</td>
<td>87.63</td>
<td>9750</td>
<td>29497</td>
</tr>
<tr>
<td>190.</td>
<td>134184</td>
<td>103</td>
<td>3643</td>
<td>3.71</td>
<td>87.63</td>
<td>9750</td>
<td>29336</td>
</tr>
<tr>
<td>195.</td>
<td>132980</td>
<td>100</td>
<td>3591</td>
<td>3.93</td>
<td>87.63</td>
<td>9750</td>
<td>29175</td>
</tr>
<tr>
<td>200.</td>
<td>131722</td>
<td>97</td>
<td>3539</td>
<td>4.16</td>
<td>87.63</td>
<td>9750</td>
<td>29014</td>
</tr>
<tr>
<td>205.</td>
<td>130409</td>
<td>94</td>
<td>3487</td>
<td>4.41</td>
<td>87.63</td>
<td>9750</td>
<td>28853</td>
</tr>
<tr>
<td>210.</td>
<td>129043</td>
<td>91</td>
<td>3435</td>
<td>4.66</td>
<td>87.63</td>
<td>9750</td>
<td>28692</td>
</tr>
<tr>
<td>215.</td>
<td>127621</td>
<td>88</td>
<td>3382</td>
<td>4.92</td>
<td>87.63</td>
<td>9750</td>
<td>28532</td>
</tr>
<tr>
<td>220.</td>
<td>126143</td>
<td>85</td>
<td>3329</td>
<td>5.19</td>
<td>87.63</td>
<td>9750</td>
<td>28371</td>
</tr>
<tr>
<td>225.</td>
<td>124611</td>
<td>82</td>
<td>3276</td>
<td>5.47</td>
<td>87.63</td>
<td>9750</td>
<td>28210</td>
</tr>
<tr>
<td>230.</td>
<td>123023</td>
<td>79</td>
<td>3223</td>
<td>5.75</td>
<td>87.63</td>
<td>9750</td>
<td>28049</td>
</tr>
<tr>
<td>235.</td>
<td>121379</td>
<td>76</td>
<td>3170</td>
<td>6.05</td>
<td>87.63</td>
<td>9750</td>
<td>27888</td>
</tr>
<tr>
<td>240.</td>
<td>119680</td>
<td>73</td>
<td>3116</td>
<td>6.36</td>
<td>87.63</td>
<td>9750</td>
<td>27727</td>
</tr>
<tr>
<td>245.</td>
<td>117926</td>
<td>71</td>
<td>3062</td>
<td>6.68</td>
<td>87.63</td>
<td>9750</td>
<td>27566</td>
</tr>
<tr>
<td>250.</td>
<td>116117</td>
<td>68</td>
<td>3008</td>
<td>7.01</td>
<td>87.63</td>
<td>9750</td>
<td>27405</td>
</tr>
<tr>
<td>255.</td>
<td>114252</td>
<td>65</td>
<td>2954</td>
<td>7.36</td>
<td>87.63</td>
<td>9750</td>
<td>27244</td>
</tr>
<tr>
<td>260.</td>
<td>112334</td>
<td>63</td>
<td>2900</td>
<td>7.71</td>
<td>87.63</td>
<td>9750</td>
<td>27084</td>
</tr>
<tr>
<td>265.</td>
<td>110362</td>
<td>60</td>
<td>2845</td>
<td>8.08</td>
<td>87.63</td>
<td>9750</td>
<td>26923</td>
</tr>
<tr>
<td>270.</td>
<td>108337</td>
<td>58</td>
<td>2791</td>
<td>8.45</td>
<td>87.63</td>
<td>9750</td>
<td>26762</td>
</tr>
<tr>
<td>275.</td>
<td>106260</td>
<td>55</td>
<td>2736</td>
<td>8.85</td>
<td>87.63</td>
<td>9750</td>
<td>26601</td>
</tr>
<tr>
<td>280.</td>
<td>104130</td>
<td>53</td>
<td>2681</td>
<td>9.25</td>
<td>87.63</td>
<td>9750</td>
<td>26440</td>
</tr>
<tr>
<td>285.</td>
<td>101951</td>
<td>50</td>
<td>2625</td>
<td>9.67</td>
<td>87.63</td>
<td>9750</td>
<td>26279</td>
</tr>
<tr>
<td>290.</td>
<td>99722</td>
<td>48</td>
<td>2570</td>
<td>10.10</td>
<td>87.63</td>
<td>9750</td>
<td>26118</td>
</tr>
<tr>
<td>295.</td>
<td>97444</td>
<td>46</td>
<td>2514</td>
<td>10.55</td>
<td>87.63</td>
<td>9750</td>
<td>25957</td>
</tr>
<tr>
<td>300.</td>
<td>95119</td>
<td>44</td>
<td>2459</td>
<td>11.01</td>
<td>87.63</td>
<td>9750</td>
<td>25796</td>
</tr>
<tr>
<td>305.</td>
<td>92749</td>
<td>41</td>
<td>2403</td>
<td>11.49</td>
<td>87.63</td>
<td>9750</td>
<td>25636</td>
</tr>
<tr>
<td>310.</td>
<td>90335</td>
<td>39</td>
<td>2347</td>
<td>11.98</td>
<td>87.63</td>
<td>9750</td>
<td>25475</td>
</tr>
<tr>
<td>315.</td>
<td>88787</td>
<td>37</td>
<td>2290</td>
<td>12.49</td>
<td>87.63</td>
<td>9750</td>
<td>25314</td>
</tr>
<tr>
<td>320.</td>
<td>85381</td>
<td>35</td>
<td>2234</td>
<td>13.02</td>
<td>87.63</td>
<td>9750</td>
<td>25153</td>
</tr>
<tr>
<td>325.</td>
<td>82845</td>
<td>33</td>
<td>2177</td>
<td>13.57</td>
<td>87.63</td>
<td>9750</td>
<td>24992</td>
</tr>
</tbody>
</table>

76
335.  77667.  30.  2064.  14.72  87.63  9750.  24670.
340.  75030.  28.  2007.  15.33  87.63  9750.  24509.
345.  72363.  26.  1950.  15.96  87.63  9750.  24348.
350.  69671.  25.  1892.  16.61  87.63  9750.  24187.
355.  66956.  23.  1835.  17.28  87.63  9750.  24027.
360.  64221.  21.  1777.  17.98  87.63  9750.  23866.
365.  61469.  20.  1720.  18.71  87.63  9750.  23705.
370.  58705.  18.  1662.  19.47  87.63  9750.  23544.
375.  55933.  17.  1604.  20.25  87.63  9750.  23383.
380.  53156.  16.  1546.  21.06  87.63  9750.  23222.
390.  47606.  13.  1429.  22.80  87.63  9750.  22900.
400.  42093.  11.  1313.  24.67  87.63  9750.  22579.
405.  39365.  10.  1254.  25.68  87.63  9750.  22418.
410.  36662.  9.   1195.  26.72  87.63  9750.  22257.
415.  33991.  8.   1137.  27.82  87.63  9750.  22096.
420.  31359.  7.   1078.  28.96  87.63  9750.  21935.
425.  28773.  6.   1019.  30.17  87.63  9750.  21774.
430.  26239.  5.   961.  31.43  87.63  9750.  21613.
435.  23765.  5.   902.  32.77  87.63  9750.  21452.
440.  21360.  4.   843.  34.18  87.63  9750.  21291.
445.  19033.  3.   784.  35.67  87.63  9750.  21131.
450.  16791.  3.   725.  37.25  87.63  9750.  20970.
455.  14645.  2.   667.  38.94  87.63  9750.  20809.
460.  12604.  2.   608.  40.74  87.63  9750.  20648.
465.  10680.  2.   549.  42.67  87.63  9750.  20487.
470.   8884.  1.   491.  44.76  87.63  9750.  20326.
475.   7227.  1.   433.  47.04  87.63  9750.  20165.
480.   5723.  1.   374.  49.53  87.63  9750.  20004.
485.   4384.  0.   316.  52.30  87.63  9750.  19843.
490.   3226.  0.   258.  55.42  87.63  9750.  19683.
495.   2263.  0.   201.  59.02  87.63  9750.  19522.
500.   1505.  0.   147.  63.28  87.63   8800.  19369.
505.   941.  0.   102.  68.34  87.63   7851.  19231.
510.   549.  0.    65.  74.26  87.63   6901.  19109.
515.   303.  0.    37.  83.35  87.63   5951.  19003.
520.   169.  0.    18.  90.00  87.63   5002.  18913.
525.   114.  0.     6.  89.99  87.63   4052.  18838.
530.    99.  0.     2.  89.97  87.63   3102.  18779.
535.    90.  0.     2.  89.97  87.63   3094.  18728.
540.    82.  0.     2.  89.97  87.63   3085.  18677.
545.    74.  0.     2.  89.97  87.63   3077.  18626.
550.    66.  0.     2.  89.98  87.63   3069.  18576.
555.    57.  0.     2.  89.98  87.63   3060.  18525.
560.    49.  0.     2.  89.98  87.63   3052.  18475.
Ideal Performance Delta Velocity is: 6525.00 <ft/s>

************ SIMULATION COMPLETE *************
LANDER Ascent Simulation of the Apollo 15 Lunar Ascent Module

The following inputs are supplied at the program prompts:

IS THIS TO BE AN ASCENT OR DESCENT SIMULATION?
Answer: A

LANDING SITE LATITUDE (-90 TO +90)
Answer: 26.1011

LANDING SITE LONGITUDE (0 TO 360)
Answer: 3.6527

INERT WEIGHT <LB>
Answer: 5326

PROPELLANT WEIGHT <LB>
Answer: 5589

THRUST <LBF>
Answer: 3500

SPECIFIC IMPULSE <S>
Answer: 306

HOVER TIME <S>
Answer: (Not Applicable)

PAYLOAD WEIGHT <LB>
Answer: 0

TIME TO MAIN ENGINE CUT-OFF (MECO)? <S>
Answer: 460

HOLDING ORBIT PERICYNTHON <NM>
Answer: 50

1 500 pounds is transferred to the propellant from the inert to prevent the simulation vehicle from running out of propellant during the simulation. This has no effect upon the results because the only propellant used is that necessary to achieve the requested orbit; the remaining fuel is assumed to be payload or otherwise inert. If the propellant were to be exhausted during the ascent then the iteration technique may become divergent, and fail to reach a satisfactory solution.
HOLDING ORBIT APOCYNTHION <NM>
Answer: 50

FLIGHT PATH ANGLE AT PITCH-OVER?
Answer: 85°

HOLDING ORBIT INCLINATION? (0 TO 360)
Answer: 26.2

DO YOU WISH TO SEE THE TRAJECTORY OF EACH ITERATION?
Answer: N

OUTPUT:

<table>
<thead>
<tr>
<th>Time &lt;s&gt;</th>
<th>Altitude &lt;ft&gt;</th>
<th>Range &lt;nm&gt;</th>
<th>Velocity &lt;ft/s&gt;</th>
<th>Gamma &lt;deg&gt;</th>
<th>Heading &lt;deg&gt;</th>
<th>Thrust &lt;lbf&gt;</th>
<th>Weight &lt;lbm&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>2.</td>
<td>90.00</td>
<td>87.63</td>
<td>3500.</td>
<td>10915.</td>
</tr>
<tr>
<td>5.</td>
<td>71.</td>
<td>0.</td>
<td>27.</td>
<td>90.00</td>
<td>87.63</td>
<td>3500.</td>
<td>10858.</td>
</tr>
<tr>
<td>10.</td>
<td>267.</td>
<td>0.</td>
<td>52.</td>
<td>84.34</td>
<td>87.63</td>
<td>3500.</td>
<td>10801.</td>
</tr>
<tr>
<td>15.</td>
<td>590.</td>
<td>0.</td>
<td>78.</td>
<td>81.47</td>
<td>87.63</td>
<td>3500.</td>
<td>10743.</td>
</tr>
<tr>
<td>20.</td>
<td>1038.</td>
<td>0.</td>
<td>104.</td>
<td>78.58</td>
<td>87.63</td>
<td>3500.</td>
<td>10686.</td>
</tr>
<tr>
<td>25.</td>
<td>1612.</td>
<td>0.</td>
<td>131.</td>
<td>75.71</td>
<td>87.63</td>
<td>3500.</td>
<td>10629.</td>
</tr>
<tr>
<td>30.</td>
<td>2310.</td>
<td>0.</td>
<td>159.</td>
<td>72.87</td>
<td>87.63</td>
<td>3500.</td>
<td>10572.</td>
</tr>
<tr>
<td>35.</td>
<td>3129.</td>
<td>0.</td>
<td>187.</td>
<td>70.08</td>
<td>87.63</td>
<td>3500.</td>
<td>10515.</td>
</tr>
<tr>
<td>40.</td>
<td>4067.</td>
<td>0.</td>
<td>216.</td>
<td>67.35</td>
<td>87.63</td>
<td>3500.</td>
<td>10457.</td>
</tr>
<tr>
<td>45.</td>
<td>5120.</td>
<td>0.</td>
<td>246.</td>
<td>64.68</td>
<td>87.63</td>
<td>3500.</td>
<td>10400.</td>
</tr>
<tr>
<td>50.</td>
<td>6286.</td>
<td>0.</td>
<td>276.</td>
<td>62.07</td>
<td>87.63</td>
<td>3500.</td>
<td>10343.</td>
</tr>
<tr>
<td>55.</td>
<td>7559.</td>
<td>1.</td>
<td>308.</td>
<td>59.55</td>
<td>87.63</td>
<td>3500.</td>
<td>10286.</td>
</tr>
<tr>
<td>60.</td>
<td>8936.</td>
<td>1.</td>
<td>340.</td>
<td>57.09</td>
<td>87.63</td>
<td>3500.</td>
<td>10229.</td>
</tr>
<tr>
<td>65.</td>
<td>10412.</td>
<td>1.</td>
<td>373.</td>
<td>54.72</td>
<td>87.63</td>
<td>3500.</td>
<td>10172.</td>
</tr>
<tr>
<td>70.</td>
<td>11981.</td>
<td>1.</td>
<td>407.</td>
<td>52.42</td>
<td>87.63</td>
<td>3500.</td>
<td>10114.</td>
</tr>
<tr>
<td>75.</td>
<td>13639.</td>
<td>1.</td>
<td>443.</td>
<td>50.21</td>
<td>87.63</td>
<td>3500.</td>
<td>10057.</td>
</tr>
<tr>
<td>80.</td>
<td>15380.</td>
<td>2.</td>
<td>479.</td>
<td>48.08</td>
<td>87.63</td>
<td>3500.</td>
<td>10000.</td>
</tr>
<tr>
<td>85.</td>
<td>17199.</td>
<td>2.</td>
<td>516.</td>
<td>46.02</td>
<td>87.63</td>
<td>3500.</td>
<td>9943.</td>
</tr>
</tbody>
</table>

2 In this case, the recommended initial guess of 70° is bad. Due to the slow ascent, the spacecraft is very sensitive to the pitch-over angle. Any choice of angle less than 80° will result in the vehicle pitching-over too rapidly -- flying back into the ground. The result is that the program falls into what is often referred to as "Bang-Bang" instability, where it continuously oscillates between two, three, or four pitch-over angles. This endless loop can be spotted by noting that the periodic screen output is cycling between the same set of apocynthion and pericynthion. The problem can be determined by printing the trajectory of each iteration. Once the problem has been analyzed, then modifications of the input data can be implemented. In this case the solution was to raise the pitch-over angle.
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>19089</td>
<td>2</td>
<td>554</td>
<td>44.04</td>
<td>87.63</td>
<td>3500</td>
<td>9886</td>
</tr>
<tr>
<td>95</td>
<td>21047</td>
<td>3</td>
<td>593</td>
<td>42.14</td>
<td>87.63</td>
<td>3500</td>
<td>9828</td>
</tr>
<tr>
<td>100</td>
<td>23066</td>
<td>3</td>
<td>633</td>
<td>40.31</td>
<td>87.63</td>
<td>3500</td>
<td>9771</td>
</tr>
<tr>
<td>105</td>
<td>25140</td>
<td>4</td>
<td>674</td>
<td>38.55</td>
<td>87.63</td>
<td>3500</td>
<td>9714</td>
</tr>
<tr>
<td>110</td>
<td>27264</td>
<td>4</td>
<td>716</td>
<td>36.86</td>
<td>87.63</td>
<td>3500</td>
<td>9657</td>
</tr>
<tr>
<td>115</td>
<td>29434</td>
<td>5</td>
<td>759</td>
<td>35.24</td>
<td>87.63</td>
<td>3500</td>
<td>9600</td>
</tr>
<tr>
<td>120</td>
<td>31642</td>
<td>5</td>
<td>803</td>
<td>33.69</td>
<td>87.63</td>
<td>3500</td>
<td>9542</td>
</tr>
<tr>
<td>125</td>
<td>33885</td>
<td>6</td>
<td>848</td>
<td>32.19</td>
<td>87.63</td>
<td>3500</td>
<td>9485</td>
</tr>
<tr>
<td>130</td>
<td>36157</td>
<td>6</td>
<td>894</td>
<td>30.76</td>
<td>87.63</td>
<td>3500</td>
<td>9428</td>
</tr>
<tr>
<td>135</td>
<td>38453</td>
<td>7</td>
<td>940</td>
<td>29.38</td>
<td>87.63</td>
<td>3500</td>
<td>9371</td>
</tr>
<tr>
<td>140</td>
<td>40768</td>
<td>8</td>
<td>988</td>
<td>28.06</td>
<td>87.63</td>
<td>3500</td>
<td>9314</td>
</tr>
<tr>
<td>145</td>
<td>43098</td>
<td>9</td>
<td>1036</td>
<td>26.79</td>
<td>87.63</td>
<td>3500</td>
<td>9257</td>
</tr>
<tr>
<td>150</td>
<td>45439</td>
<td>10</td>
<td>1086</td>
<td>25.57</td>
<td>87.63</td>
<td>3500</td>
<td>9199</td>
</tr>
<tr>
<td>155</td>
<td>47784</td>
<td>11</td>
<td>1136</td>
<td>24.40</td>
<td>87.63</td>
<td>3500</td>
<td>9142</td>
</tr>
<tr>
<td>160</td>
<td>50131</td>
<td>12</td>
<td>1187</td>
<td>23.28</td>
<td>87.63</td>
<td>3500</td>
<td>9085</td>
</tr>
<tr>
<td>165</td>
<td>52475</td>
<td>13</td>
<td>1239</td>
<td>22.20</td>
<td>87.63</td>
<td>3500</td>
<td>9028</td>
</tr>
<tr>
<td>170</td>
<td>54813</td>
<td>14</td>
<td>1292</td>
<td>21.16</td>
<td>87.63</td>
<td>3500</td>
<td>8971</td>
</tr>
<tr>
<td>175</td>
<td>57140</td>
<td>15</td>
<td>1346</td>
<td>20.17</td>
<td>87.63</td>
<td>3500</td>
<td>8913</td>
</tr>
<tr>
<td>180</td>
<td>59452</td>
<td>16</td>
<td>1401</td>
<td>19.21</td>
<td>87.63</td>
<td>3500</td>
<td>8856</td>
</tr>
<tr>
<td>185</td>
<td>61747</td>
<td>17</td>
<td>1456</td>
<td>18.29</td>
<td>87.63</td>
<td>3500</td>
<td>8799</td>
</tr>
<tr>
<td>190</td>
<td>64021</td>
<td>18</td>
<td>1512</td>
<td>17.41</td>
<td>87.63</td>
<td>3500</td>
<td>8742</td>
</tr>
<tr>
<td>195</td>
<td>66270</td>
<td>20</td>
<td>1569</td>
<td>16.56</td>
<td>87.63</td>
<td>3500</td>
<td>8685</td>
</tr>
<tr>
<td>200</td>
<td>68492</td>
<td>21</td>
<td>1627</td>
<td>15.74</td>
<td>87.63</td>
<td>3500</td>
<td>8627</td>
</tr>
<tr>
<td>205</td>
<td>70683</td>
<td>23</td>
<td>1686</td>
<td>14.96</td>
<td>87.63</td>
<td>3500</td>
<td>8570</td>
</tr>
<tr>
<td>210</td>
<td>72842</td>
<td>24</td>
<td>1745</td>
<td>14.21</td>
<td>87.63</td>
<td>3500</td>
<td>8513</td>
</tr>
<tr>
<td>215</td>
<td>74964</td>
<td>26</td>
<td>1805</td>
<td>13.48</td>
<td>87.63</td>
<td>3500</td>
<td>8456</td>
</tr>
<tr>
<td>220</td>
<td>77048</td>
<td>27</td>
<td>1866</td>
<td>12.78</td>
<td>87.63</td>
<td>3500</td>
<td>8399</td>
</tr>
<tr>
<td>225</td>
<td>79092</td>
<td>29</td>
<td>1927</td>
<td>12.11</td>
<td>87.63</td>
<td>3500</td>
<td>8341</td>
</tr>
<tr>
<td>230</td>
<td>81093</td>
<td>31</td>
<td>1990</td>
<td>11.47</td>
<td>87.63</td>
<td>3500</td>
<td>8284</td>
</tr>
<tr>
<td>235</td>
<td>83049</td>
<td>32</td>
<td>2053</td>
<td>10.85</td>
<td>87.63</td>
<td>3500</td>
<td>8227</td>
</tr>
<tr>
<td>240</td>
<td>84958</td>
<td>34</td>
<td>2117</td>
<td>10.26</td>
<td>87.63</td>
<td>3500</td>
<td>8170</td>
</tr>
<tr>
<td>245</td>
<td>86819</td>
<td>36</td>
<td>2182</td>
<td>9.69</td>
<td>87.63</td>
<td>3500</td>
<td>8113</td>
</tr>
<tr>
<td>250</td>
<td>88629</td>
<td>38</td>
<td>2247</td>
<td>9.14</td>
<td>87.63</td>
<td>3500</td>
<td>8056</td>
</tr>
<tr>
<td>255</td>
<td>90388</td>
<td>40</td>
<td>2313</td>
<td>8.61</td>
<td>87.63</td>
<td>3500</td>
<td>7999</td>
</tr>
<tr>
<td>260</td>
<td>92093</td>
<td>42</td>
<td>2380</td>
<td>8.11</td>
<td>87.63</td>
<td>3500</td>
<td>7941</td>
</tr>
<tr>
<td>265</td>
<td>93744</td>
<td>44</td>
<td>2447</td>
<td>7.62</td>
<td>87.63</td>
<td>3500</td>
<td>7884</td>
</tr>
<tr>
<td>270</td>
<td>95340</td>
<td>47</td>
<td>2516</td>
<td>7.16</td>
<td>87.63</td>
<td>3500</td>
<td>7827</td>
</tr>
<tr>
<td>275</td>
<td>96879</td>
<td>49</td>
<td>2585</td>
<td>6.71</td>
<td>87.63</td>
<td>3500</td>
<td>7770</td>
</tr>
<tr>
<td>280</td>
<td>98360</td>
<td>51</td>
<td>2655</td>
<td>6.28</td>
<td>87.63</td>
<td>3500</td>
<td>7712</td>
</tr>
<tr>
<td>285</td>
<td>99784</td>
<td>54</td>
<td>2725</td>
<td>5.87</td>
<td>87.63</td>
<td>3500</td>
<td>7655</td>
</tr>
<tr>
<td>290</td>
<td>101148</td>
<td>56</td>
<td>2796</td>
<td>5.48</td>
<td>87.63</td>
<td>3500</td>
<td>7598</td>
</tr>
<tr>
<td>295</td>
<td>102454</td>
<td>59</td>
<td>2868</td>
<td>5.10</td>
<td>87.63</td>
<td>3500</td>
<td>7541</td>
</tr>
<tr>
<td>300</td>
<td>103700</td>
<td>61</td>
<td>2941</td>
<td>4.74</td>
<td>87.63</td>
<td>3500</td>
<td>7484</td>
</tr>
<tr>
<td>305</td>
<td>104887</td>
<td>64</td>
<td>3014</td>
<td>4.40</td>
<td>87.63</td>
<td>3500</td>
<td>7426</td>
</tr>
<tr>
<td>310</td>
<td>106013</td>
<td>67</td>
<td>3089</td>
<td>4.07</td>
<td>87.63</td>
<td>3500</td>
<td>7369</td>
</tr>
<tr>
<td>315</td>
<td>107080</td>
<td>70</td>
<td>3163</td>
<td>3.76</td>
<td>87.63</td>
<td>3500</td>
<td>7312</td>
</tr>
<tr>
<td>320</td>
<td>108088</td>
<td>72</td>
<td>3239</td>
<td>3.46</td>
<td>87.63</td>
<td>3500</td>
<td>7255</td>
</tr>
</tbody>
</table>
Ideal Performance Delta Velocity is: \(6254.70\) ft/s

Boost Orbit:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Apocynion</td>
<td>--</td>
<td>36.600 &lt;nm&gt;</td>
<td></td>
</tr>
<tr>
<td>Pericynion</td>
<td>--</td>
<td>19.300 &lt;nm&gt;</td>
<td></td>
</tr>
<tr>
<td>Inclination</td>
<td>--</td>
<td>26.250 &lt;deg&gt;</td>
<td></td>
</tr>
<tr>
<td>Longitude of the Ascending Node</td>
<td>--</td>
<td>283.000 &lt;deg&gt;</td>
<td></td>
</tr>
<tr>
<td>Argument of Pericynion</td>
<td>--</td>
<td>91.260 &lt;deg&gt;</td>
<td></td>
</tr>
<tr>
<td>Eccentricity</td>
<td>--</td>
<td>0.0089 &lt;nd&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Velocity Required at Apocynion to Achieve the Holding Orbit ( 37. X 50. ) : \(42.50\) ft/s

Fuel Required for the Apocynion Burn : \(24.94\) lbm

Weight After Apocynion Burn : \(5754.00\) lbm

Weight of the Payload Placed in Orbit : \(428.00\) lbm

************** SIMULATION COMPLETE ***************