Lambert's Problem

This document describes several 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.

lambert1.m – Earth orbit solution

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

```
            program lambert1

  < two-body Earth orbit lambert problem >

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
```

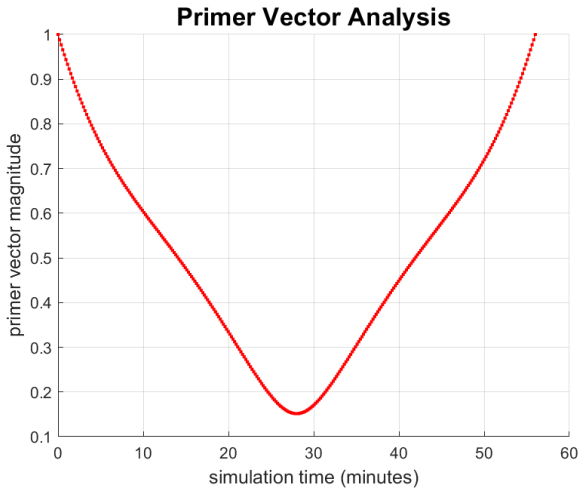
```
please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 170

please input the transfer time in minutes
? 56

orbital direction

  <1> posigrade

  <2> retrograde

 selection (1 or 2)
? 1

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

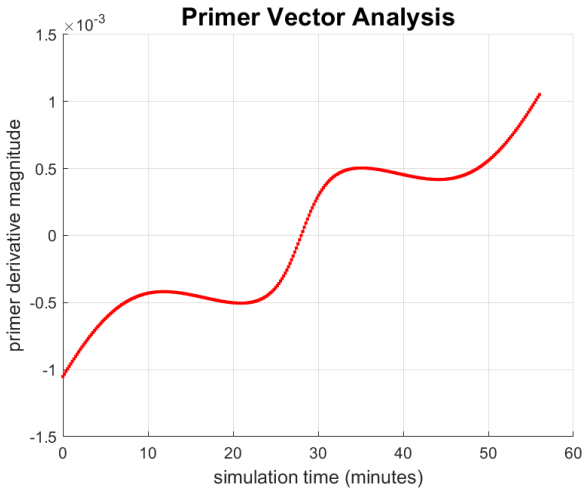
The following is the numerical output created by this application.

```
         program lambert1

< two-body Earth orbit lambert problem >

orbital elements of the initial orbit

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.000000000000000e+03  +0.000000000000000e+00  +2.850000000000000e+01  +0.000000000000000e+00

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.000000000000000e+02  +0.000000000000000e+00  +0.000000000000000e+00  +8.24199256974283e-02

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00047140990639e+03  +6.70937482986849e-04  +2.850000000000000e+01  +8.500000000000360e+01

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.000000000000000e+02  +2.749999999999964e+02  +0.000000000000000e+00  +8.24272108490207e-02

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00047140990639e+03  +6.70937482986917e-04  +2.850000000000000e+01  +8.499999999999750e+01

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.000000000000000e+02  +8.500000000000025e+01  +1.700000000000000e+02  +8.24272108490207e-02

orbital elements of the final orbit

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.000000000000000e+03  +0.000000000000000e+00  +2.850000000000000e+01  +0.000000000000000e+00

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.000000000000000e+02  +1.700000000000000e+02  +1.700000000000000e+02  +8.24199256974283e-02

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 simulation time along the transfer orbit.





The `lambert1` script will create graphics disk files of the primer and primer derivative behavior in three formats with the following code;

```
% create primer graphics disk files

print('primer.eps', '-depsc');

print('primer.jpg', '-djpeg');

print('primer.tif', '-dtiff');
```

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)
 +1.49598022290632e+08  +1.67091810467699e-02  +0.00000000000000e+00  +1.02914397517503e+02

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +0.00000000000000e+00  +2.35460503804471e+02  +3.38374901321974e+02  +3.65257450907746e+02

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +1.71456070132957e+08  +3.30645715805967e-01  +1.40254066572781e+00  +8.80201900152134e+01

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +3.38374901321974e+02  +2.71979809984787e+02  +0.00000000000000e+00  +4.48166720042418e+02

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +1.71456070132957e+08  +3.30645715805967e-01  +1.40254066572781e+00  +8.80201900152134e+01

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +3.38374901321974e+02  +2.06682206959294e+02  +2.94702396974507e+02  +4.48166720042418e+02

heliocentric ecliptic orbital elements of the arrival planet
------------------------------------------------------------

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +2.27939184126202e+08  +9.34002744169353e-02  +1.84972829558654e+00  +2.86498058394164e+02

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +4.95551441466857e+01  +2.97045526664333e+02  +2.23543585058497e+02  +6.86971610377730e+02

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  kilometers^2/seconds^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

specific orbital energy        26.366751  kilometers2/seconds^2
```

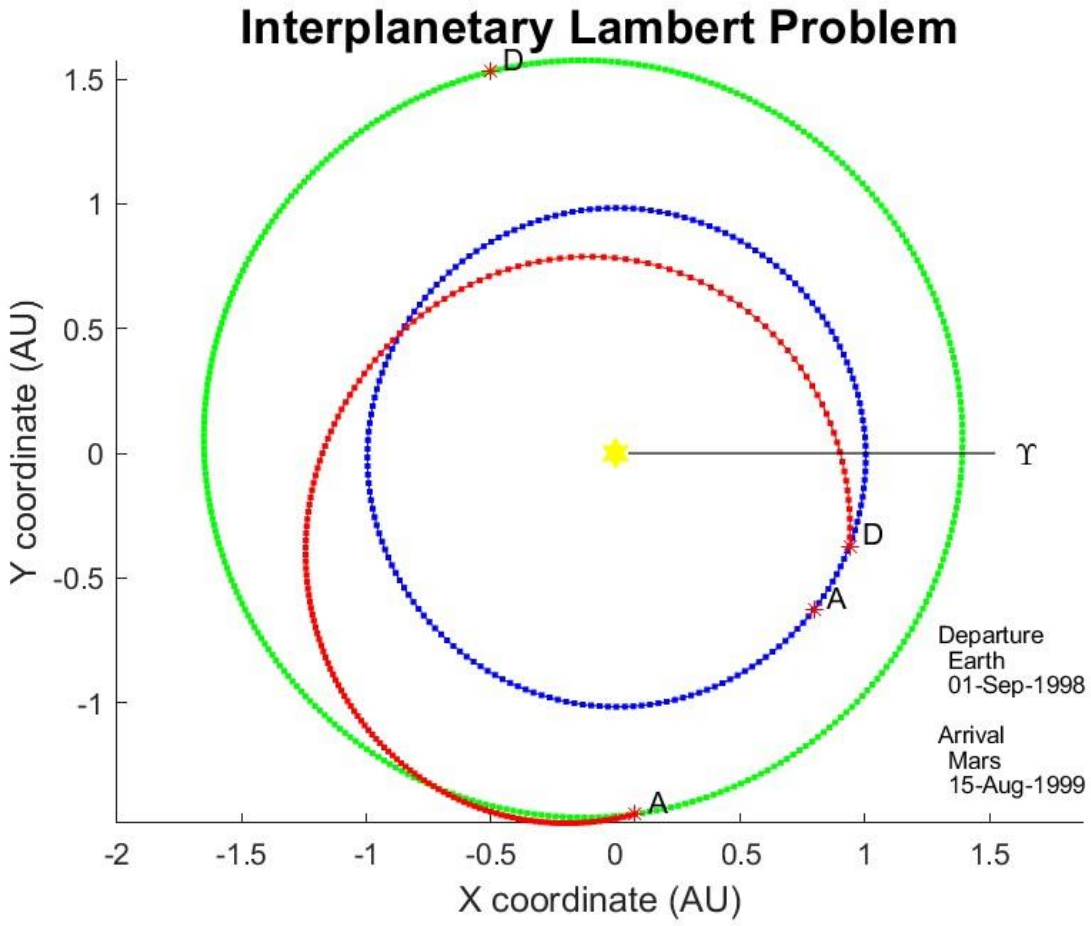
After the script computes the numerical data, it will ask the user if he or she would like to create a graphics display of the trajectory. This prompt appears as

```
would you like to display 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 graphics 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 orbit is blue, the arrival planet orbit is green and the interplanetary transfer orbit is red. The location of each planet at departure is labeled with an D and the A label is the location of each planet at arrival. These locations are marked with an asterisk. The plot step size for this example is two days. The x- and y axes are measured in astronomical units (AU).



The `lambert2` script will create graphics disk files of the orbital motion and events in three formats with the following code;

```
% create trajectory graphics disk files

print('lambert2.eps', '-depsc');

print('lambert2.jpg', '-djpeg');

print('lambert2.tif', '-dtiff');
```

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

    shooting method with state transition matrix updates

 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 program output for this example. Please note that the program displays both the Keplerian (two-body) and J_2-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+03  +0.00000000000000e+00  +2.85000000000000e+01  +0.00000000000000e+00

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +0.00000000000000e+00  +0.00000000000000e+00  +8.24199262419658e-02

orbital elements of the final orbit
-----------------------------------

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00000000000000e+03  +0.00000000000000e+00  +2.85000000000000e+01  +0.00000000000000e+00

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +1.70000000000000e+02  +1.70000000000000e+02  +8.24199262419658e-02

two-body transfer orbit
-----------------------
```

```
       sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00047141376779e+03  +6.70942937076585e-04  +2.85000000000000e+01  +8.50000000000454e+01

      raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +2.74999999999955e+02  +0.00000000000000e+00  +8.24272114532810e-02

j2 perturbed transfer orbit
---------------------------

       sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00723490972948e+03  +9.68259924494168e-04  +2.89460861696500e+01  +2.10862176058786e+01

      raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +3.38913782394121e+02  +0.00000000000000e+00  +8.25317579027424e-02

initial delta-v vector and magnitude

x-component of delta-v             23.689166  meters/second
y-component of delta-v              1.681318  meters/second
z-component of delta-v             49.737090  meters/second

delta-v magnitude                  55.116074  meters/second

final delta-v vector and magnitude

x-component of delta-v             23.555639  meters/second
y-component of delta-v              7.318621  meters/second
z-component of delta-v             50.123150  meters/second

delta-v magnitude                  55.116074  meters/second

transfer time                      56.000000  minutes

final position vector error components and magnitude

x-component of delta-r              0.000010  meters
y-component of delta-r              0.000002  meters
z-component of delta-r              0.000005  meters

delta-r magnitude                   0.000011  meters
```

lambert4_otb.m and lambert4_snopt – J_2-perturbed motion solutions – NLP solution

These two MATLAB applications demonstrate how to solve the J_2-perturbed Earth orbit Lambert problem using a *nonlinear programming* technique. This script can solve both the flyby and rendezvous problems. For the flyby problem, the programs attempt to match all three components of the position vector. For the rendezvous problem, the scripts attempt to match all three components of both the *target* or final position and velocity vectors.

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

```
            program lambert4_snopt

   < j2-perturbed Earth orbit Lambert problem >


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

   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

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

 Nonzero derivs  Jij        27
 Non-constant    Jij's      27     Constant Jij's           0

 SNJAC  EXIT 100 -- finished successfully
 SNJAC  INFO 102 -- Jacobian structure estimated

 Scale option  0


 Nonlinear constraints       6     Linear constraints       1
 Nonlinear variables         6     Linear variables         0
 Jacobian  variables         6     Objective variables      6
 Total constraints           7     Total variables          6

 Itn      0: Feasible linear rows
 Itn      0: PP1.  Minimizing  Norm(x-x0)

 Itn      0: PP1.  Norm(x-x0) approximately minimized  (0.00E+00)


 The user has defined       0   out of      27   first  derivatives

 Itn      0: Hessian set to a scaled identity matrix

 Major Minors     Step   nCon Feasible  Optimal  MeritFunction     nS Penalty
     0      2               1  2.9E+01  1.0E-01  9.4446559E-03                 r
 Itn      2: Hessian set to a scaled identity matrix
     1      0  4.7E-03      2  2.8E+01  3.4E-02  1.4747657E-02        9.2E-06   r
     2      0  1.7E-03      3  2.8E+01  1.0E+00  1.4745801E-02        9.2E-06 s
     3      0  1.0E+00      4  1.6E+00  2.1E-01  7.3172755E-02        1.8E-03
     4      0  1.0E+00      5  1.3E-03  5.1E-04  1.1097988E-01        2.5E-02  m
 Itn      2 -- Central differences invoked.  Small reduced gradient.
     5      0  1.0E+00      6 (5.9E-09)(2.5E-09) 1.1097984E-01        2.5E-02       c
     5      0  1.0E+00      6 (5.9E-09)(2.5E-09) 1.1097984E-01        2.5E-02       c

 SNOPTA EXIT   0 -- finished successfully
 SNOPTA INFO   1 -- optimality conditions satisfied

 Problem name                 matlabMx
 No. of iterations                   2   Objective            1.1097984039E-01
 No. of major iterations             5   Linear    obj. term  0.0000000000E+00
 Penalty parameter           2.487E-02   Nonlinear obj. term  1.1097984039E-01
 User function calls (total)        66   Calls with modes 1,2 (known g)      6
 Calls for forward differencing     42   Calls for central differencing      6
 No. of degenerate steps             0   Percentage                       0.00
 Max x                       6 5.0E-02   Max pi                       7 1.0E+00
 Max Primal infeas           0 0.0E+00   Max Dual infeas              1 2.9E-09
 Nonlinear constraint violn    5.9E-09

 Solution printed on file   9

 Time for MPS input                            0.00 seconds
 Time for solving problem                      0.17 seconds
 Time for solution output                      0.00 seconds
 Time for constraint functions                 0.17 seconds
 Time for objective function                   0.00 seconds


         program lambert4_snopt

< j2-perturbed Earth orbit Lambert problem >

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00000000000000e+03  +0.00000000000000e+00  +2.85000000000000e+01  +0.00000000000000e+00
```

```
       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +0.00000000000000e+00  +0.00000000000000e+00  +8.24199262419658e-02

        rx (km)                ry (km)                rz (km)                rmag (km)
 -1.38918542133544e+03  +7.87846202409766e+03  +0.00000000000000e+00  +8.00000000000000e+03

        vx (kps)               vy (kps)               vz (kps)               vmag (kps)
 -6.10905247177800e+00  -1.07719077733820e+00  +3.36811407992746e+00  +7.05868645918807e+00


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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00723489628510e+03  +9.68258357805992e-04  +2.89460861700782e+01  +2.10862530877801e+01

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +3.38913746912220e+02  +0.00000000000000e+00  +8.25317576948826e-02

        rx (km)                ry (km)                rz (km)                rmag (km)
 -1.38918542133544e+03  +7.87846202409766e+03  +0.00000000000000e+00  +8.00000000000000e+03

        vx (kps)               vy (kps)               vz (kps)               vmag (kps)
 -6.08536330087384e+00  -1.07550945881470e+00  +3.41785116756285e+00  +7.06187465926950e+00


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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00712131217711e+03  +9.65333285299813e-04  +2.89453516511171e+01  +1.47239268486326e+02

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +9.98377817600496e+01  +2.29029908749311e+01  +1.70142259361257e+02  +8.25300016087388e-02

        rx (km)                ry (km)                rz (km)                rmag (km)
 +1.65787953977997e+02  -7.97076711063515e+03  +6.62861993419401e+02  +7.99999999999920e+03

        vx (kps)               vy (kps)               vz (kps)               vmag (kps)
 +6.20553203642311e+00  -1.53599271893434e-01  -3.36706800685960e+00  +7.06182466181491e+00


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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +7.99999999998364e+03  +2.05359252577278e-12  +2.84999999999968e+01  +0.00000000000000e+00

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000017e+02  +1.69999999999933e+02  +1.69999999999933e+02  +8.24199262417130e-02

        rx (km)                ry (km)                rz (km)                rmag (km)
 +1.65787953977997e+02  -7.97076711063515e+03  +6.62861993419401e+02  +7.99999999999920e+03

        vx (kps)               vy (kps)               vz (kps)               vmag (kps)
 +6.22908767828374e+00  -1.46280648236848e-01  -3.31694485893845e+00  +7.05868645918156e+00


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 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         0.00000589  meters
y-component of delta-r        -0.00000104  meters
z-component of delta-r        -0.00000426  meters

delta-r magnitude              0.00000735  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
```

The numerical output from the OTB version of this script with the same initial conditions is as follows;

```
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

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

                                          First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       7    9.444656e-03    2.851e+01    2.054e-15
    1      14    1.143779e-01    1.631e+00    1.011e+00    8.084e-02
    2      21    1.109852e-01    9.047e-04    3.813e-02    2.420e-03
    3      28    1.109798e-01    1.165e-07    1.442e-05    3.798e-06
    4      35    1.109798e-01    9.209e-12    8.892e-07    2.198e-11

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

           program lambert4 - OTB

< j2-perturbed Earth orbit Lambert problem >

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

```
        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00000000000000e+03  +0.00000000000000e+00  +2.85000000000000e+01  +0.00000000000000e+00

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +0.00000000000000e+00  +0.00000000000000e+00  +8.24199262419658e-02

        rx (km)                 ry (km)                rz (km)               rmag (km)
 -1.38918542133544e+03  +7.87846202409766e+03  +0.00000000000000e+00  +8.00000000000000e+03

        vx (kps)                vy (kps)               vz (kps)              vmag (kps)
 -6.10905247177800e+00  -1.07719077733820e+00  +3.36811407992746e+00  +7.05868645918807e+00

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00723490972905e+03  +9.68259924501702e-04  +2.89460861700223e+01  +2.10862176158155e+01

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +3.38913782384184e+02  +0.00000000000000e+00  +8.25317579027358e-02

        rx (km)                 ry (km)                rz (km)               rmag (km)
 -1.38918542133544e+03  +7.87846202409766e+03  +0.00000000000000e+00  +8.00000000000000e+03

        vx (kps)                vy (kps)               vz (kps)              vmag (kps)
 -6.08536330597188e+00  -1.07550945974264e+00  +3.41785117042088e+00  +7.06187466518716e+00

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00712131870563e+03  +9.65333861454100e-04  +2.89453516510395e+01  +1.47239311817426e+02

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +9.98377817600624e+01  +2.29029475438714e+01  +1.70142259361297e+02  +8.25300017096737e-02

        rx (km)                 ry (km)                rz (km)               rmag (km)
 +1.65787953983883e+02  -7.97076711063619e+03  +6.62861993415148e+02  +8.00000000000001e+03

        vx (kps)                vy (kps)               vz (kps)              vmag (kps)
 +6.20553203888767e+00  -1.53599268795383e-01  -3.36706800848446e+00  +7.06182466468797e+00

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

        sma (km)             eccentricity         inclination (deg)       argper (deg)
 +8.00000000000001e+03  +8.32797788973772e-16  +2.85000000000000e+01  +0.00000000000000e+00

       raan (deg)          true anomaly (deg)       arglat (deg)         period (days)
 +1.00000000000000e+02  +1.70000000000000e+02  +1.70000000000000e+02  +8.24199262419659e-02

        rx (km)                 ry (km)                rz (km)               rmag (km)
 +1.65787953983883e+02  -7.97076711063619e+03  +6.62861993415148e+02  +8.00000000000001e+03

        vx (kps)                vy (kps)               vz (kps)              vmag (kps)
 +6.22908767828918e+00  -1.46280648235339e-01  -3.31694485894214e+00  +7.05868645918806e+00

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

x-component of delta-v         23.689166  meters/second
y-component of delta-v          1.681318  meters/second
z-component of delta-v         49.737090  meters/second

delta-v magnitude              55.116074  meters/second

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

x-component of delta-v         23.555639  meters/second
y-component of delta-v          7.318621  meters/second
```

```
z-component of delta-v          50.123150  meters/second

delta-v magnitude               55.863767  meters/second

total delta-v                  110.979841  meters/second

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

x-component of delta-r          0.00000001  meters
y-component of delta-r          0.00000001  meters
z-component of delta-r         -0.00000001  meters

delta-r magnitude               0.00000001  meters

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

x-component of delta-v          0.00000000  meters/second
y-component of delta-v          0.00000000  meters/second
z-component of delta-v         -0.00001490  meters/second

delta-v magnitude               0.00001490  meters/second

transfer time                    56.000000  minutes
```

The final position and velocity vector errors indicate how well the problem has been solved.

Lambert MATLAB functions

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

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

The algorithm used in this MATLAB function 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.

This two-body Lambert function has the following syntax.

```
function [vi, vf] = lambert_gooding(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
% References

%  R. H. Gooding, Technical Report 88027
%  On the Solution of Lambert's Orbital Boundary-Value Problem,
%  Royal Aerospace Establishment, April 1988

%  R. H. Gooding, Technical Memo SPACE 378
%  A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem
%  Royal Aerospace Establishment, March 1991
```

lambert_gedeon.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

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

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:

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

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

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

The iteration for z is

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

This Lambert function has the following syntax.

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

% solve Lambert's orbital two point boundary value problem

% Geza Gedeon's method

% input

%  ri     = initial position vector (kilometers)
%  rf     = final 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.

Technical discussion

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

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

the time equation becomes

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

From the elliptic relationships given by

$$r = a\left(1 - e\cos E\right) \qquad x = a\left(\cos E - e\right) \qquad y = a\sin E\sqrt{1-e^2}$$

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

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

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.

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 computational 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

$$\Phi_{11} = \begin{bmatrix} \dfrac{\partial \mathbf{r}}{\partial \mathbf{r}_0} \end{bmatrix} = \begin{bmatrix} \partial x/\partial x_0 & \partial x/\partial y_0 & \partial x/\partial z_0 \\ \partial y/\partial x_0 & \partial y/\partial y_0 & \partial y/\partial z_0 \\ \partial z/\partial x_0 & \partial z/\partial y_0 & \partial z/\partial z_0 \end{bmatrix}$$

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 as a function of time 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:

$$\mathbf{p}(t_0) = \mathbf{p}_0 = \frac{\Delta\mathbf{V}_0}{|\Delta\mathbf{V}_0|} \qquad \mathbf{p}(t_f) = \mathbf{p}_f = \frac{\Delta\mathbf{V}_f}{|\Delta\mathbf{V}_f|}$$

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.

Finally, the scalar magnitude of the derivative of the primer vector can be determined from

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

Algorithm Resources

An Introduction to the Mathematics and Methods of Astrodynamics, Richard H. Battin, AIAA Education Series, 1987.

Analytical Mechanics of Space Systems, Hanspeter Schaub and John L. Junkins, AIAA Education Series, 2003.

Orbital Mechanics, Vladimir A. Chobotov, AIAA Education Series, 2002.

Modern Astrodynamics, Victor R. Bond and Mark C. Allman, Princeton University Press, 1996.

Spacecraft Mission Design, Charles D. Brown, AIAA Education Series, 1992.

"Partial derivatives of the Lambert problem", Nitin Arora, Ryan Russell and Nathan Strange, AIAA Space 2014, San Diego, California, August 5-7, 2014.

Appendix A

SNOPT algorithm implementation

This appendix provides details about the MATLAB code of the main `lambert4_snopt` 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.*

Information about MATLAB versions of SNOPT for several computer platforms can be found at Professor Philip Gill's web site which is located at https://ccom.ucsd.edu/~optimizers.

The SNOPT algorithm requires an initial guess for the control variables. In this implementation, 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 delta-v initial guesses (**xg**) as follows:

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

xlwr(1:3) = min(-1.1 * norm(xg(1:3)), - 0.75);

xupr(1:3) = max(+1.1 * norm(xg(1:3)), + 0.75);

if (otype == 2)

    % rendezvous

    xlwr(4:6) = min(-1.1 * norm(xg(4:6)), - 0.75);

    xupr(4:6) = max(+1.1 * norm(xg(4:6)), + 0.75);

end
```

Notice that the lower and upper bounds are allowed to change each component of the delta-v velocity vectors by either 110% of the initial guess or 0.75 kilometers/second.

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

```
% bounds on objective function

flwr(1) = 0.0;
fupr(1) = +Inf;
```

Finally, the SNOPT algorithm also requires the following position vector *equality* constraints and velocity vector equality constraints for a rendezvous mission.

```
% enforce final position vector equality constraints (kilometers)

flwr(2:4) = 0.0;
```

```
fupr(2:4) = 0.0;

if (otype == 2)

    % rendezvous - enforce final velocity vector equality constraints
    % (kilometers/second)

    flwr(5:7) = 0.0;
    fupr(5:7) = 0.0;

end
```

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

```
[x, f, inform, xmul, fmul] = snopt(xg, xlwr, xupr, xmul, xstate, ...
    flwr, fupr, fmul, fstate, 'tpbvp_snopt');
```

where **'tpbvp_snopt** 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 computational steps performed by this function are as follows;

(1) load the current initial state vector of the transfer orbit

(2) using this state vector, integrate the J_2-equations of motion to the user-specified time-of-flight

(3) from the state vector predicted in step (2), calculate the current magnitude of the initial delta-v velocity vector and for a rendezvous problem the final delta-v velocity vector

(4) for a flyby-only trajectory set the objective function to the magnitude of the initial delta-v, and for rendezvous orbits the sum of the initial and final delta-v magnitudes

(5) calculate the current final position vector equality constraints. For a rendezvous mission, also calculate the final velocity vector equality constraints

The following is the MATLAB source code for this function.

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

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

% input

%  x = current delta-v vector (kilometers/second)

% output

%  f(1) = objective function (delta-v magnitude, kilometers/second)
%  f(2) = rx equality constraint delta (kilometers)
%  f(3) = ry equality constraint delta (kilometers)
%  f(4) = rz equality constraint delta (kilometers)
%  f(5) = vx equality constraint delta (kilometers/seconds)
%  f(6) = vy equality constraint delta (kilometers/seconds)
```

```
%  f(7) = vz equality constraint delta (kilometers/seconds)
% Orbital Mechanics with MATLAB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global otype neq tetol
global ri vi tof rtarget vtarget drf dvf
if (otype == 1)
   f = zeros(4, 1);
else
   f = zeros(7, 1);
end
% load current state vector of transfer orbit
xi(1:3) = ri(1:3);
xi(4:6) = vi(1:3) + x(1:3);
% initial guess for step size (seconds)
h = 10.0;
% initial time (seconds)
ti = 0.0;
% final time (seconds)
tf = tof;
% integrate j2 equations of motion
xf = rkf78 ('j2eqm', neq, ti, tf, h, tetol, xi);
% objective function (delta-v magnitude, kilometers/second)
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 (kilometers)

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

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

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

if (otype == 2)

   % final velocity vector (kilometers/second)

   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 equality constraints (kilometers/second)

   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

% no derivatives
```

```
g = [];
```

Here is the MATLAB source code which evaluates the first-order equations of motion.

```
function ydot = j2eqm (~, y)

% first order equations of orbital motion

% j2 gravitational acceleration

% input

%  t = simulation time (seconds)
%  y = state vector (kilometers and kilometers/second)

% output

%  ydot = integration vector (kilometers/second/second)

% Orbital Mechanics with MATLAB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu req j2

r2 = y(1) * y(1) + y(2) * y(2) + y(3) * y(3);

r1 = sqrt(r2);

r3 = r2 * r1;

r5 = r2 * r3;

d1 = -1.5 * j2 * req * req * mu / r5;

d2 = 1.0 - 5.0 * y(3) * y(3) / r2;

% integration vector

ydot(1) = y(4);

ydot(2) = y(5);

ydot(3) = y(6);

ydot(4) = y(1) * (d1 * d2 - mu / r3);

ydot(5) = y(2) * (d1 * d2 - mu / r3);

ydot(6) = y(3) * (d1 * (d2 + 2) - mu / r3);
```

Appendix B

OTB algorithm implementation

This appendix provides details about the part of the `lambert4_otb` script that solves this nonlinear programming (NLP) problem using algorithms from the Mathworks™ Optimization Toolbox. 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.*

Array pre-allocation for this script depends on the transfer type defined by the user and held in the parameter **`otype`**.

```
% pre-allocate lower and upper bounds vectors

if (otype == 1)

    % flyby

    xlwr = zeros(3, 1);

    xupr = zeros(3, 1);

else

    % rendezvous

    xlwr = zeros(6, 1);

    xupr = zeros(6, 1);

end

% 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)), - 0.75);

    xupr(i) = max(+1.1 * norm(xg(1:3)), + 0.75);

end

if (otype == 2)

    % rendezvous

    for i = 4:1:6

        xlwr(i) = min(-1.1 * norm(xg(4:6)), - 0.75);
```

```
            xupr(i) = max(+1.1 * norm(xg(4:6)), + 0.75);

        end

    end
```

The source code for the algorithm options and call to the **fmincon** algorithm of the OTB is

```
options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', 'interior
                        point', 'MaxFunctionEvaluations', 5000);

[x, fval] = fmincon('tpbvp_objective_otb', xg, [], [], [], [], ...
    xlwr, xupr, 'tpbvp_constraints_otb', options);
```

In the calling arguments, **'tpbvp_objective_otb** is the MATLAB function that calculates the current numerical value of the objective function depending on the value of **otype**.

```
function f = tpbvp_objective_otb(x)

% OTB two point boundary value objective function

% input

%  x = current delta-v vector (kilometers/second)

% output

%  f = objective function (delta-v magnitude, kilometers/second)

% Orbital Mechanics with MATLAB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global otype

% objective function (delta-v magnitude, kilometers/second)

if (otype == 1)

   % initial delta-v (flyby)

   f = norm(x);

else

   % total delta-v (rendezvous)

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

end
```

The following MATLAB function computes the state constraints required by the **fmincon** algorithm.

```
function [c, ceq] = tpbvp_constraints_otb(x)

% OTB two point boundary value state vector constraints

% input

%  x = current delta-v vector (kilometers/second)

% output

%  ceq(1) = x-component of position constraint delta (kilometers)
%  ceq(2) = y-component of position constraint delta (kilometers)
%  ceq(3) = z-component of position constraint delta (kilometers)
%  ceq(4) = x-component of velocity constraint delta (kilometers/second)
%  ceq(5) = y-component of velocity constraint delta (kilometers/second)
%  ceq(6) = z-component of velocity constraint delta (kilometers/second)

% Orbital Mechanics with MATLAB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global otype neq tetol

global ri vi tof rtarget vtarget drf dvf

% load current initial 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);

% final position vector equality constraints (kilometers)
```

```
ceq(1) = rtarget(1) - xf(1);

ceq(2) = rtarget(2) - xf(2);

ceq(3) = rtarget(3) - xf(3);

if (otype == 2)

    % final velocity vector (kilometers/second)

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

end

if (otype == 2)

    % final velocity vector constraints (kilometers/second)

    ceq(4) = vtarget(1) - vf(1);

    ceq(5) = vtarget(2) - vf(2);

    ceq(6) = vtarget(3) - vf(3);

end

% save state vector deltas for print summary

for i = 1:1:3

    % flyby

    drf(i) = ceq(i);

    if (otype == 2)

        % rendezvous

        dvf(i) = ceq(i + 3);

    end

end

% no inequality constraints

c = [];
```