Lunar Free-Return Trajectory Analysis with MATLAB

This document is the user's guide for a MATLAB script named `free_return` that can be used to design two-dimensional lunar free-return trajectories. The system dynamics are modeled as a *simplified* circular-restricted three-body problem (CRTBP). Additional modeling assumptions are;

- the massless third body is subject only to the point-mass gravity of the earth and moon
- the moon is subject only to the point mass gravity of the earth
- the moon's orbit is circular and all motion lies in the moon's orbital plane
- departure from the earth park orbit is in the counterclockwise direction
- trans-lunar injection (TLI) occurs *impulsively* from a circular earth park orbit
- the TLI departure maneuver is applied tangential to the earth park orbit
- the earth orbit insertion (EOI) impulsive maneuver is applied tangential to the return trajectory
- the lunar flyby is a no maneuver, ballistic trajectory

The algorithm implemented in this MATLAB script is based on the technical report "Optimal Free-Return Trajectories for Moon and Mars Missions", by A. Miele, T. Wang and S. Mancuso, *The Journal of the Astronautical Sciences*, Vol. 48, Nos. 2 and 3, April-September 2000, pp. 183-206.

The `free_return` script uses the SNOPT nonlinear programming (NLP) method for optimizing the lunar free-return problem implemented in this script. MATLAB versions of SNOPT mex and support m-files for several computer platforms can be found at Professor Philip Gill's web site which is located at http://scicomp.ucsd.edu/~peg/. Professor Gill's web site also includes a PDF version of the SNOPT software user's guide.

Input data file

The `free_return` MATLAB script is "data-driven" by a simple text file created by the user. This section describes a typical input data file. In the following discussion the actual input file contents are in *courier* font and all explanations are in *times italic* font.

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input data.

The first four lines of any input file are reserved for user comments. These lines are ignored by the software. However the input file must begin with four and only four initial text lines.

```
================================
input file for free_return.m
free_return1.in  January 17, 2014
================================
```

The first four inputs define the gravitational constants and radius of the earth and moon.

```
gravitational constant of the earth (kilometers^3/second^2)
398600.4415
```

```
gravitational constant of the moon (kilometers^3/second^2)
4902.8

radius of the earth (kilometers)
6378.14

radius of the moon (kilometers)
1738.0
```

The next two inputs specify the (constant) distance from the earth to the moon and the radius of the lunar sphere-of-influence (SOI).

```
distance from the earth to the moon (kilometers)
384400.0

lunar sphere-of-influence radius (kilometers)
64000.0
```

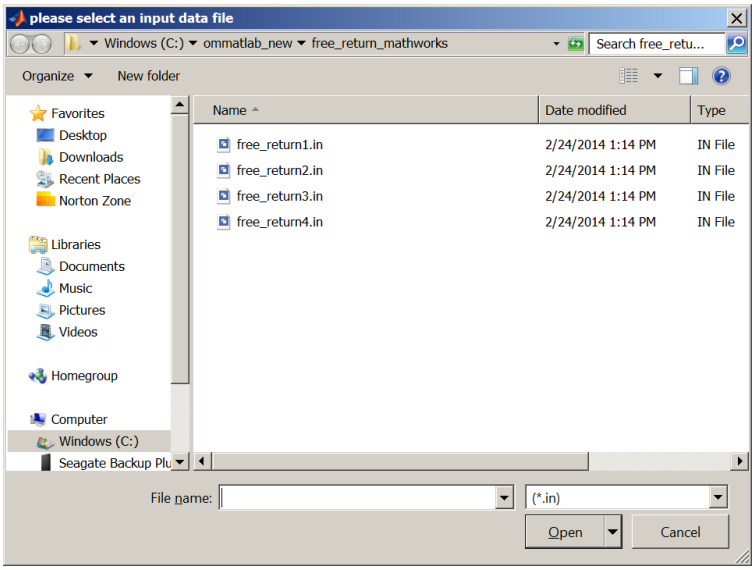
The next two inputs define the altitudes of the circular earth park orbit and the lunar close approach.

```
altitude of the circular earth park orbit (kilometers)
463.0

altitude of lunar close approach (kilometers)
100.0
```

The final two numeric inputs specify the user's initial guess for the geocentric angular location of the TLI maneuver and the scalar magnitude of the TLI delta-v.

```
initial guess for geocentric angular location of the tli delta-v (degrees)
227.5

initial guess for tli delta-v magnitude (kilometers/second)
3.093
```

After typing `free_return` in the MATLAB command window, the software will ask you for the name of the input data file with a display similar to



The file type defaults to names with a `*.in` filename extension. However, you can select any `free_return` compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

Script example and trajectory graphics

The following is the script output for the example given earlier. The first part of the output summarize the SNOPT algorithm results.

```
Lunar free return trajectory analysis
=====================================

please wait, solving optimization problem ...


 Nonlinear constraints       2     Linear constraints       1
 Nonlinear variables         2     Linear variables         0
 Jacobian  variables         2     Objective variables      0
 Total constraints           3     Total variables          2


 The user has defined       0   out of       4   first  derivatives

 Major Minors     Step   nCon Feasible  Optimal  MeritFunction    nS Penalty
     0      1              1  9.5E+00  7.8E-05  3.0930000E+00                 r
     1      0  1.0E+00      2  5.2E-01  6.9E-06  3.0929029E+00               n r
     2      0  1.0E+00      3  4.6E-03  6.5E-03  3.0928921E+00               sm
     2      0  1.0E+00      3  4.6E-03  6.5E-03  3.0928922E+00       7.8E-05 sm    c
     3      0  1.0E+00      4  8.4E-06  5.3E-06  3.0928922E+00       8.1E-05  m    c
     4      0  1.0E+00      5 (7.4E-08)(4.6E-08) 3.0928922E+00       2.9E-03 n     c

 SNOPTA EXIT   0 -- finished successfully
 SNOPTA INFO   1 -- optimality conditions satisfied
 Problem name
 No. of iterations                    1   Objective value      3.0928921545E+00
 No. of major iterations              4   Linear objective     3.0928921545E+00
 Penalty parameter            2.919E-03   Nonlinear objective  0.0000000000E+00
 No. of calls to funobj              39   No. of calls to funcon              39
 Calls with modes 1,2 (known g)       5   Calls with modes 1,2 (known g)       5
 Calls for forward differencing       6   Calls for forward differencing       6
 Calls for central differencing      12   Calls for central differencing      12
 No. of degenerate steps              0   Percentage                         .00
 Max x                        1 4.0E+00   Max pi                        3 1.0E+00
 Max Primal infeas            0 0.0E+00   Max Dual infeas               2 9.3E-07
 Nonlinear constraint violn     4.4E-07

Solution printed on file   9


TLI delta-v                 3092.89215449 meters/second

one way time of flight        68.86984088 hours
                               2.86957670 days

round trip time of flight    137.73968176 hours
                               5.73915341 days


geocentric orbital elements and state vector at Earth departure
---------------------------------------------------------------

        sma (km)             eccentricity        inclination (deg)         argper (deg)
 +2.68940565889181e+05  +9.74562632537856e-01  +0.00000000000000e+00  +2.27464212649094e+02

       raan (deg)        true anomaly (deg)         arglat (deg)          period (hrs)
 +0.00000000000000e+00  +0.00000000000000e+00  +2.27464212649094e+02  +3.85560869878036e+02
```

```
        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 -4.62495669538951e+03  -5.04092968264503e+03  +0.00000000000000e+00  +6.84114000000000e+03

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 +7.90355112502884e+00  -7.25135718891348e+00  +0.00000000000000e+00  +1.07260571258572e+01

altitude                  463.00000000 kilometers

flight path angle          -0.00000000 degrees


selenocentric orbital elements and state vector at lunar flyby
----------------------------------------------------------------

        sma (km)              eccentricity        inclination (deg)         argper (deg)
 -4.12030624842125e+03  +1.44608334663421e+00  +1.80000000000000e+02  +3.22368803066883e+02

       raan (deg)          true anomaly (deg)         arglat (deg)
 +0.00000000000000e+00  +4.07110999227330e-13  +3.22368803066884e+02

        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 +1.45561753333692e+03  +1.12223954586784e+03  +0.00000000000000e+00  +1.83800000045362e+03

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 +1.55964235903821e+00  -2.02295737296920e+00  +0.00000000000000e+00  +2.55437679698135e+00

altitude                  100.00000045 kilometers

flight path angle           0.00000000 degrees

mission time at flyby      68.86984088 hours
                            2.86957670 days


geocentric orbital elements and state vector at Earth arrival
---------------------------------------------------------------

        sma (km)              eccentricity        inclination (deg)         argper (deg)
 +2.68940565843722e+05  +9.74562632622988e-01  +0.00000000000000e+00  +2.07798181207688e+02

       raan (deg)         true anomaly (deg)         arglat (deg)          period (hrs)
 +0.00000000000000e+00  +3.59999995751323e+02  +2.07798176959011e+02  +3.85560869780278e+02

        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 -6.05164378768879e+03  -3.19042373948091e+03  +0.00000000000000e+00  +6.84113997594814e+03

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 +5.00218820872306e+00  -9.48822507116623e+00  +0.00000000000000e+00  +1.07260571449436e+01

altitude                  462.99997595 kilometers

flight path angle          -0.00000210 degrees

circularization deltav   3092.89216016 meters/second


image trajectory mission constraints at lunar closest approach
---------------------------------------------------------------

y-component of relative position vector          0.00044392 meters

x-component of relative velocity vector         -0.00000000 meters/second

moon-third body geocentric separation angle      0.00000000 degrees

geocentric inertial flight path angle            0.00000002 degrees
```

Please see the Technical Discussion section for an explanation of these last four items.

The simulation summary screen display contains the following information:

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (hrs) = orbital period in hours

rx (km) = x-component of the third body position vector in kilometers

ry (km) = y-component of the third body position vector in kilometers

rz (km) = z-component of the third body position vector in kilometers

rmag (km) = scalar magnitude of the third body position vector in kilometers

vx (km/sec) = x-component of the third body velocity vector in kilometers per second

vy (km/sec) = y-component of the third body velocity vector in kilometers per second

vz (km/sec) = z-component of the third body velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the third body velocity vector in kilometers per second

All altitudes are with respect to a spherical earth or moon. The `flight path angle` is the angle of the velocity vector relative to the "local" horizontal, measured positive above and negative below the horizontal. The `circularization delta-v` is the scalar magnitude of the maneuver required to create a circular orbit at earth closest approach on the return trajectory.

The `moon-third body geocentric separation angle` is the geocentric separation angle between the third body and the moon at lunar closest approach.

After the main calculations are finished, the `free_return` script will ask if you would like to create graphic displays of the solution with the following prompt;

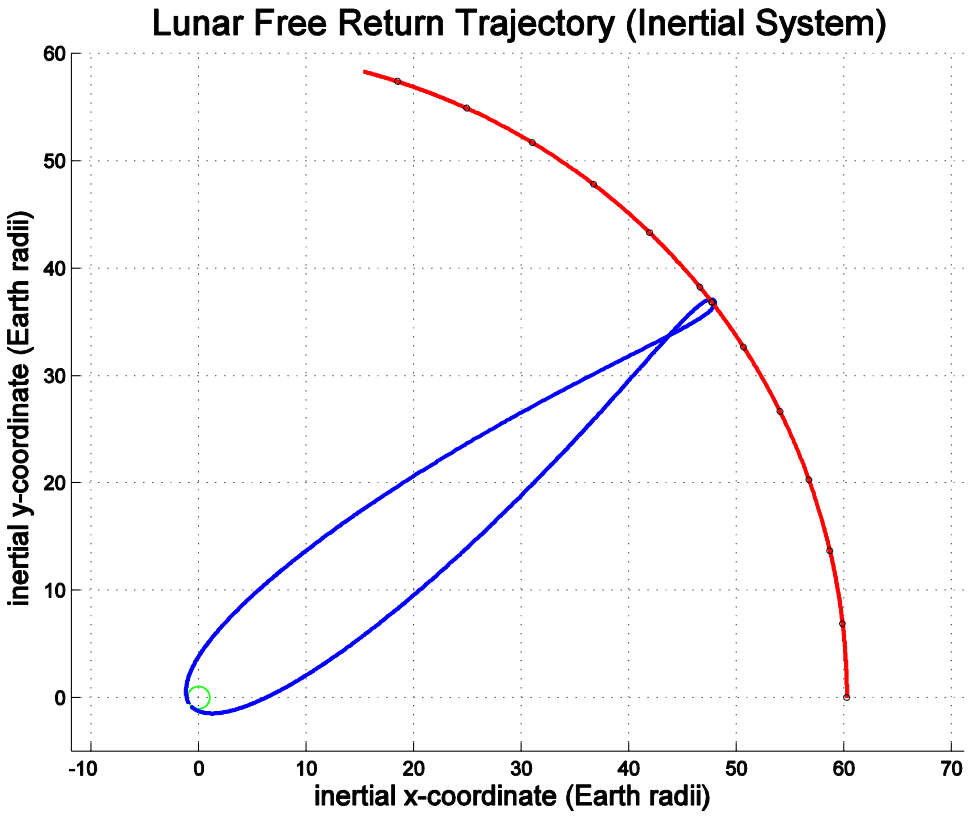
```
would you like to display trajectory graphics (y = yes, n = no)
?
```

A user response of `yes` will create several graphic displays which are described in the next section. Each display will also be saved to a disk file in encapsulated Postscript format with a MATLAB command similar to the following;

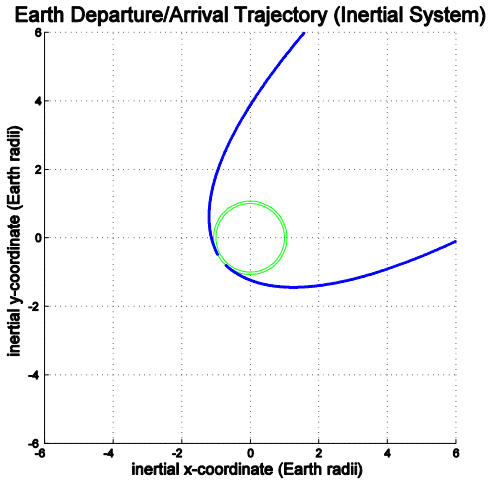
```
print -depsc -tiff -r300 free_return1.eps
```

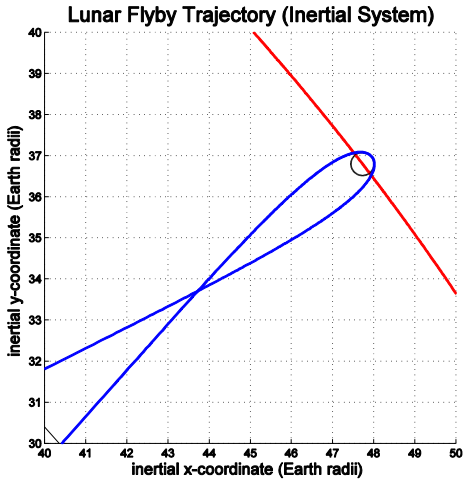
The following are typical graphic displays created by this script.

This first plot illustrates the entire free-return trajectory in a geocentric inertial coordinate system. The surface of the earth and the circular park orbit are green, the third body trajectory is blue, and the circular orbit of the moon is red. The moon in its orbit is marked every twelve hours. Note that all graphics are to scale with the fundamental measurement unit being the radius of the earth.

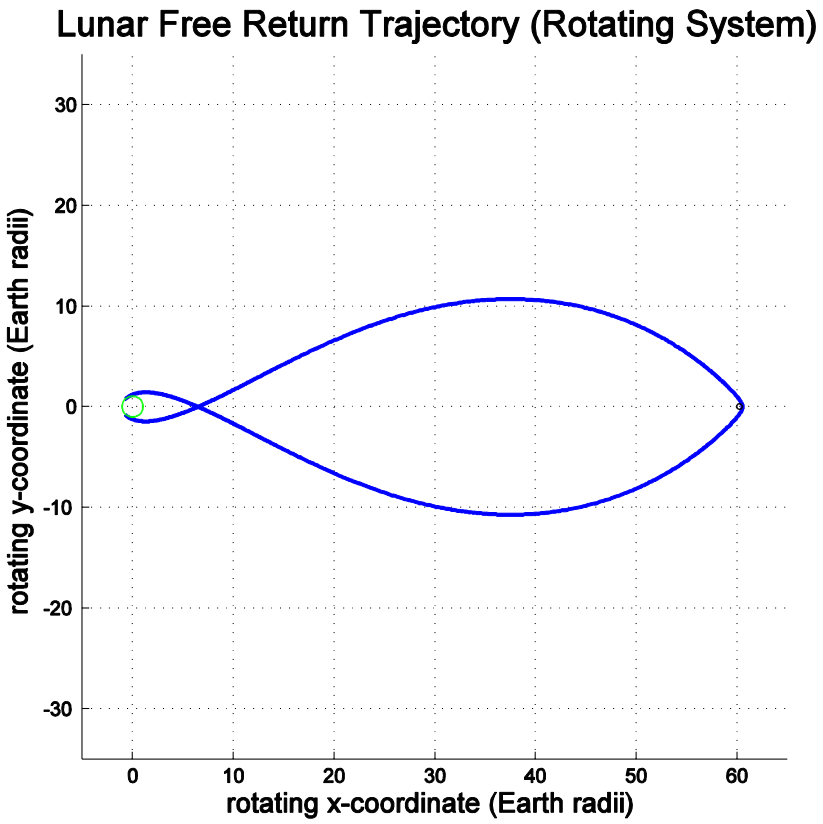


The next two plots depict the earth departure and arrival trajectory and the lunar flyby trajectory. Both plots are relative to the inertial coordinate system. The departure/arrival trajectory is geocentric and the flyby trajectory is selenocentric or "moon-centered". The surface of the moon and the lunar circular park orbit are black. Notice that the third body travels around the moon in a clockwise direction.

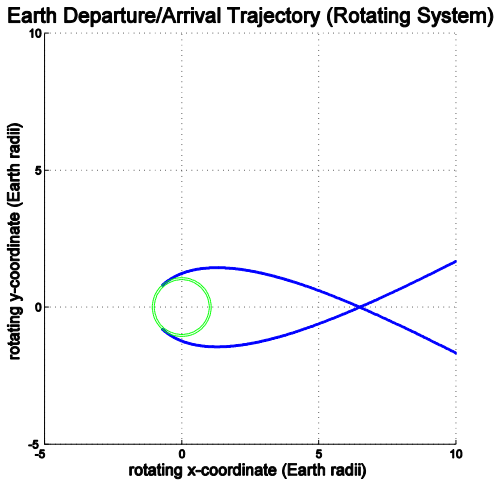


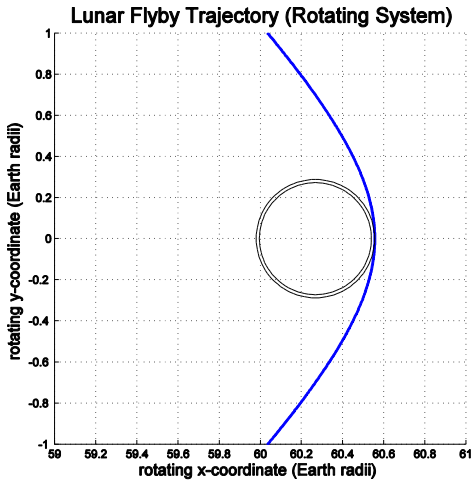


This next plot illustrates the entire free-return trajectory in a geocentric rotating coordinate system. As before, all graphics are to scale with the fundamental measurement unit being the earth radius.

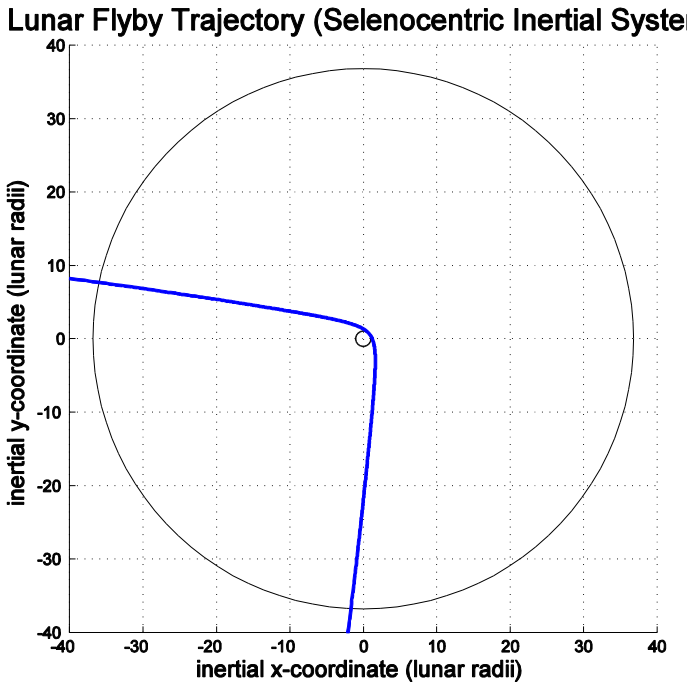


The next two "zoomed-in" plots illustrate the earth departure and arrival trajectory and the lunar flyby trajectory. Both plots are relative to the rotating coordinate system.





The final plot is the hyperbolic lunar flyby trajectory in a selenocentric inertial coordinate system. The fundamental distance unit for this display is the radius of the moon. The large black circle represents the lunar sphere-of-influence with a radius equal to 64,000 kilometers (~36.8 lunar radii).



The field of view of each trajectory display is set with a MATLAB command similar to

```
axis([-5.0 60.0 -5.0 60.0]);
```

The syntax of this MATLAB command is $\text{axis}\left[x_{\min}\ x_{\max}\ y_{\min}\ y_{\max}\right]$ where the minimum and maximum x-coordinates are $x_{\min}$ and $x_{\max}$. The minimum and maximum y-coordinates are $y_{\min}$ and $y_{\max}$. The user can edit any or all of the default axis commands in the `free_return` script to change the geocentric and/or selenocentric views.

Technical Discussion

This section provides additional details about the numerical algorithms implemented in this computer program. It includes a discussion about coordinate systems, equations of motion, and verification of the "image trajectory" solution. A brief outline of trajectory optimization is also provided.

Coordinate systems

The origin of the inertial coordinate system is the center of the earth. The x-axis points in the direction of the moon at the initial time and the y-axis is perpendicular to the x-axis. Since the moon's orbit around the earth is circular, the y-axis is also parallel to the initial inertial velocity of the moon.

The origin of the rotating coordinate is also the center of the earth. The x-axis of this system points at the instantaneous location of the moon. It is also parallel to the initial x-axis of the inertial system. The y-axis is perpendicular to the instantaneous x-axis. The axes of this system rotate at constant angular speed relative to the axes of the inertial system. In this rotating coordinate system, the moon appears at rest relative to the earth.

In the graphic images described earlier, the inertial x-axis is the horizontal line positive to the right.

Equations of motion

The geocentric second-order differential equations of motion of the third body in a simplified two-dimensional circular-restricted system are given by

$$\ddot{x}_{E-B} = -\left(\frac{\mu_E}{r_{E-B}^3}\right)x_{E-B} - \left(\frac{\mu_M}{r_{M-B}^3}\right)\left(x_{E-B} - x_{E-M}\right)$$

$$\ddot{y}_{E-B} = -\left(\frac{\mu_E}{r_{E-B}^3}\right)y_{E-B} - \left(\frac{\mu_M}{r_{M-B}^3}\right)\left(y_{E-B} - y_{E-M}\right)$$

where the subscript *E-B* denotes the location of the third body relative to the earth, *M-B* is the location of the third body relative to the moon, and *E-M* is the location of the moon relative to the earth. In these two equations, μ_E and μ_M are the gravitational constants of the earth and moon, respectively.

The geocentric and selenocentric (moon-centered) distances of the third body are determined from the *x* and *y* components of the position vector according to

$$r_{E-B} = \sqrt{x_{E-B}^2 + y_{E-B}^2} \qquad r_{M-B} = \sqrt{x_{M-B}^2 + y_{M-B}^2}$$

The selenocentric *x* and *y* coordinates of the third body are determined from

$$x_{M-B} = x_{E-B} - x_{E-M} \qquad y_{M-B} = y_{E-B} - y_{E-M}$$

The geocentric inertial *x* and *y* coordinates of the moon are computed from

$$x_{E-M} = r_{E-M}\cos(\omega t) \qquad y_{E-M} = r_{E-M}\sin(\omega t)$$

where *t* is the elapsed time since the TLI departure maneuver. In these equations, ω is the rotation rate or mean motion of the moon in its circular orbit around the earth which is given by $\omega = \sqrt{\mu_E / r_{E-M}^3}$. The velocity of the moon in its orbit is $V_M = \sqrt{\mu_E / r_{E-M}}$.

Using the method of *order reduction*, we can define the following

$$y(1) = x_{E-B} \quad y(2) = y_{E-B} \quad y(3) = \dot{x}_{E-B} \quad y(4) = \dot{y}_{E-B}$$

and create the equivalent first-order system of differential equations given by

$$\dot{y}(1) = \dot{x}_{E-B} \qquad \dot{y}(2) = \dot{y}_{E-B}$$

$$\dot{y}(3) = -\left(\frac{\mu_E}{r_{E-B}^3}\right)x_{E-B} - \left(\frac{\mu_M}{r_{M-B}^3}\right)\left(x_{E-B} - x_{E-M}\right)$$

$$\dot{y}(4) = -\left(\frac{\mu_E}{r_{E-B}^3}\right)y_{E-B} - \left(\frac{\mu_M}{r_{M-B}^3}\right)\left(y_{E-B} - y_{E-M}\right)$$

The x- and y-coordinates of the third body in the rotating system are given by

$$x_{E-B}^{R} = x_{E-B}\cos(\omega t) + y_{E-B}\sin(\omega t)$$

$$y_{E-B}^{R} = x_{E-B}\sin(\omega t) + y_{E-B}\cos(\omega t)$$

Predicting closest approach to the moon

The mission elapsed time at which the third body reaches closest approach to the moon is calculated using the event prediction capability of the MATLAB `ode45` algorithm. During the numerical integration of the third body's geocentric equations of motion, the `ode45` numerical method searches for the time at which the sine of the flight path angle *with respect to the moon* is nearly zero.

Close approach is predicted with the following *mission constraint*

$$\sin\gamma = \left(\frac{\mathbf{r}\bullet\mathbf{v}}{|\mathbf{r}\bullet\mathbf{v}|}\right)$$

where **r** and **v** are the moon-centered or selenocentric position and velocity vectors, respectively.

The following is the MATLAB source code that determines the time and trajectory conditions at closest approach to the moon.

```
% set up options for ode45

options = odeset('RelTol', 1.0e-10, 'AbsTol', 1.0e-10, 'Events', @fpa_event);

% solve for lunar closest approach conditions

tend = 7.0 * 86400.0;

[t, ysol, tevent, yevent, ie] = ode45(@free_return_eqm, [0 tend], [ri vi], options);
```

Note that the numerical values for `RelTol` and `AbsTol` determine how well the algorithm integrates the equations of motion.

The following is the MATLAB source code which calculates the *sine* of the selenocentric flight path angle required by the event prediction code.

```
function [value, isterminal, direction] = fpa_event(t, y)

% selenocentric flight path angle event function

% input

%  t = simulation time (seconds)
%  y = third body geocentric state vector (km, km/sec)

% output

%  value = sine of selenocentric flight path angle

% Orbital Mechanics with MATLAB
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global distance_e2m vlc_moon omega_moon
% inertial geocentric lunar position vector (kilometers)
theta_moon = omega_moon * t;
rmoon(1) = distance_e2m * cos(theta_moon);
rmoon(2) = distance_e2m * sin(theta_moon);
% inertial geocentric lunar velocity vector (kilometers/second)
vmoon(1) = -vlc_moon * sin(theta_moon);
vmoon(2) = vlc_moon * cos(theta_moon);
% form the selenocentric third body position and velocity
rm2sc(1) = y(1) - rmoon(1);
rm2sc(2) = y(2) - rmoon(2);
vm2sc(1) = y(3) - vmoon(1);
vm2sc(2) = y(4) - vmoon(2);
% sine of the selenocentric flight path angle
value = rm2sc * vm2sc' / (norm(rm2sc) * norm(vm2sc));
isterminal = 1;
direction =  [];
```

Nonlinear programming problem

A trajectory optimization problem can be described by a system of *dynamic variables*

$$\mathbf{z} = \begin{bmatrix} \mathbf{y}(t) \\ \mathbf{u}(t) \end{bmatrix}$$

consisting of the *state variables* $\mathbf{y}$ and the *control variables* $\mathbf{u}$ for any time *t*. In this discussion vectors are denoted in bold.

The system dynamics are defined by a vector system of ordinary differential equations called the *state equations* that can be represented as follows:

$$\dot{\mathbf{y}} = \frac{d\,\mathbf{y}}{dt} = \mathbf{f}\left[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t\right]$$

where **p** is a vector of problem *parameters* that is not time dependent.

The initial dynamic variables at time t_0 are defined by $\boldsymbol{\psi}_0 \equiv \boldsymbol{\psi}\left[\mathbf{y}(t_0), \mathbf{u}(t_0), t_0\right]$ and the terminal conditions at the final time t_f are defined by $\boldsymbol{\psi}_f \equiv \boldsymbol{\psi}\left[\mathbf{y}(t_f), \mathbf{u}(t_f), t_f\right]$. These conditions are called the *boundary values* of the trajectory problem.

The problem may also be subject to *path constraints* of the form $\mathbf{g}\left[\mathbf{y}(t), \mathbf{u}(t), t\right] = 0$.

For any mission time *t* there are also simple bounds on the state variables

$$\mathbf{y}_l \le \mathbf{y}(t) \le \mathbf{y}_u$$

the control variables

$$\mathbf{u}_l \le \mathbf{u}(t) \le \mathbf{u}_u$$

and the problem parameters

$$\mathbf{p}_l \le \mathbf{p}(t) \le \mathbf{p}_u$$

The basic nonlinear programming problem (NLP is to determine the control vector history and problem parameters that minimize the scalar performance index or objective function given by

$$J = \phi\left[\mathbf{y}(t_0), t_0, \mathbf{y}(t_f), t_f, \mathbf{p}\right]$$

while satisfying all the user-defined mission constraints.

In this MATLAB script, the control variables consist of the geocentric angular location of the TLI maneuver and the scalar magnitude of this maneuver. The objective function or performance index is the scalar magnitude of the TLI delta-v.

The geocentric angular location of the TLI maneuver during the optimization is bounded according to

$$\theta_g - 10^\circ \le \theta \le \theta_g + 10^\circ$$

where θ_g corresponds to the user's initial guess in degrees. Note that this angle is also equal to the argument of latitude of the TLI maneuver at earth departure. It is measured positive counterclockwise relative to the x-axis of the inertial coordinate system.

The bounds on the scalar value of TLI delta-v are given by

$$\Delta V_g - 100 \le \Delta V \le \Delta V_g + 100$$

where ΔV_g is the initial guess for the scalar magnitude of the TLI delta-v maneuver in meters per second. Values for both the angular location and maneuver magnitude initial guesses are taken from the technical paper mentioned at the being of this document.

The additional mission constraints for this problem are the user-defined lunar radius at closest approach and the y-component of the third body velocity in the geocentric rotating coordinate system. The first constraint ensures that the user's lunar flyby altitude is satisfied. The second constraint ensures that at the moment of lunar closest approach, the third body is exactly on the line from the earth to the moon in either the inertial or rotating coordinate systems.

These are implemented as equality constraints according to

$$r_{fb}^{p} / r_M - r_{fb}^{u} / r_M = 0$$

$$y_{E-B}^{R} = 0$$

In the first equation, r_{fb}^{p} represents the radius at the lunar flyby *predicted* by the software and r_{fb}^{u} is the user-defined flyby radius. Notice that both values are normalized by the radius of the moon (r_M) for improved problem scaling.

Notice that the flight path angle constraint described earlier indirectly satisfies the *image trajectory* mission constraint that the x-component of third body velocity in the rotating system is zero at the moment of lunar closest approach.

The following MATLAB source code illustrates how the main script interacts with the SNOPT nonlinear programming function. This code provides initial guesses (`xg`) and lower and upper bounds (`xlwr` and `xupr`) for the control variables (`theta_tli` and `deltav_tli`) and the objective function (`flow` and `fupp`).

```
% initial guesses

xg(1) = dtr * theta_tli;

xg(2) = deltav_tli;

xg = xg';

% lower and upper bounds for TLI theta (radians)

xlwr(1) = xg(1) - 10.0 * dtr;

xupr(1) = xg(1) + 10.0 * dtr;

% lower and upper bounds for TLI delta-v (meters/second)

xlwr(2) = xg(2) - 0.100;

xupr(2) = xg(2) + 0.100;

xlwr = xlwr';

xupr = xupr';

% bounds on objective function

flow(1) = 0.0d0;

fupp(1) = +Inf;

% bounds on selenocentric close approach (normalized) radius constraint
```

```
flow(2) = radius_lmo / radius_moon;

fupp(2) = radius_lmo / radius_moon;

% bounds on selenocentric relative position constraint (kilometers)

flow(3) = 0.0;

fupp(3) = 0.0;

flow = flow';

fupp = fupp';

snscreen on;

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

% extract solution

theta_tli = x(1);

deltav_tli = x(2);
```

In the call to the SNOPT algorithm, the `free_return_shoot` MATLAB function implements a simple *shooting method* that predicts lunar closest approach while solving the mission constraint equations.

The following is the source code for this function.

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

% lunar free return simple shooting method

% input

%  x(1) = current update to TLI departure angle
%  x(2) = current update to TLI deltav

% output

%  f(1) = objective function (tli delta-v magnitude)
%  f(2) = selenocentric radius (normalized)
%  f(3) = y-component of relative position vector (kilometers)
%  f(4) = x-component of relative velocity vector (kilometers/second)

% Orbital Mechanics with MATLAB

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

global distance_e2m vlc_moon radius_moon omega_moon

global radius_leo vlc_leo tevent yevent rm2sc vm2sc rrel vrel

% current TLI maneuver angle

theta_pe = x(1);

% current inertial TLI state vector

ri(1) = radius_leo * cos(theta_pe);

ri(2) = radius_leo * sin(theta_pe);
```

```
vpe = vlc_leo + x(2);

vi(1) = -vpe * sin(theta_pe);

vi(2) = vpe * cos(theta_pe);

% set up options for ode45

options = odeset('RelTol', 1.0e-10, 'AbsTol', 1.0e-10, 'Events', @fpa_event);

% solve for lunar closest approach conditions

tend = 7.0 * 86400.0;

[t, ysol, tevent, yevent, ie] = ode45(@free_return_eqm, [0 tend], [ri vi], options);

% geocentric inertial lunar position vector (kilometers)

theta_moon = omega_moon * tevent;

re2m(1) = distance_e2m * cos(theta_moon);

re2m(2) = distance_e2m * sin(theta_moon);

% geocentric inertial lunar velocity vector (kilometers/second)

ve2m(1) = -vlc_moon * sin(theta_moon);

ve2m(2) = vlc_moon * cos(theta_moon);

% compute selenocentric inertial state vector of the third body

rm2sc(1) = yevent(1) - re2m(1);

rm2sc(2) = yevent(2) - re2m(2);

vm2sc(1) = yevent(3) - ve2m(1);

vm2sc(2) = yevent(4) - ve2m(2);

% third body position vector in rotating system

rrel(1) = yevent(1) * cos(theta_moon) ...
        + yevent(2) * sin(theta_moon);

rrel(2) = -yevent(1) * sin(theta_moon) ...
        + yevent(2) * cos(theta_moon);

rrel(3) = 0.0;

% angular rotation vector (radians/second)

omegav = omega_moon * [0.0 0.0 1.0];

% third body velocity vector in rotating system

vrel = cross(omegav, rrel);

% objective function (tli deltav, kilometers/second)

f(1) = x(2);

% current selenocentric periapsis radius (normalized)

f(2) = norm(rm2sc) / radius_moon;

% y-component of relative position vector (kilometers)
```

```
f(3) = rrel(2);

% x-component of relative velocity vector (kilometers/second)

% f(4) = vrel(1);

f = f';

% no derivatives

g = [];
```

After the `free_return` script solves for the orbital characteristics and transfer time of the outbound or departure trajectory, the program then propagates the equations of motion for the entire free-return mission. Here is the MATLAB source code that calculates the initial conditions for the optimal solution (inertial geocentric position and velocity vectors), the total time of flight and performs these calculations using the MATLAB `ode45` function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% propagate system of first-order differential equations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ttof = 2.0 * tevent;

r(1) = radius_leo * cos(theta_tli);

r(2) = radius_leo * sin(theta_tli);

vpe = vlc_leo + deltav_tli;

v(1) = -vpe * sin(theta_tli);

v(2) = vpe * cos(theta_tli);

options = odeset('RelTol', 1.0e-10, 'AbsTol', 1.0e-10);

[tn, xn] = ode45('free_return_eqm', [0.0 ttof], [r v], options);
```

The following is the MATLAB source code for the function that evaluates the first-order inertial equations of motion. It uses Professor Richard Battin's $f(q)$ functions to calculate the point-mass gravity acceleration due to the moon.

```
function ydot = eci_eqm (t, y)

% geocentric equations of motion

% version for free_return.m

% input

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

% output

%  ydot = integration vector

% Orbital Mechanics with MATLAB

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

```
global mu_moon mu_earth omega_moon distance_e2m

% acceleration due to the earth

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

r1 = sqrt(r2);

r3 = r2 * r1;

for i = 1:1:3

    agrav(i) = -mu_earth * y(i) / r3;

end

theta_moon = omega_moon * t;

% geocentric position vector of the moon (kilometers)

rmoon(1) = distance_e2m * cos(theta_moon);

rmoon(2) = distance_e2m * sin(theta_moon);

rmoon(3) = 0.0;

% selenocentric position vector of the third body

for i = 1:1:3

    rm2sc(i) = y(i) - rmoon(i);

end

% f(q) formulation

for i = 1:1:3

    vtmp(i) = y(i) - 2.0d0 * rmoon(i);

end

dot1 = dot(y(1:3), vtmp);

dot2 = dot(rmoon, rmoon);

qmoon = dot1 / dot2;

fmoon = qmoon * ((3.0d0 + 3.0d0 * qmoon + qmoon * qmoon) ...
    / (1.0d0 + (1.0d0 + qmoon)^1.5d0));

d3moon = norm(rm2sc)^3;

% point-mass gravity of the moon

for i = 1:1:3

    amoon(i) = -mu_moon * (y(i) + fmoon * rmoon(i)) / d3moon;

end

% compute total integration vector

ydot = [ y(4)
    y(5)
    y(6)
    agrav(1) + amoon(1)
```

```
agrav(2) + amoon(2)
agrav(3) + amoon(3)];
```

Image trajectory verification

At the end of the numerical calculations, the `free_return` script provides information about how well the solution satisfied the trajectory constraints at lunar closest approach. Here is that information for the trajectory example solved earlier.

```
image trajectory mission constraints at lunar closest approach
---------------------------------------------------------------

y-component of relative position vector          0.00044392 meters

x-component of relative velocity vector         -0.00000000 meters/second

moon-third body geocentric separation angle      0.00000000 degrees

geocentric inertial flight path angle            0.00000002 degrees
```

The first data item is the value for the y-component of the relative position vector. This number defines how close the third body was to the earth-moon line at the time of lunar closest approach.

The second item is the x-component of the relative velocity vector, also at lunar closest approach. This number indicates how "perpendicular" the trajectory was at closest approach.

The third data item is the angle between the moon and third body as seen from the earth, also at closest approach. This angle is computed from $\delta = \cos^{-1}\left(\hat{\mathbf{u}}_{E-B} \cdot \hat{\mathbf{u}}_{E-M}\right)$ where $\hat{\mathbf{u}}_{E-B}$ is a unit pointing vector from the earth to the third body and $\hat{\mathbf{u}}_{E-M}$ is a unit pointing vector from the earth to the moon, both calculated at the time of lunar closest approach.

Finally, the last data item is the geocentric inertial flight path angle of the third body at lunar closest approach. It too is a measure of the orthogonality of the flyby trajectory.

Additional information about the unique properties of image trajectories can be found in Professor Angelo Miele's classic paper, "Theorem of Image Trajectories in the Earth-Moon Space", XI International Astronautical Congress, Stockholm, 1960.

Algorithm resources

"Theorem of Image Trajectories in the Earth-Moon Space", Angelo Miele, XI International Astronautical Congress, Stockholm, 1960.

"Revisit of the Theorem of Image Trajectories in the Earth-Moon Space", Angelo Miele, *Journal of Optimization Theory and Applications*, 2010.

"Automated Lunar Free Return Trajectory Generation", Mark C. Jesick and Cesar A. Ocampo, AAS 09-192, American Astronautical Society.

"Circumlunar Trajectory Calculations", MIT Instrumentation Laboratory Report R-353, April 1962.

"Design of Earth-Moon Free-Return Trajectories", Qinqin Luo, Jianfeng Yin and Chao Han, AIAA *Journal of Guidance, Control and Dynamics*, Vol. 36, No. 1, January-February 2013, pp. 263-271.

"Injection Conditions for Lunar Trajectories", R. Kolenkiewicz and W. Putney, NASA TM X-55390, November 1965.

"Coplanar Three-Body Trans-Earth Lunar Trajectory Simulation Methodology", H. Ikawa, AIAA 88-0381, AIAA 26th Aerospace Sciences Meeting, Reno, Nevada, January 11-14, 1988.

"Lunar Constants and Models Document", JPL D-32296, September 23, 2005.