Lambert's Problem

This document describes four MATLAB scripts that demonstrate how to solve the Earth orbit, interplanetary, and J_2-perturbed form of Lambert's problem. Lambert's problem is concerned with the determination of an orbit that passes between two positions within a specified time-of-flight. This classic astrodynamic problem is also known as the orbital two-point boundary value problem (TPBVP) or the flyby and rendezvous problems.

Lambert's theorem

The time to traverse a trajectory depends only upon the length of the semimajor axis a of the transfer trajectory, the sum $r_i + r_f$ of the distances of the initial and final positions relative to a central body, and the length c of the chord joining these two positions. This relationship can be stated as follows:

$$tof = tof\left(r_i + r_f, c, a\right)$$

From the following form of Kepler's equation

$$t - t_0 = \sqrt{\frac{a^3}{\mu}\left(E - e\sin E\right)}$$

we can write

$$t = \sqrt{\frac{a^3}{\mu}}\left[E - E_0 - e\left(\sin E - \sin E_0\right)\right]$$

where E is the eccentric anomaly associated with radius r, E_0 is the eccentric anomaly at r_0, and $t = 0$ when $r = r_0$.

At this point we need to introduce the following trigonometric sun and difference identities:

$$\sin\alpha - \sin\beta = 2\sin\frac{\alpha-\beta}{2}\cos\frac{a+\beta}{2}$$
$$\cos\alpha - \cos\beta = -2\sin\frac{\alpha-\beta}{2}\sin\frac{a+\beta}{2}$$
$$\cos\alpha + \cos\beta = 2\cos\frac{\alpha-\beta}{2}\cos\frac{a+\beta}{2}$$

If we let $E = \alpha$ and $E_0 = \beta$ and substitute the first trig identity into the second equation above, we have the following equation:

$$t = \sqrt{\frac{a^3}{\mu}}\left\{E - E_0 - 2\sin\frac{E-E_0}{2}\left(e\cos\frac{E+E_0}{2}\right)\right\}$$

With the two substitutions given by

$$e\cos\frac{E+E_0}{2}=\cos\frac{\alpha+\beta}{2}$$
$$\sin\frac{E-E_0}{2}=\sin\frac{\alpha-\beta}{2}$$

the time equation becomes

$$t=\sqrt{\frac{a^3}{\mu}}\left\{(\alpha-\beta)-2\sin\frac{\alpha-\beta}{2}\cos\frac{\alpha+\beta}{2}\right\}$$

From the elliptic relationships given by

$$r=a\left(1-e\cos E\right)$$

$$x=a\left(\cos E-e\right)$$

$$y=a\sin E\sqrt{1-e^2}$$

and some more manipulation, we have the following equations:

$$\cos\alpha=\left(1-\frac{r+r_0}{2a}\right)-\frac{c}{2a}=1-\frac{r+r_0+c}{2a}=1-\frac{s}{a}$$

$$\sin\beta=\left(1-\frac{r+r_0}{2a}\right)+\frac{c}{2a}=1-\frac{r+r_0-c}{2a}=1-\frac{s-c}{a}$$

This part of the derivation makes use of the following three relationships:

$$\cos\frac{\alpha-\beta}{2}\cos\frac{\alpha+\beta}{2}=1-\frac{r+r_0}{2}$$

$$\sin\frac{\alpha-\beta}{2}\sin\frac{\alpha+\beta}{2}=\sin\frac{E-E_0}{2}\sqrt{1-\left(e\cos\frac{E+E_0}{2}\right)^2}$$

$$\left(\sin\frac{\alpha-\beta}{2}\sin\frac{\alpha+\beta}{2}\right)^2=\left(\frac{x-x_0}{2a}\right)^2+\left(\frac{y-y_0}{2a}\right)^2=\left(\frac{c}{2a}\right)^2$$

With the use of the half angle formulas given by

$$\sin\frac{\alpha}{2}=\sqrt{\frac{s}{2a}}\qquad\sin\frac{\beta}{2}=\sqrt{\frac{s-c}{2a}}$$

and several additional substitutions, we have the time-of-flight form of Lambert's theorem

$$t = \sqrt{\frac{a^3}{\mu}}\left[(\alpha - \beta) - (\sin\alpha - \sin\beta)\right]$$

A discussion about the angles α and β can be found in "Geometrical Interpretation of the Angles α and β in Lambert's Problem" by J. E. Prussing, AIAA *Journal of Guidance and Control*, Volume 2, Number 5, Sept.-Oct. 1979, pages 442-443.

The algorithm used in these MATLAB scripts is based on the method described in "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem" by R. H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990. This iterative solution is valid for elliptic, parabolic and hyperbolic transfer orbits which may be either posigrade or retrograde, and involve one or more revolutions about the central body.

Primer Vector Analysis

This section summarizes the primer vector analysis included with the `lambert1.m` MATLAB script. The term primer vector was invented by Derek F. Lawden and represents the adjoint vector for velocity. A technical discussion about primer theory can be found in Lawden's classic text, *Optimal Trajectories for Space Navigation*, Butterworths, London, 1963. Another excellent resource is "Primer Vector Theory and Applications", Donald J. Jezewski, NASA TR R-454, November 1975, along with "Optimal, Multi-burn, Space Trajectories", also by Jezewski.

As shown by Lawden, the following four necessary conditions must be satisfied in order for an impulsive orbital transfer to be *locally optimal*:

(1) the primer vector and its first derivative are everywhere continuous

(2) whenever a velocity impulse occurs, the primer is a unit vector aligned with the impulse and has unit magnitude $\left(\mathbf{p} = \hat{\mathbf{p}} = \hat{\mathbf{u}}_T \text{ and } \|\mathbf{p}\| = 1\right)$

(3) the magnitude of the primer vector may not exceed unity on a coasting arc $\left(\|\mathbf{p}\| = p \le 1\right)$

(4) at all interior impulses (not at the initial or final times) $\mathbf{p} \bullet \dot{\mathbf{p}} = 0$; therefore, $d\|\mathbf{p}\|/dt = 0$ at the intermediate impulses

Furthermore, the scalar magnitudes of the primer vector derivative at the initial and final impulses provide information about how to improve the nominal transfer trajectory by changing the endpoint times and/or moving the impulse times. These four cases for non-zero slopes are summarized as follows;

- If $\dot{p}_0 > 0$ and $\dot{p}_f < 0 \rightarrow$ perform an initial coast before the first impulse and add a final coast after the second impulse
- If $\dot{p}_0 > 0$ and $\dot{p}_f > 0 \rightarrow$ perform an initial coast before the first impulse and move the second impulse to a later time

- If $\dot{p}_0 < 0$ and $\dot{p}_f < 0 \rightarrow$ perform the first impulse at an earlier time and add a final coast after the second impulse
- If $\dot{p}_0 < 0$ and $\dot{p}_f > 0 \rightarrow$ perform the first impulse at an earlier time and move the second impulse to a later time

The primer vector analysis of a two impulse orbital transfer involves the following steps.

First partition the two-body state transition matrix as follows:

$$\Phi(t,t_0) = \begin{bmatrix} \dfrac{\partial \mathbf{r}}{\partial \mathbf{r}_0} & \dfrac{\partial \mathbf{r}}{\partial \mathbf{v}_0} \\ \dfrac{\partial \mathbf{v}}{\partial \mathbf{r}_0} & \dfrac{\partial \mathbf{v}}{\partial \mathbf{v}_0} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix}$$

where

$$\Phi_{11} = \left[\frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} \right] = \begin{bmatrix} \partial x / \partial x_0 & \partial x / \partial y_0 & \partial x / \partial z_0 \\ \partial y / \partial x_0 & \partial y / \partial y_0 & \partial y / \partial z_0 \\ \partial z / \partial x_0 & \partial z / \partial y_0 & \partial z / \partial z_0 \end{bmatrix}$$

and so forth.

The value of the primer vector at any time *t* along a two body trajectory is given by

$$\mathbf{p}(t) = \Phi_{11}(t,t_0)\mathbf{p}_0 + \Phi_{12}(t,t_0)\dot{\mathbf{p}}_0$$

and the value of the primer vector derivative is

$$\dot{\mathbf{p}}(t) = \Phi_{21}(t,t_0)\mathbf{p}_0 + \Phi_{22}(t,t_0)\dot{\mathbf{p}}_0$$

which can also be expressed as

$$\begin{Bmatrix} \mathbf{p} \\ \dot{\mathbf{p}} \end{Bmatrix} = \Phi(t,t_0) \begin{Bmatrix} \mathbf{p}_0 \\ \dot{\mathbf{p}}_0 \end{Bmatrix}$$

The primer vector boundary conditions at the initial and final impulses are as follows:

$$\mathbf{p}(t_0) = \mathbf{p}_0 = \frac{\Delta \mathbf{V}_0}{\left| \Delta \mathbf{V}_0 \right|}$$

$$\mathbf{p}(t_f) = \mathbf{p}_f = \frac{\Delta \mathbf{V}_f}{\left| \Delta \mathbf{V}_f \right|}$$

These two conditions illustrate that at the locations of velocity impulses, the primer vector is a unit vector in the direction of the impulses.

The value of the primer vector derivative at the initial time is

$$\dot{\mathbf{p}}(t_0) = \dot{\mathbf{p}}_0 = \Phi_{12}^{-1}(t_f, t_0)\left\{\mathbf{p}_f - \Phi_{11}(t_f, t_0)\mathbf{p}_0\right\}$$

provided the Φ_{12} sub-matrix is non-singular.

The scalar magnitude of the derivative of the primer vector can be determined from

$$\frac{d\|\mathbf{p}\|}{dt} = \frac{d}{dt}(\mathbf{p}\bullet\mathbf{p})^2 = \frac{\dot{\mathbf{p}}\bullet\mathbf{p}}{\|\mathbf{p}\|}$$

lambert1.m – Earth orbit solution

This MATLAB application demonstrates how to solve the two-body form of Lambert's problem for a satellite in Earth orbit. The following is a typical user interaction with this script.

```
              program lambert1

     < Earth orbit lambert problem >


   initial orbit

   please input the semimajor axis (kilometers)
   (semimajor axis > 0)
   ? 8000

   please input the orbital eccentricity (non-dimensional)
   (0 <= eccentricity < 1)
   ? 0

   please input the orbital inclination (degrees)
   (0 <= inclination <= 180)
   ? 28.5

   please input the right ascension of the ascending node (degrees)
   (0 <= raan <= 360)
   ? 100

   please input the true anomaly (degrees)
   (0 <= true anomaly <= 360)
   ? 0

   final orbit

   please input the semimajor axis (kilometers)
   (semimajor axis > 0)
   ? 8000
```

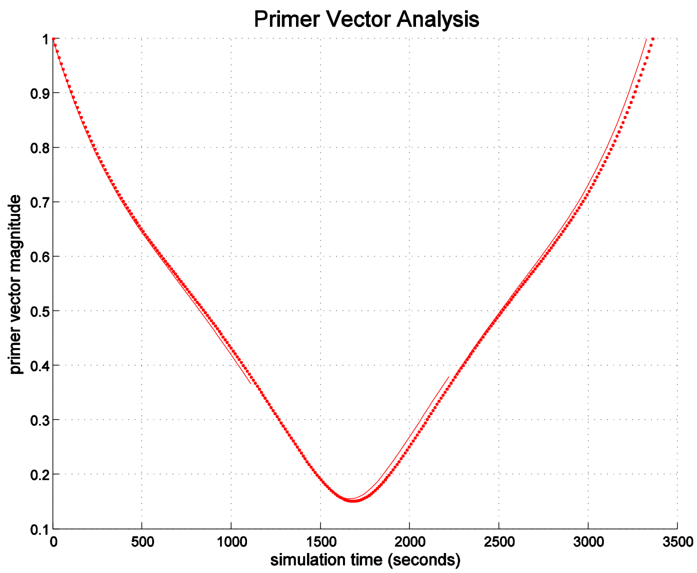
```
please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 170


please input the transfer time in minutes
? 56


 orbital direction

  <1> posigrade

  <2> retrograde

 selection (1 or 2)
? 1

please input the maximum number of transfer orbits around the Earth
? 0
```

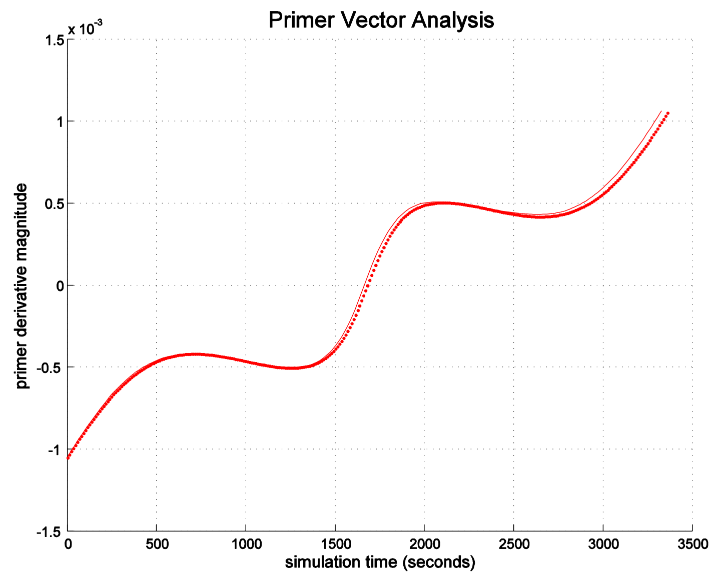
The following is the output created by this application.

```
        program lambert1

< Earth orbit lambert problem >

orbital elements of the initial orbit

       sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000000e+003  +0.00000000000000e+000  +2.85000000000000e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000000e+002  +0.00000000000000e+000  +0.00000000000000e+000  +1.18684693004297e+002


orbital elements of the transfer orbit after the initial delta-v

       sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00047140990639e+003  +6.70937482986916e-004  +2.85000000000000e+001  +8.49999999999880e+001

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000000e+002  +2.75000000000012e+002  +0.00000000000000e+000  +1.18695183622590e+002


orbital elements of the transfer orbit prior to the final delta-v

       sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00047140990639e+003  +6.70937482986917e-004  +2.85000000000000e+001  +8.49999999999975e+001
```

```
       raan (deg)          true anomaly (deg)          arglat (deg)           period (min)
 +1.00000000000000e+002  +8.50000000000025e+001  +1.70000000000000e+002  +1.18695183622590e+002


orbital elements of the final orbit

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000000e+003  +0.00000000000000e+000  +2.85000000000000e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)          arglat (deg)           period (min)
 +1.00000000000000e+002  +1.70000000000000e+002  +1.70000000000000e+002  +1.18684693004297e+002


initial delta-v vector and magnitude

x-component of delta-v          0.640619  meters/second
y-component of delta-v         -4.677599  meters/second
z-component of delta-v          0.098476  meters/second

delta-v magnitude               4.722290  meters/second

final delta-v vector and magnitude

x-component of delta-v         -0.279892  meters/second
y-component of delta-v          4.704815  meters/second
z-component of delta-v         -0.293925  meters/second

delta-v magnitude               4.722290  meters/second

total delta-v                   9.444579  meters/second

transfer time                  56.000000  minutes
```

The graphical primer vector analysis for this example is shown below. These plots illustrate the behavior of the scalar magnitudes of the primer vector and its derivative as a function of the orbit transfer time.





lambert2.m – interplanetary solution

This MATLAB script demonstrates how to solve the two-body interplanetary Lambert problem. The following is a typical user interaction with this script.

```
program lambert2

< interplanetary lambert problem >


departure conditions

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 9,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0


arrival conditions

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 8,15,1999

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

 planetary menu

  <1> Mercury
  <2> Venus
  <3> Earth
```

```
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto

please select the departure planet
? 3

 planetary menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto

please select the arrival planet
? 4
```

The following is the program output for this example.

```
         program lambert2

< interplanetary lambert problem >

departure planet            'Earth'

departure calendar date      01-Sep-1998
departure universal time     00:00:00.000

departure julian date        2451057.500000

arrival planet              'Mars'

arrival calendar date        15-Aug-1999
arrival universal time       00:00:00.000

arrival julian date          2451405.500000

transfer time                  348.000000  days

heliocentric ecliptic orbital elements of the departure planet

      sma (km)        eccentricity     inclination (deg)    argper (deg)
 1.4959802229e+008  1.6709181047e-002  0.0000000000e+000  1.0291439752e+002

     raan (deg)     true anomaly (deg)   arglat (deg)       period (days)
 0.0000000000e+000  2.3546050380e+002  3.3837490132e+002  3.6525745091e+002


heliocentric ecliptic orbital elements of the transfer orbit after the initial delta-v

      sma (km)        eccentricity     inclination (deg)    argper (deg)
 1.7145607013e+008  3.3064571581e-001  1.4025406657e+000  8.8020190015e+001
```

```
     raan (deg)     true anomaly (deg)   arglat (deg)       period (days)
 3.3837490132e+002  2.7197980998e+002  0.0000000000e+000  4.4816672004e+002


heliocentric ecliptic orbital elements of the transfer orbit prior to the final delta-v

      sma (km)        eccentricity     inclination (deg)    argper (deg)
 1.7145607013e+008  3.3064571581e-001  1.4025406657e+000  8.8020190015e+001

     raan (deg)     true anomaly (deg)   arglat (deg)       period (days)
 3.3837490132e+002  2.0668220696e+002  2.9470239697e+002  4.4816672004e+002


heliocentric ecliptic orbital elements of the arrival planet

      sma (km)        eccentricity     inclination (deg)    argper (deg)
 2.2793918413e+008  9.3400274417e-002  1.8497282956e+000  2.8649805839e+002

     raan (deg)     true anomaly (deg)   arglat (deg)       period (days)
 4.9555144147e+001  2.9704552666e+002  2.2354358506e+002  6.8697161038e+002


initial delta-v vector and magnitude

x-component of delta-v      -8563.836300  meters/second
y-component of delta-v       3718.454469  meters/second
z-component of delta-v        729.797537  meters/second

delta-v magnitude            9364.763759  meters/second

energy                         87.698800  km^2/sec^2


final delta-v vector and magnitude

x-component of delta-v       4608.052564  meters/second
y-component of delta-v      -2097.558698  meters/second
z-component of delta-v       -856.066401  meters/second

delta-v magnitude            5134.856434  meters/second

energy                         26.366751  km^2/sec^2
```

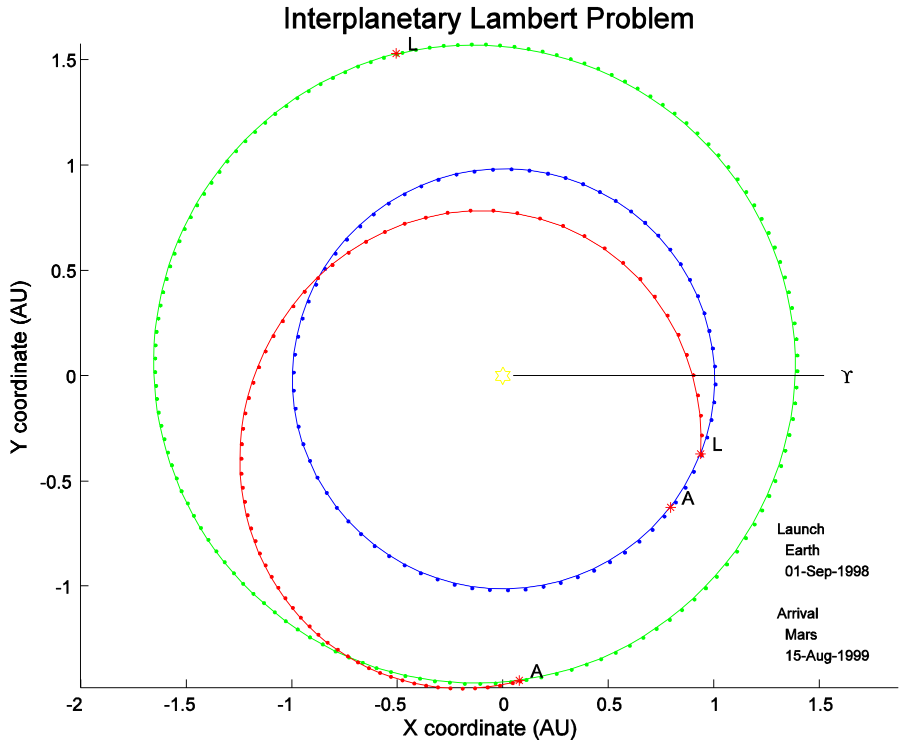
After the script computes the numerical data, it will ask the user if he or she would like to create a graphics display of the trajectory. This prompt appears as

```
would you like to plot this trajectory (y = yes, n = no)
?
```

If the user responds with `y` for yes, the script will ask for the plot step size with

```
please input the plot step size (days)
?
```

The following is a typical graphics display created with this MATLAB script. The plot is a *north ecliptic* view where we are looking down on the ecliptic plane from the north celestial pole. The vernal equinox direction is the labeled line pointing to the right, the launch planet is labeled with an L and the arrival planet is labeled with an A. The location of the launch and arrival planets at the launch time is marked with an asterisk. The plot step size for this example is 5 days.



lambert3.m – perturbed motion solution – shooting method with state transition matrix updates

This MATLAB script demonstrates how to solve the J_2-perturbed Earth orbit Lambert problem. However, more sophisticated equations of motion can easily be implemented. The algorithm solves this problem using a simple *shooting* technique.

An initial guess for this algorithm is created by first solving the two-body form of Lambert's problem. At each *shooting* iteration, the initial delta-velocity vector is updated according to

$$\Delta\mathbf{v} = \left[\Phi_{12}\right]^{-1}\Delta\mathbf{r}$$

where the error in the final position vector $\Delta\mathbf{r}$ is determined from the difference between the two body final position vector $\mathbf{r}_{tb}$ and the final position vector predicted by numerical integration $\mathbf{r}_{\text{int}}$ of the orbital equations of motion as follows:

$$\Delta\mathbf{r} = \mathbf{r}_{tb} - \mathbf{r}_{\text{int}}$$

The new initial velocity vector can now be calculated from

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta\mathbf{v}$$

The sub-matrix Φ_{12} of the full state transition matrix is as follows:

$$\Phi_{12} = \left[\frac{\partial \mathbf{r}}{\partial \mathbf{v}_0} \right] = \begin{bmatrix} \partial x / \partial \dot{x}_0 & \partial x / \partial \dot{y}_0 & \partial x / \partial \dot{z}_0 \\ \partial y / \partial \dot{x}_0 & \partial y / \partial \dot{y}_0 & \partial y / \partial \dot{z}_0 \\ \partial z / \partial \dot{x}_0 & \partial z / \partial \dot{y}_0 & \partial z / \partial \dot{z}_0 \end{bmatrix}$$

This sub-matrix consists of the partial derivatives of the rectangular components of the final position vector with respect to the initial velocity vector.

The following is a typical user interaction with this script.

```
              program lambert3

< j2 perturbed Earth orbit lambert problem >


initial orbit

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 0


final orbit

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100
```

```
please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 170


please input the transfer time in minutes
? 56
```

The following is the program output for this example. Please note that the program displays both the Keplerian (two body) and perturbed solutions for the transfer orbit.

```
                  program lambert3

        j2 perturbed Earth orbit lambert problem

  shooting method with state transition matrix updates


orbital elements of the initial orbit

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000000e+003  +0.00000000000000e+000  +2.85000000000000e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)        arglat (deg)             period (min)
 +1.00000000000000e+002  +0.00000000000000e+000  +0.00000000000000e+000  +1.18684693788431e+002


orbital elements of the final orbit

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000000e+003  +0.00000000000000e+000  +2.85000000000000e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)        arglat (deg)             period (min)
 +1.00000000000000e+002  +1.70000000000000e+002  +1.70000000000000e+002  +1.18684693788431e+002


keplerian transfer orbit

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00047141376779e+003  +6.70942937076629e-004  +2.85000000000000e+001  +8.50000000000164e+001

       raan (deg)          true anomaly (deg)        arglat (deg)             period (min)
 +1.00000000000000e+002  +2.74999999999984e+002  +0.00000000000000e+000  +1.18695184492725e+002


j2 perturbed transfer orbit

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00723489628623e+003  +9.68258357992949e-004  +2.89460861696644e+001  +2.10862530931085e+001

       raan (deg)          true anomaly (deg)        arglat (deg)             period (min)
 +1.00000000000000e+002  +3.38913746906891e+002  +0.00000000000000e+000  +1.18845731080656e+002


delta-v vector and magnitude

x-component of delta-v         23.6892  meters/second
y-component of delta-v          1.6813  meters/second
z-component of delta-v         49.7371  meters/second

total delta-v                  55.1161  meters/second

transfer time                  56.0000  minutes
```

```
final position vector error components and magnitude

x-component of delta-r        0.00000999  meters
y-component of delta-r        0.00000171  meters
z-component of delta-r        0.00000460  meters

delta-r magnitude             0.00001113  meters
```

lambert4.m – perturbed motion solution – NLP solution

This MATLAB application demonstrates how to solve the J_2-perturbed Earth orbit Lambert problem. However, more sophisticated equations of motion can easily be implemented. The algorithm solves this problem using a *nonlinear programming* technique. This script can solve both the flyby and rendezvous problems. For the flyby problem, the program attempts to match all three components of the position vector. For the rendezvous problem, the script attempts to match all three components of both the target position and velocity vectors.

SNOPT algorithm implementation

This section provides details about the part of the `lambert4` script that solves this nonlinear programming (NLP) problem using the SNOPT algorithm. In this classic trajectory optimization problem, the components of the initial and final delta-v vectors are the *control variables* and the scalar magnitude of the flyby or rendezvous ΔV is the *objective function* or *performance index*.

MATLAB versions of SNOPT for several computer platforms can be found at Professor Philip Gill's web site which is located at http://cam.ucsd.edu/~peg/Software.html.

The SNOPT algorithm requires an initial guess for the control variables. For this example they are determined from the two-body solution of Lambert's problem. The algorithm also requires lower and upper bounds for the control variables. These are determined from the initial guesses as follows:

```
% define lower and upper bounds for components of delta-v vectors
(kilometers/second)

for i = 1:1:3
    xlwr(i) = min(-1.1 * norm(xg(1:3)), -75.0);

    xupr(i) = max(+1.1 * norm(xg(1:3)), +75.0);
end

if (otype == 2)
   for i = 4:1:6
       xlwr(i) = min(-1.1 * norm(xg(4:6)), -75.0);

       xupr(i) = max(+1.1 * norm(xg(4:6)), +75.0);
   end
end
```

The algorithm requires lower and upper bounds on the objective function. For this problem these bounds are given by

```
% bounds on objective function

flow(1) = 0.0d0;
fupp(1) = +Inf;
```

Finally, the NLP algorithm also requires the following state vector *equality* constraints.

```
% enforce final position vector equality constraints

flow(2) = 0.0d0;
fupp(2) = 0.0d0;

flow(3) = 0.0d0;
fupp(3) = 0.0d0;

flow(4) = 0.0d0;
fupp(4) = 0.0d0;

if (otype == 2)
   % enforce final velocity vector equality constraints

   flow(5) = 0.0d0;
   fupp(5) = 0.0d0;

   flow(6) = 0.0d0;
   fupp(6) = 0.0d0;

   flow(7) = 0.0d0;
   fupp(7) = 0.0d0;
end
```

The actual call to the SNOPT MATLAB interface function is as follows

```
[x, f, inform, xmul, fmul] = snopt(xg, xlwr, xupr, flow, fupp, 'tpbvp');
```

where `tpbvp` is the name of the MATLAB function that solves Lambert's problem and computes the current value of the objective function and equality constraints. The following is the MATLAB source code for this example.

```
function [f, g] = tpbvp(x)

% two point boundary value objective function
% and state vector constraints

% input

%  x = current delta-v vector

% output

%  f(1) = objective function (delta-v magnitude)
%  f(2) = rx constraint delta
%  f(3) = ry constraint delta
%  f(4) = rz constraint delta
%  f(5) = vx constraint delta
%  f(6) = vy constraint delta
%  f(7) = vz constraint delta

% Orbital Mechanics with Matlab
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global otype neq tetol

global ri vi tof rtarget vtarget drf dvf

% load current state vector of transfer orbit

xi(1) = ri(1);
xi(2) = ri(2);
xi(3) = ri(3);

xi(4) = vi(1) + x(1);
xi(5) = vi(2) + x(2);
xi(6) = vi(3) + x(3);

% initial guess for step size (seconds)

h = 10.0;

% initial time (seconds)

ti = 0.0;

% final time (seconds)

tf = tof;

% integrate equations of motion

xf = rkf78 ('j2eqm', neq, ti, tf, h, tetol, xi);

% objective function (delta-v magnitude)

if (otype == 1)
   % initial delta-v only (flyby)

   f(1) = norm(x);
else
   % total delta-v (rendezvous)

   f(1) = norm(x(1:3)) + norm(x(4:6));
end

% final position vector equality constraints

f(2) = rtarget(1) - xf(1);

f(3) = rtarget(2) - xf(2);

f(4) = rtarget(3) - xf(3);

if (otype == 2)
   % final velocity vector
```

```
   vf(1) = xf(4) + x(4);
   vf(2) = xf(5) + x(5);
   vf(3) = xf(6) + x(6);
end

if (otype == 2)
   % enforce final velocity vector constraints

   f(5) = vtarget(1) - vf(1);

   f(6) = vtarget(2) - vf(2);

   f(7) = vtarget(3) - vf(3);
end

% save state vector deltas for print summary

for i = 1:1:3
    drf(i) = f(i + 1);

    if (otype == 2)
       % rendezvous

       dvf(i) = f(i + 4);
    end
end

% transpose objective function/constraints vector

f = f';

% no derivatives

g = [];
```

The following is a typical user interaction with this MATLAB script.

```
             program lambert4

< perturbed Earth orbit Lambert problem >


trajectory type (1 = flyby, 2 = rendezvous)
? 1

classical orbital elements of the initial orbit

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0
```

```
please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 0

classical orbital elements of the final orbit

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 170

please input the transfer time in minutes
? 56
```

The following is the script output for this example. The first part of the display includes the two-body solution for the initial guess and the optimization summary from SNOPT.

```
two-body guess for initial delta-v vector and magnitude

x-component of delta-v          0.640625  meters/second
y-component of delta-v         -4.677637  meters/second
z-component of delta-v          0.098476  meters/second

delta-v magnitude               4.722328  meters/second

 Major Minors     Step   nCon Feasible  Optimal  MeritFunction    nS Penalty
     0      2               1  2.8E+01  1.5E-02  4.7223280E-03                r
     1      0  3.6E-03      2  2.8E+01  1.5E-02  4.7179225E-03              n r
     2      0  1.4E-03      3  2.8E+01  5.4E+00  6.4112054E-03       3.0E-06 s
     3      0  1.0E+00      4  1.6E+00  4.0E-02  3.7031413E-02       8.9E-04
     4      0  1.0E+00      5  1.2E-03  7.7E-05  5.5116083E-02       1.2E-02  m
     5      0  1.0E+00      6 (2.6E-09)(1.2E-10) 5.5116073E-02       1.2E-02 n     c
     5      0  1.0E+00      6 (2.6E-09)(1.2E-10) 5.5116073E-02       1.2E-02 n     c

   EXIT -- optimal solution found
```

```
            program lambert4

< perturbed Earth orbit Lambert problem >


orbital elements and state vector of the initial orbit
------------------------------------------------------

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000000e+003  +0.00000000000000e+000  +2.85000000000000e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000000e+002  +0.00000000000000e+000  +0.00000000000000e+000  +1.18684693788431e+002

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 -1.38918542133544e+003  +7.87846202409766e+003  +0.00000000000000e+000  +8.00000000000000e+003

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 -6.10905247177800e+000  -1.07719077733820e+000  +3.36811407992746e+000  +7.05868645918807e+000


orbital elements and state vector of the transfer orbit after the initial delta-v
---------------------------------------------------------------------------------

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00723489628470e+003  +9.68258357783872e-004  +2.89460861700788e+001  +2.10862530925057e+001

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000000e+002  +3.38913746907494e+002  +0.00000000000000e+000  +1.18845731080622e+002

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 -1.38918542133544e+003  +7.87846202409766e+003  +0.00000000000000e+000  +8.00000000000000e+003

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 -6.08536330087357e+000  -1.07550945881513e+000  +3.41785116756283e+000  +7.06187465926933e+000


orbital elements and state vector of the transfer orbit prior to the final delta-v
----------------------------------------------------------------------------------

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00712131217672e+003  +9.65333285374587e-004  +2.89453516511177e+001  +1.47239268484716e+002

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +9.98377817600496e+001  +2.29029908765670e+001  +1.70142259361283e+002  +1.18843202316575e+002

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 +1.65787953981183e+002  -7.97076711063437e+003  +6.62861993417606e+002  +7.99999999999834e+003

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 +6.20553203642353e+000  -1.53599271890635e-001  -3.36706800686018e+000  +7.06182466181549e+000


orbital elements and state vector of the final orbit
----------------------------------------------------

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +7.99999999999668e+003  +4.93755404440110e-013  +2.85000000000013e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000015e+002  +1.69999999999961e+002  +1.69999999999961e+002  +1.18684693788357e+002

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 +1.65787953981183e+002  -7.97076711063437e+003  +6.62861993417606e+002  +7.99999999999834e+003

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 +6.22908767828918e+000  -1.46280648235324e-001  -3.31694485894215e+000  +7.05868645918807e+000
```

```
initial delta-v vector and magnitude
------------------------------------

x-component of delta-v         23.689171  meters/second
y-component of delta-v          1.681319  meters/second
z-component of delta-v         49.737088  meters/second

delta-v magnitude              55.116073  meters/second


final position vector error components and magnitude
----------------------------------------------------

x-component of delta-r        0.00000271  meters
y-component of delta-r       -0.00000181  meters
z-component of delta-r       -0.00000246  meters

delta-r magnitude             0.00000409  meters

transfer time                  56.000000  minutes
```

Here's the rendezvous option solution for the same initial conditions.

```
two-body guess for initial delta-v vector and magnitude

x-component of delta-v          0.640625  meters/second
y-component of delta-v         -4.677637  meters/second
z-component of delta-v          0.098476  meters/second

delta-v magnitude               6.678380  meters/second


two-body guess for final delta-v vector and magnitude

x-component of delta-v         -0.279895  meters/second
y-component of delta-v          4.704854  meters/second
z-component of delta-v         -0.293927  meters/second

delta-v magnitude               4.722328  meters/second

total delta-v                   9.444656  meters/second

 Major Minors     Step   nCon Feasible  Optimal  MeritFunction    nS Penalty
     0      2               1  2.8E+01  1.5E-02  9.4446559E-03                 r
     1      0  3.2E-03      2  2.8E+01  1.5E-02  9.4374421E-03               n r
     2      0  1.4E-03      3  2.8E+01  9.2E-01  1.2614431E-02       5.5E-06 s
     3      0  1.0E+00      4  1.5E+00  9.6E-02  7.2963247E-02       1.8E-03
     4      0  1.0E+00      5  1.2E-03  3.1E-04  1.1097988E-01       2.5E-02  m
     5      0  1.0E+00      6 (4.9E-09)(1.5E-09) 1.1097984E-01       2.5E-02 n     c
     5      0  1.0E+00      6 (4.9E-09)(1.5E-09) 1.1097984E-01       2.5E-02 n     c

   EXIT -- optimal solution found

            program lambert4

< perturbed Earth orbit Lambert problem >


orbital elements and state vector of the initial orbit
------------------------------------------------------

        sma (km)              eccentricity           inclination (deg)         argper (deg)
 +8.00000000000000e+003  +0.00000000000000e+000  +2.85000000000000e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000000e+002  +0.00000000000000e+000  +0.00000000000000e+000  +1.18684693788431e+002
```

```
        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 -1.38918542133544e+003  +7.87846202409766e+003  +0.00000000000000e+000  +8.00000000000000e+003

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 -6.10905247177800e+000  -1.07719077733820e+000  +3.36811407992746e+000  +7.05868645918807e+000


orbital elements and state vector of the transfer orbit after the initial delta-v
---------------------------------------------------------------------------------

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00723489628503e+003  +9.68258357800548e-004  +2.89460861700790e+001  +2.10862530883712e+001

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000000e+002  +3.38913746911629e+002  +0.00000000000000e+000  +1.18845731080629e+002

        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 -1.38918542133544e+003  +7.87846202409766e+003  +0.00000000000000e+000  +8.00000000000000e+003

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 -6.08536330087376e+000  -1.07550945881474e+000  +3.41785116756291e+000  +7.06187465926947e+000


orbital elements and state vector of the transfer orbit prior to the final delta-v
----------------------------------------------------------------------------------

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00712131217702e+003  +9.65333285316214e-004  +2.89453516511178e+001  +1.47239268486159e+002

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +9.98377817600496e+001  +2.29029908751038e+001  +1.70142259361262e+002  +1.18843202316582e+002

        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 +1.65787953978664e+002  -7.97076711063496e+003  +6.62861993419040e+002  +7.99999999999899e+003

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 +6.20553203642318e+000  -1.53599271892845e-001  -3.36706800685980e+000  +7.06182466181505e+000


orbital elements and state vector of the final orbit
----------------------------------------------------

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +7.99999999998343e+003  +2.05098256580623e-012  +2.84999999999968e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000018e+002  +1.69999999999938e+002  +1.69999999999938e+002  +1.18684693788062e+002

        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 +1.65787953978664e+002  -7.97076711063496e+003  +6.62861993419040e+002  +7.99999999999899e+003

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 +6.22908767828381e+000  -1.46280648236246e-001  -3.31694485893854e+000  +7.05868645918165e+000


initial delta-v vector and magnitude
------------------------------------

x-component of delta-v         23.689171  meters/second
y-component of delta-v          1.681319  meters/second
z-component of delta-v         49.737088  meters/second

delta-v magnitude              78.476379  meters/second

final delta-v vector and magnitude
----------------------------------

x-component of delta-v         23.555642  meters/second
y-component of delta-v          7.318624  meters/second
z-component of delta-v         50.123148  meters/second
```

```
delta-v magnitude              55.863767  meters/second


total delta-v                 110.979840  meters/second


final position vector error components and magnitude
----------------------------------------------------

x-component of delta-r        0.00000523  meters
y-component of delta-r       -0.00000123  meters
z-component of delta-r       -0.00000390  meters

delta-r magnitude             0.00000664  meters

final velocity vector error components and magnitude
----------------------------------------------------

x-component of delta-v        0.00000001  meters/second
y-component of delta-v        0.00000000  meters/second
z-component of delta-v       -0.00000000  meters/second

delta-v magnitude             0.00000001  meters/second


transfer time                  56.000000  minutes
```

Lambert Functions

This section describes two MATLAB functions that solve the two-body form of Lambert's boundary value problem.

glambert.m – Gooding's solution of Lambert's problem

This two-body Lambert function has the following syntax.

```
function [vi, vf] = glambert(cbmu, sv1, sv2, tof, nrev)

% Gooding's solution of Lambert's problem

% input

%  cbmu = central body gravitational constant
%  sv1  = initial 6-element state vector (position + velocity)
%  sv2  = final 6-element state vector (position + velocity)
%  tof  = time of flight (+ posigrade, - retrograde)
%  nrev = number of full revolutions
%         (positive for long period orbit,
%          negative for short period orbit)

% output

%  vi = initial velocity vector of the transfer orbit
%  vf = final velocity vector of the transfer orbit
```

lambfunc.m – Gedeon's solution of Lambert's problem

The algorithm used in this MATLAB function is based on the method described in "A Practical Note on the Use of Lambert's Equation" by Geza Gedeon, *AIAA Journal*, Volume 3, Number 1, 1965, pages 149-150. This iterative solution is valid for elliptic, parabolic and hyperbolic transfer orbits which may be either posigrade or retrograde, and involve one or more revolutions about the central body. Additional information can also be found in G. S. Gedeon, "Lambertian Mechanics", Proceedings of the 12th International Astronautical Congress, Vol. I, 172-190.

The *elliptic* form of the general Lambert Theorem is

$$t = \sqrt{\frac{a^3}{\mu}}\left[(1-k)m\pi + k(\alpha - \sin\alpha) \mp (\beta - \sin\beta)\right]$$

where k may be either +1 (posigrade) or –1 (retrograde), and m is the number of revolutions about the central body.

The Gedeon algorithm introduces the following variable

$$z = \frac{s}{2a}$$

and solves the problem with a Newton-Raphson procedure. In this equation, a is the semimajor axis of the transfer orbit and

$$s = \frac{r_1 + r_2 + c}{2}$$

This algorithm also makes use of the following constant:

$$w = \pm\sqrt{1 - \frac{c}{s}}$$

The function to be solved iteratively is given by:

$$N(z) = \frac{1}{z|z|^{1/2}\,2^{1/2}}\left\{\frac{1-k}{2}m\pi + k\left[|z|^{1/2} - |z|^{1/2}(1-z)^{1/2}\right] - \left[w|z|^{1/2} - w|z|^{1/2} - w|z|^{1/2}\left(1-w^2 z\right)^{1/2}\right]\right\}$$

The Newton-Raphson algorithm also requires the derivative of this equation given by

$$N'(z) = \frac{dN}{dz} = \frac{1}{|z|2^{1/2}}\left\{\frac{k}{(1-z)^{1/2}} - \frac{w^3}{\left(1-w^2 z\right)^{1/2}} - \frac{3N(z)}{2^{1/2}}\right\}$$

The iteration for z is as follows:

$$z_{n+1} = z_n - \frac{N\left(z_n\right)}{N'\left(z_n\right)}$$

This Lambert function has the following syntax.

```
function [statev, nsol] = lambfunc(ri, rf, tof, direct, revmax)

% solve Lambert's orbital two point boundary value problem

% input

%  ri     = initial ECI position vector (kilometers)
%  rf     = final ECI position vector (kilometers)
%  tof    = time of flight (seconds)
%  direct = transfer direction (1 = posigrade, -1 = retrograde)
%  revmax = maximum number of complete orbits

% output

%  nsol   = number of solutions
%  statev = matrix of state vector solutions of the
%           transfer trajectory after the initial delta-v

%  statev(1, sn) = position vector x component
%  statev(2, sn) = position vector y component
%  statev(3, sn) = position vector z component
%  statev(4, sn) = velocity vector x component
%  statev(5, sn) = velocity vector y component
%  statev(6, sn) = velocity vector z component
%  statev(7, sn) = semimajor axis
%  statev(8, sn) = orbital eccentricity
%  statev(9, sn) = orbital inclination
%  statev(10, sn) = argument of perigee
%  statev(11, sn) = right ascension of the ascending node
%  statev(12, sn) = true anomaly

%  where sn is the solution number
```

Please note the value of the central body gravitational constant (`mu`) should be passed to this function with a `global` statement located in the main MATLAB script.