

Program fb_lsocs

Optimal Finite-burn Trans-lunar Injection with SOCS

This document is the user's manual for a Fortran computer program called `fb_lsocs` that uses the Sparse Optimal Control Software (SOCS) object code library developed by the Boeing Company (www.boeing.com/phantom/socs/) to solve the classic finite-burn, trans-lunar injection (TLI) trajectory optimization problem. The software attempts to maximize the spacecraft mass at the end of the TLI propulsive maneuver while targeting to either a user-defined periapsis radius and orbital inclination or B-plane coordinates of the arrival hyperbola relative to the moon. Since the TLI is a continuous thrust maneuver, maximizing the spacecraft mass is equivalent to minimizing the propellant required.

The important features of this scientific simulation are as follows:

- Modified equinoctial equations of motion during the TLI propulsive maneuver
- Cartesian equations of motion during the geocentric coast to lunar encounter
- Earth J_2 gravity model and sun and moon point-mass perturbations
- JPL DE410 lunar and solar ephemeris
- B-plane coordinates at lunar close approach

SOCS is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in SOCS can be found in the book, "Practical Methods for Optimal Control Using Nonlinear Programming" by John. T. Betts, SIAM, 2001.

The `fb_lsocs` software consists of Fortran routines that perform the following tasks:

- main program that sets algorithm control parameters and calls the SOCS transcription/optimal control subroutine
- define problem characteristics and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- evaluate the *right-hand-side* differential equations
- define and compute any point and path constraints
- display the optimal solution results

SOCS will use this information to *automatically* transcribe the user's problem and perform the optimization. The software allows the user to select the type of collocation method and other important algorithm control parameters.

Typical Input File

The `fb_lsocs` computer program is “data-driven” by a simple user-created text file. The following is a typical input or “simulation definition” file used by the software. In this 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. ASCII text input is not case sensitive but must be spelled correctly. Please note that the fundamental time argument in this simulation is ephemeris time (TDB) which is the time argument of the DE410 ephemeris. Furthermore, the fundamental coordinate system is the Earth mean equator and equinox of J2000 (EME2000).

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

```
*****
finite-burn lunar trajectory optimization
two phase, 3-body geocentric motion
program: fb_lsocs   input:lsocsl.in
December 17, 2005
*****
```

The software allows the user to specify an initial guess for the calendar date and time of the TLI maneuver and lower and upper bounds on the actual date found during the optimization process. For any guess for maneuver time t_{TLI} and user-defined lower and upper bounds Δt_l and Δt_u , the TLI maneuver time t is constrained as follows:

$$t_{TLI} - \Delta t_l \leq t \leq t_{TLI} + \Delta t_u$$

For a fixed maneuver time, the lower and upper bounds should be set to $\pm 1.0D-6$.

The user inputs for the initial calendar date, TDB time, and search boundary are as follows:

```
initial calendar date (month, day, year)
10, 11, 2008

initial TDB time guess (hours, minutes, seconds)
16, 20, 6

lower bound for initial time guess (hours)
-12.0

upper bound for initial time guess (hours)
+12.0
```

The next three inputs define the initial spacecraft mass, and the thrust and specific impulse of the propulsion system.

```
initial spacecraft mass (kilograms)
1000.0

thrust magnitude (newtons)
5000.0
```

specific impulse (seconds)
450.0

The type of propulsion initial guess is determined by the next integer input.

```
*****  
type of propulsive initial guess  
*****  
1 = thrust duration  
2 = delta-v  
-----  
2
```

For option 1, the next input is the user's initial guess for the magnitude of the maneuver delta-v.

initial guess for delta-v (meters/second)
3150.0

For option 2, the next three inputs are the user's initial guess for the thrust duration and lower and upper bounds for this duration.

initial guess for thrust duration (seconds)
3000.0

lower bound for thrust duration (seconds)
1.0

upper bound for thrust duration (seconds)
10000.0

The next series of inputs define the characteristics of the initial park orbit. The orbital inclination should be with respect to the EME2000 system.

```
*****  
initial park orbit  
*****  
  
semimajor axis (kilometers)  
6563.3363  
  
orbital eccentricity (non-dimensional)  
0.0d0  
  
orbital inclination (degrees)  
28.5d0  
  
argument of perigee (degrees)  
0.0d0  
  
right ascension of the ascending node (degrees)  
254.334  
  
true anomaly (degrees)  
318.48
```

The next input specifies the type of initial park orbit constraints to enforce.

```
*****  
park orbit constraint options  
*****  
1 = constrain all initial orbital elements except true longitude  
2 = constrained a, e, i; bounded raan & true longitude  
-----
```

The next set of inputs defines the user's initial guess for the lunar transfer time, along with a lower and upper bound for the transfer time. The transfer time here refers to the time from burnout of the propulsive maneuver to the entrance to the lunar sphere-of-influence.

```
transfer time initial guess (hours)
95.0

transfer time lower bound (hours)
84.0d0

transfer time upper bound (hours)
108.0d0
```

For a fixed transfer time, both the lower and upper bounds should be set to the value of the transfer time initial guess.

The next integer input defines the type of final orbit targeting to use during the trajectory optimization. Option 1 will use a selenocentric radius and inclination input by the user. Program option 2 will use b-plane coordinates provided by the user.

```
*****
type of final orbit targeting
-----
1 = periapsis radius and inclination
2 = user-defined b-plane coordinates
*****
1
```

The next two inputs define the periapsis radius and orbital inclination of the lunar trajectory relative to the moon. The inclination should be specified relative to the mean lunar equator.

```
-----
final lunar orbit characteristics
(mean lunar equator)
-----

periapsis radius (kilometers)
1838.0

orbital inclination (degrees)
90.0
```

The next two inputs define the b-plane coordinates of the encounter hyperbola. These coordinates should be specified relative to the lunar mean equator and IAU node of epoch.

```
-----
user-defined b-plane targets
(lunar mean equator and IAU node of epoch)
-----

b dot r target (kilometers)
5016.917844109449106d0

b dot t target (kilometers)
0.0d0
```

The next three inputs define the types of perturbations to include during the numerical solution of the spacecraft's equations of motion.

```

*****
trajectory perturbations
*****

include j2 gravity perturbation (1 = yes, 0 = no)
1

include solar perturbations (1 = yes, 0 = no)
1

include lunar perturbations (1 = yes, 0 = no)
1

```

The next program input is the user-defined radius of the lunar sphere-of-influence (SOI) used by the software during the trajectory optimization.

```

-----
radius of lunar sphere-of-influence (kilometers)
-----
25000.0

```

The next input is an integer that defines the type of initial guess to use. Additional information about options 1 and 2 can be found in the technical discussion section. Option 3 uses a binary disk file created from a previous simulation.

```

*****
* initial guess/restart option *
*****
1 = numerical integration with gravity-turn thrusting
2 = numerical integration with tangential thrusting
3 = binary data file
-----
1

```

This program input is the name of the disk file to use for a binary data file initial guess.

```

name of initial guess/restart input data file
lsocsl.rsbin

```

This next section includes user-defined inputs related to the creation of binary restart and solution data files.

```

*****
* restart and solution data file options *
*****

```

The next input allows the creation of a binary file of the solution. This binary file can also be used as an initial guess for other simulations.

```

create/update binary restart data file (yes or no)
no

```

The name of the binary data file is specified in this next input.

```

name of binary restart data file
lsocsl.rsbin

```

The next program input is an integer that defines the format of the solution file created by the software.

```

*****
* type of comma-delimited solution data file *
*****

```

```

1 = SOCS-defined nodes
2 = user-defined nodes
3 = user-defined step size
-----
1

```

For options 2 or 3, this next input defines either the number of data points or the time step size of the data output in the solution file.

```

number of user-defined nodes or print step size in solution data file
1

```

This text input specifies the name of the solution file created by the software.

```

name of solution output file
lsocs1.csv

```

The next series of program inputs are algorithm control options and parameters for the SOCS software. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```

*****
* algorithm control parameters *
*****

discretization/collocation method
-----
1 = trapezoidal
2 = separated Hermite-Simpson
3 = compressed Hermite-Simpson
4 = Runge-Kutta 4-stage
-----
1

```

The next integer defines the number of initial grid points to use in the collocation modeling of the propulsive maneuver (phase 1) and the lunar coast phase (phase 2).

```

number of grid points in phase 1 (TLI thrust maneuver)
10

number of grid points in phase 2 (coast to lunar encounter)
75

```

The next input defines the relative error in the objective function. A value of 1.0d-5 is recommended.

```

relative error in the objective function (performance index)
1.0d-5

```

The next input defines the relative error in the solution of the differential equations. A value of 1.0d-7 is recommended.

```

relative error in the solution of the differential equations
1.0d-7

```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```

maximum number of mesh refinement iterations
20

```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
5000
```

The level of output from the SOCS NLP algorithm is controlled with the following integer input.

```
*****
sparse NLP iteration output
-----
1 = none
2 = terse
3 = standard
4 = interpretive
5 = diagnostic
-----
1
```

The level of output from the SOCS optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
*****
optimal control output
-----
1 = none
2 = terse
3 = standard
4 = interpretive
-----
1
```

The level of output from the SOCS differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
*****
differential equation output
-----
1 = none
2 = terse
3 = standard
4 = interpretive
5 = diagnostic
-----
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the SOCS user's manual. To ignore this special output control, input the simple character string no.

```
*****
user-defined output
-----
input no to ignore
-----
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

Optimal Control Solution and Trajectory Plots

The following is the program output created by the fb_lsocs simulation for this example. This display summarizes the characteristics of the optimized TLI maneuver and the orbital transfer conditions before and after the maneuver. It also displays the conditions at closest approach to the moon along with the corresponding B-plane characteristics.

```
-----
program fb_lsocs
-----

input data file ==> lsocsl.in

gravity-turn initial guess

a, e, i constrained; bounded RAAN & true longitude

periapsis radius and inclination targets

-----
park orbit initial conditions
(geocentric - Earth equator and equinox of J2000)
-----

calendar date          October  11, 2008

TDB time               18:59:15.812

TDB Julian date       2454751.291155228856951

      sma (km)          eccentricity      inclination (deg)      argper (deg)
0.656333630000D+04    0.498144785817D-15    0.285000000000D+02    0.000000000000D+00

      raan (deg)        true anomaly (deg)      arglat (deg)          period (min)
0.264951679228D+03    0.295463645956D+03    0.295463645956D+03    0.881955589276D+02

      rx (km)           ry (km)                   rz (km)               rmag (km)
-.543576938825D+04    -.235263082820D+04    -.282752943943D+04    0.656333630000D+04

      vx (kps)          vy (kps)                   vz (kps)              vmag (kps)
0.231393623600D+01    -.726781474738D+01    0.159873181417D+01    0.779303378153D+01

-----
time and conditions at end of TLI finite burn
(geocentric - Earth equator and equinox of J2000)
-----

calendar date          October  11, 2008

TDB time               19:06:46.000

TDB Julian date       2454751.296365740709007

      sma (km)          eccentricity      inclination (deg)      argper (deg)
0.189281815888D+06    0.965091796701D+00    0.285150259839D+02    0.312589841009D+03

      raan (deg)        true anomaly (deg)      arglat (deg)          period (min)
0.264912456774D+03    0.184456698090D+02    0.331035510818D+03    0.136591310209D+05

      rx (km)           ry (km)                   rz (km)               rmag (km)
-.339896367581D+04    -.565151631447D+04    -.156708605206D+04    0.677852118214D+04
```

| | | | |
|--------------------|--------------------|--------------------|--------------------|
| vx (kps) | vy (kps) | vz (kps) | vmag (kps) |
| 0.682294036014D+01 | -.725347986390D+01 | 0.404172743508D+01 | 0.107471412922D+02 |

| | |
|-----------------|------------------------------------|
| final mass | 489.928631060075361 kilograms |
| propellant mass | 510.071368939924639 kilograms |
| thrust duration | 450.188225068319298 seconds |
| delta-v | 3148.653155266281374 meters/second |

time and conditions at beginning of lunar coast
(geocentric - Earth equator and equinox of J2000)

| | |
|-----------------|-------------------------|
| calendar date | October 11, 2008 |
| TDB time | 19:06:46.000 |
| TDB Julian date | 2454751.296365740709007 |

| | | | |
|--------------------|--------------------|--------------------|--------------------|
| sma (km) | eccentricity | inclination (deg) | argper (deg) |
| 0.189281816093D+06 | 0.965091796739D+00 | 0.285150259838D+02 | 0.312589841010D+03 |
| raan (deg) | true anomaly (deg) | arglat (deg) | period (min) |
| 0.264912456774D+03 | 0.184456698077D+02 | 0.331035510818D+03 | 0.136591310431D+05 |
| rx (km) | ry (km) | rz (km) | rmag (km) |
| -.339896367587D+04 | -.565151631448D+04 | -.156708605209D+04 | 0.677852118218D+04 |
| vx (kps) | vy (kps) | vz (kps) | vmag (kps) |
| 0.682294036020D+01 | -.725347986394D+01 | 0.404172743512D+01 | 0.107471412922D+02 |

time and conditions at lunar sphere-of-influence
(geocentric - Earth equator and equinox of J2000)

| | |
|-----------------|-------------------------|
| calendar date | October 16, 2008 |
| TDB time | 00:33:51.871 |
| TDB Julian date | 2454755.523517020046711 |

| | | | |
|--------------------|--------------------|--------------------|--------------------|
| sma (km) | eccentricity | inclination (deg) | argper (deg) |
| 0.183497635998D+06 | 0.984861595676D+00 | 0.799460956320D+02 | 0.339977706293D+03 |
| raan (deg) | true anomaly (deg) | arglat (deg) | period (min) |
| 0.222743458398D+03 | 0.179348997182D+03 | 0.159326703475D+03 | 0.130378338035D+05 |
| rx (km) | ry (km) | rz (km) | rmag (km) |
| 0.264382927940D+06 | 0.213899403492D+06 | 0.126078914496D+06 | 0.362694334228D+06 |
| vx (kps) | vy (kps) | vz (kps) | vmag (kps) |
| 0.885366807066D-01 | 0.102586173743D+00 | -.860021746892D-01 | 0.160496233367D+00 |

time and conditions at lunar sphere-of-influence
(selenocentric - lunar equator and equinox of J2000)

calendar date October 16, 2008
TDB time 00:33:51.871
TDB Julian date 2454755.523517020046711

| | | | |
|--------------------|--------------------|--------------------|--------------------|
| sma (km) | eccentricity | inclination (deg) | argper (deg) |
| -.596050692806D+04 | 0.130782744423D+01 | 0.902322005572D+02 | 0.312210012950D+03 |
| raan (deg) | true anomaly (deg) | arglat (deg) | period (min) |
| 0.315795189331D+03 | 0.230571349659D+03 | 0.182781362608D+03 | 0.000000000000D+00 |
| rx (km) | ry (km) | rz (km) | rmag (km) |
| -.178967621868D+05 | 0.174136229216D+05 | -.121311183086D+04 | 0.250000000068D+05 |
| vx (kps) | vy (kps) | vz (kps) | vmag (kps) |
| 0.785000656700D+00 | -.762777082084D+00 | -.129292841033D+00 | 0.110216675087D+01 |

time and conditions at lunar closest approach
(selenocentric - lunar equator and equinox of J2000)

calendar date October 16, 2008
TDB time 05:59:00.436
TDB Julian date 2454755.749310600571334

| | | | |
|--------------------|--------------------|--------------------|--------------------|
| sma (km) | eccentricity | inclination (deg) | argper (deg) |
| -.592797079585D+04 | 0.131005550869D+01 | 0.900000000032D+02 | 0.312094691149D+03 |
| raan (deg) | true anomaly (deg) | arglat (deg) | period (min) |
| 0.315764166904D+03 | 0.000000000000D+00 | 0.312094691149D+03 | 0.000000000000D+00 |
| rx (km) | ry (km) | rz (km) | rmag (km) |
| 0.882780902386D+03 | -.859541768726D+03 | -.136386576624D+04 | 0.183800000058D+04 |
| vx (kps) | vy (kps) | vz (kps) | vmag (kps) |
| 0.131973698135D+01 | -.128499501563D+01 | 0.166405341935D+01 | 0.248233593065D+01 |

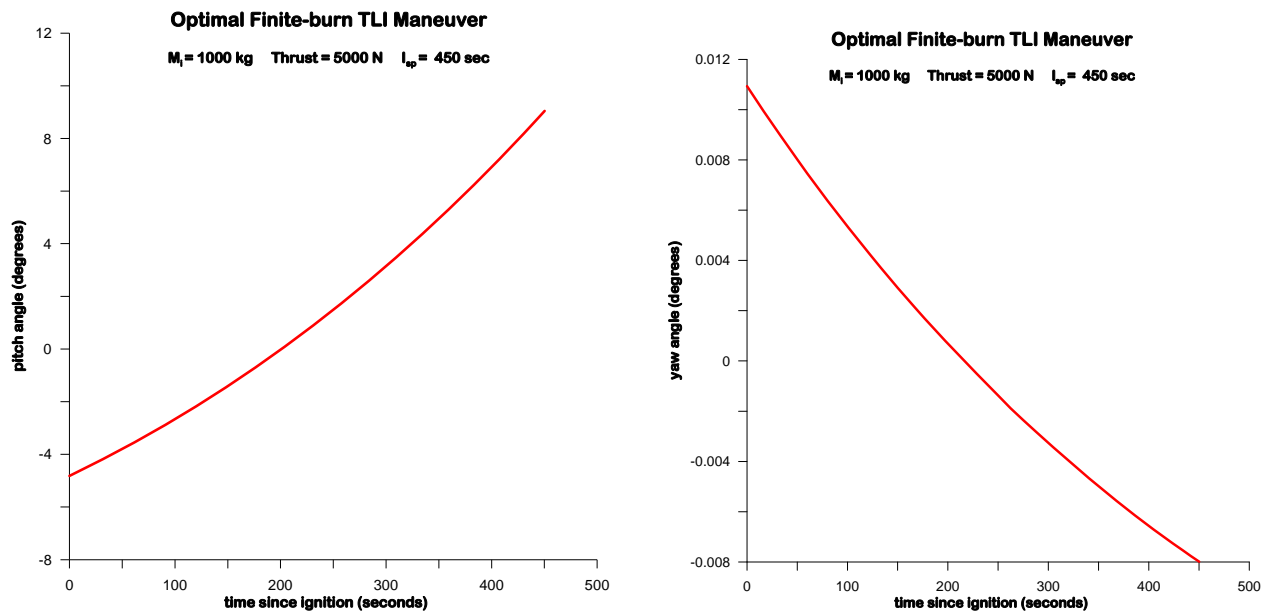
b-plane coordinates of incoming hyperbola
(selenocentric - lunar equator and equinox of J2000)

| | |
|-------------|-----------------------------------|
| b-magnitude | 5016.917844109449106 kilometers |
| b dot r | 5016.917844109449106 |
| b dot t | -2.856209562196454E-007 |
| theta | 90.000000003261945 degrees |
| v-infinity | 909.429570852182451 meters/second |
| r-periapsis | 1838.000000580224423 kilometers |
| decl-asy | -7.663641965081077 degrees |
| rasc-asy | 315.764166904254012 degrees |

coast duration to SOI 101.576682988554239 hours
coast duration to closest approach 106.995728921145201 hours

The delta-v is determined from a cubic spline integration of the thrust acceleration evaluated at the grid points determined by SOCS.

The following plots illustrate the behavior of the pitch and yaw angles during the propulsive maneuver.



The `fb_lsocs` software will create three comma-separated-variable (csv) output files. The first file is named `parkorb.csv` and contains the geocentric, EME 2000 state vector of the park orbit. The second file contains the state vectors and orbital elements of the geocentric transfer orbit with the name specified by the user in the main input data file. The third file is named `soiorb.csv` and contains the state vector of the selenocentric orbit at the lunar sphere-of-influence.

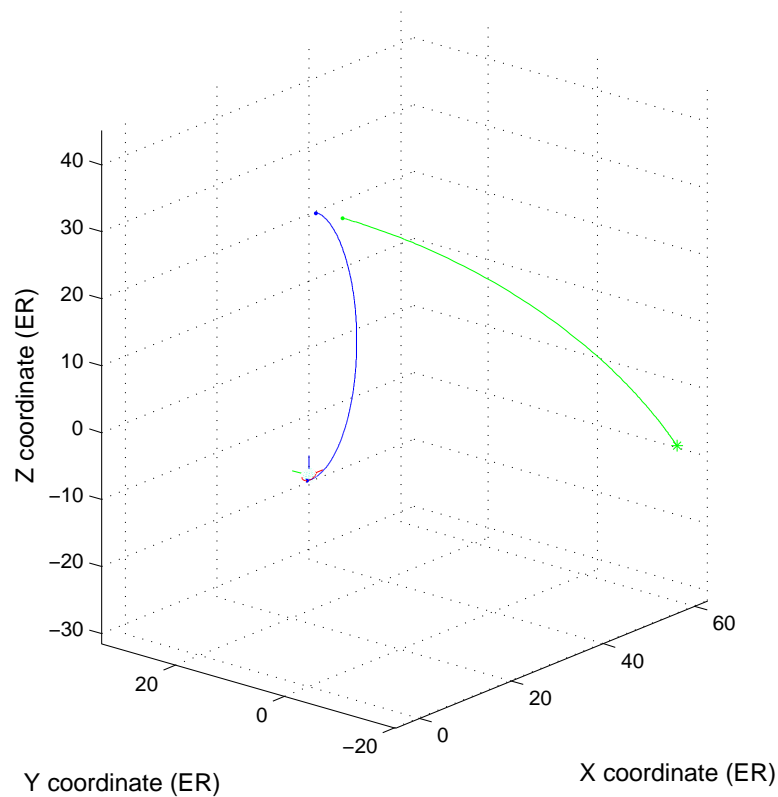
This software package also includes a MATLAB script called `lplot.m` that can be used to create trajectory graphic displays using these three data files. The interactive graphic features of MATLAB allow the user to rotate and zoom the displays. These capabilities allow the user to interactively find the best viewpoint as well as verify basic orbital geometry of the geocentric and selenocentric trajectories.

Important note!!

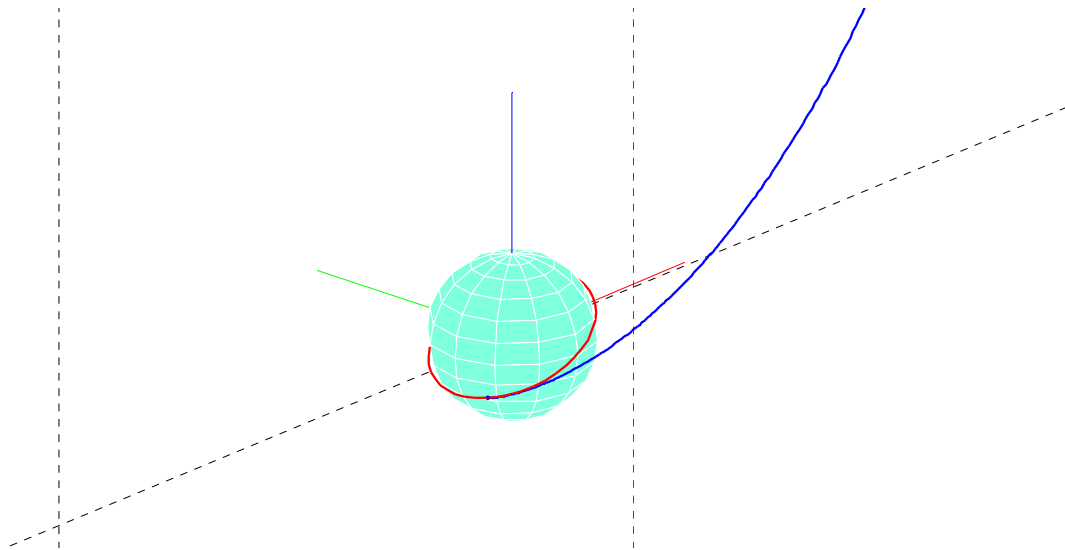
You must delete the first or “header” line of the user-defined solution file in order for the `lplot` script to work properly. This script uses the MATLAB `csvread` function to read the data file which can only contain comma-separated-variable numerical data.

The following is the transfer trajectory from burnout of the TLI maneuver to the lunar SOI for this example. The park orbit trajectory is red, the lunar transfer is blue and the moon’s orbit is green. The location of the moon at the time of the TLI maneuver is marked with a green asterisk. The coordinates are in the units of Earth radius (ER). This display is also labeled with a geocentric, inertial coordinate system. The x-axis of this system is red, the y-axis is green and the z-axis is blue.

Geocentric Transfer Trajectory

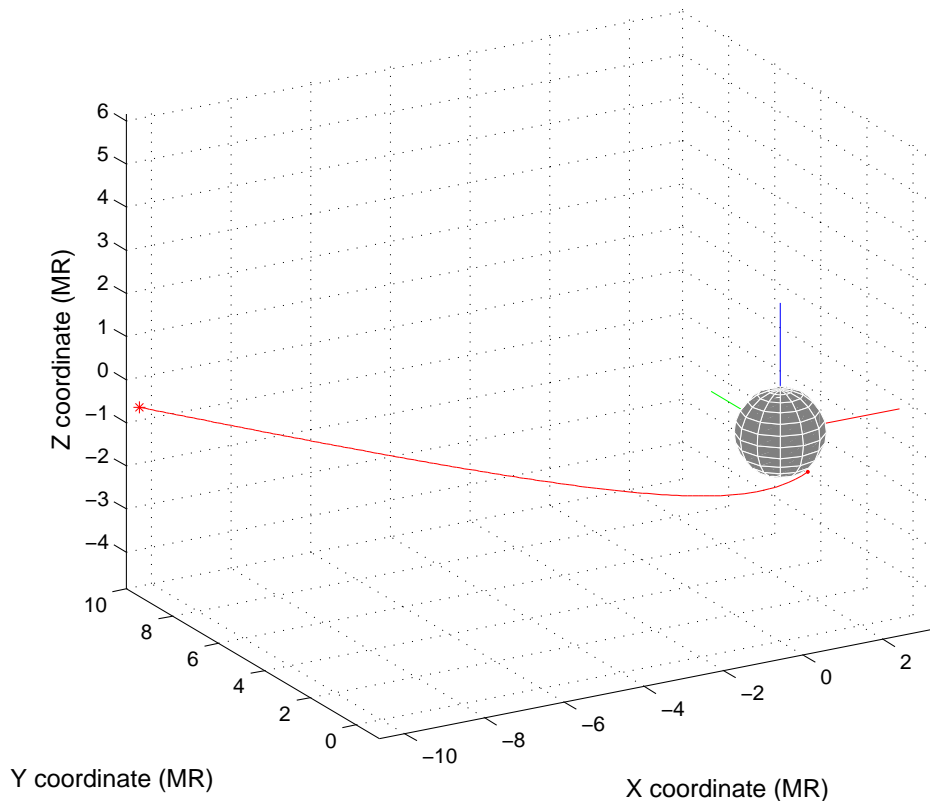


The following is a “zoomed” plot of the park orbit and initial portion of the lunar transfer trajectory.



The following is a plot of the selenocentric hyperbola within the user-defined lunar sphere-of-influence (SOI). The coordinate units are lunar radii (MR). The entry into the SOI is marked with an asterisk and the location of the periapsis or lunar closest approach is marked with a red dot. This display is labeled with a selenocentric, inertial coordinate system. The x-axis is red, the y-axis is green and the z-axis is blue.

Selenocentric Trajectory



Problem setup for SOCS

This section provides additional details about the software implementation. It explains such things as point and path constraints, the performance index and the numerical technique used to create an initial guess for the software.

Creating an initial guess for SOCS

This section describes the algorithms used in `fb_1socs` to create an initial guess for SOCS. The SOCS software attempts to obtain problem feasibility before trying to solve the optimal control problem. If it can not get feasible, the user will have to provide a better initial guess.

The software allows the user to input either a delta-v or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. For either type of initial guess, the user should also provide lower and upper bounds for the total thrust duration.

An estimate of the thrust duration can be determined from the following expression:

$$t_d = \frac{I_{sp} m_p g}{F} = \frac{m_p V_{ex}}{F}$$

The propellant mass required for a given ΔV is a function of the initial (or final) mass of the spacecraft and the exhaust velocity as follows:

$$m_p = m_i \left(1 - e^{\frac{-\Delta V}{V_{ex}}} \right) = m_f \left(e^{\frac{\Delta V}{V_{ex}}} - 1 \right)$$

In these equations

m_i = initial mass

m_f = final mass

m_p = propellant mass

V_{ex} = exhaust velocity = $g I_{sp}$

I_{sp} = specific impulse

ΔV = impulsive velocity increment

F = thrust

g = acceleration of gravity

The software requires an initial guess for the thrust duration. The user should also provide lower and upper bounds for the total thrust duration. All of these inputs should be in seconds.

The software also allows the user to select the type of steering method used to generate an initial guess for the trajectory. The two options available are *tangential* or *gravity turn* steering. For gravity turn steering the thrust vector is aligned with the instantaneous velocity vector. For tangential thrusting the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = [0 \ 1 \ 0]^T$. The software numerically integrates the vehicle equations of motion. Please note that either type of steering method creates a *coplanar* initial guess.

The initial guess for the SOCS algorithm is created by numerically integrating the modified equinoctial equations of motion for the user-defined initial guess or rocket equation calculation for the orbital maneuver time.

For the gravity-turn steering option, the software creates the unit thrust vector in the Earth-centered-inertial (ECI) coordinate system and integrates the equations of motion in the modified equinoctial or radial frame system. The relationship between a unit thrust vector in the ECI coordinate system $\hat{\mathbf{u}}_{T_{ECI}}$ and the corresponding unit thrust vector in the modified equinoctial system $\hat{\mathbf{u}}_{T_{MEE}}$ is given by

$$\hat{\mathbf{u}}_{T_{ECI}} = \begin{bmatrix} \hat{\mathbf{i}}_r & \hat{\mathbf{i}}_t & \hat{\mathbf{i}}_n \end{bmatrix} \hat{\mathbf{u}}_{T_{MEE}}$$

where

$$\hat{\mathbf{i}}_r = \frac{\mathbf{r}}{|\mathbf{r}|} \quad \hat{\mathbf{i}}_n = \frac{\mathbf{r} \times \mathbf{v}}{|\mathbf{r} \times \mathbf{v}|} \quad \hat{\mathbf{i}}_t = \hat{\mathbf{i}}_n \times \hat{\mathbf{i}}_r = \frac{(\mathbf{r} \times \mathbf{v}) \times \mathbf{r}}{|\mathbf{r} \times \mathbf{v}| |\mathbf{r}|}$$

This relationship can also be expressed as

$$\hat{\mathbf{u}}_{T_{ECI}} = [\mathcal{Q}] \hat{\mathbf{u}}_{T_{MEE}} = \begin{bmatrix} \hat{\mathbf{r}}_x & (\hat{\mathbf{h}} \times \hat{\mathbf{r}})_x & \hat{\mathbf{h}}_x \\ \hat{\mathbf{r}}_y & (\hat{\mathbf{h}} \times \hat{\mathbf{r}})_y & \hat{\mathbf{h}}_y \\ \hat{\mathbf{r}}_z & (\hat{\mathbf{h}} \times \hat{\mathbf{r}})_z & \hat{\mathbf{h}}_z \end{bmatrix} \hat{\mathbf{u}}_{T_{MEE}}$$

Finally, the transformation of the unit thrust vector in the ECI system to the modified equinoctial coordinate system is given by

$$\hat{\mathbf{u}}_{T_{MEE}} = [\mathcal{Q}]^T \hat{\mathbf{u}}_{T_{ECI}}$$

(1) Initial orbit constraints

The software allows the user to select one of the following initial orbit constraint options:

- 1) constrain all initial orbital elements except true longitude
- 2) constrain semimajor axis, eccentricity and inclination; bounded RAAN and true longitude

For both options, the initial orbit inclination is constrained by enforcing

$$\sqrt{h^2 + k^2} = \tan\left(\frac{i}{2}\right)$$

where i is the park orbit inclination. Furthermore, the semiparameter is constrained to the initial value according to

$$p_L = p_U = p_i$$

If the park orbit is circular, the software enforces the following two constraints:

$$f = 0 \quad g = 0$$

Otherwise, for an elliptical park orbit, the single equality constraint

$$\sqrt{f^2 + g^2} = e$$

is enforced, where e is the park orbit eccentricity.

For program option 1, both lower and upper bounds for the h and k modified equinoctial elements are set equal to the initial elements as follows:

$$h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

For both options, the initial true longitude is bounded according to

$$-2\pi \leq L_i \leq +2\pi$$

and for option 2, the RAAN of the initial park orbit is bounded according to

$$\Omega_i - 30^\circ \leq \Omega \leq \Omega_i + 30^\circ$$

where Ω_i is the user's initial guess for RAAN.

In SOCS terminology, these constraints or boundary conditions are called *point functions*.

(2) Performance index – maximize final spacecraft mass

The objective function or performance index J for this simulation is the mass of the spacecraft at burnout of the lunar injection stage. This is simply

$$J = m_f$$

The value of the `maxmin` indicator in SOCS tells the software whether the user is minimizing or maximizing the performance index. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass. The spacecraft mass at the initial time is constrained to the user-defined initial value.

(3) Path constraint – unit thrust vector scalar magnitude

During the TLI propulsive maneuver, the scalar magnitude of the components of the unit thrust vector at any time during the maneuver is constrained as follows:

$$|\mathbf{u}_T| = \sqrt{u_{T_x}^2 + u_{T_y}^2 + u_{T_z}^2} = 1$$

(4) Point functions – periapsis altitude and selenocentric orbital inclination

At the sphere-of-influence of the Moon, the point function enforced by the software is given by

$$r_{SOI_p} - r_{SOI} = 0$$

where r_{SOI_p} is the predicted SOI radius and r_{SOI} is the value defined by the user.

Targeting to a selenocentric periapsis radius and orbital inclination

For user-defined periapsis radius and orbital inclination targets at the moon, the following point functions or equality constraints are enforced

$$r_p - r_{ca} = 0$$

$$\cos i - \hat{\mathbf{h}}_z = 0$$

where r_p and i are the user-defined periapsis radius and selenocentric orbital inclination of the encounter hyperbola, respectively. In the second equation, $\hat{\mathbf{h}}_z$ is the z-component of the predicted unit angular momentum vector. These orbital elements are determined from the spacecraft's state vector at closest approach to the moon. The orbital inclination point function is expressed in the lunar mean equator and IAU node of epoch coordinate system.

Targeting to user-defined B-plane coordinates

For this program option, the two equality constraints are simply the difference between the predicted and the user-defined $\mathbf{B}\cdot\mathbf{T}$ and $\mathbf{B}\cdot\mathbf{R}$ components. These coordinates are also determined from the spacecraft's state vector at closest approach to the moon. The B-plane coordinates are expressed in the lunar mean equator and equinox of J2000 coordinate system.

Time from the lunar SOI to closest approach

The elapsed time from the lunar SOI until closest approach to the moon is determined by an algorithm that includes Brent's one-dimensional root-finder embedded within a Runge-Kutta-Fehlberg 7(8) numerical integration method. This technique searches for the time at which the selenocentric flight path angle γ of the spacecraft is zero. This mission constraint is computed as follows

$$\gamma = \sin^{-1}\left(\frac{\mathbf{r}\cdot\mathbf{v}}{|\mathbf{r}\cdot\mathbf{v}|}\right)$$

where \mathbf{r} and \mathbf{v} are the selenocentric position and velocity vectors, respectively.

The RKF78 method numerically solves the following system of six first-order, nonlinear differential equations of orbital motion

$$\begin{aligned}\dot{y}_1 &= v_x \\ \dot{y}_2 &= v_y \\ \dot{y}_3 &= v_z \\ \dot{y}_4 &= -\mu\frac{r_x}{r^3}\left\{1 + \frac{3}{2}\frac{J_2 r_{eq}^2}{r^2}\left(1 - \frac{5r_z^2}{r^2}\right)\right\} + a_{s_x} + a_{e_x} \\ \dot{y}_5 &= -\mu\frac{r_y}{r^3}\left\{1 + \frac{3}{2}\frac{J_2 r_{eq}^2}{r^2}\left(1 - \frac{5r_z^2}{r^2}\right)\right\} + a_{s_y} + a_{e_y} \\ \dot{y}_6 &= -\mu\frac{r_z}{r^3}\left\{1 + \frac{3}{2}\frac{J_2 r_{eq}^2}{r^2}\left(3 - \frac{5r_z^2}{r^2}\right)\right\} + a_{s_z} + a_{e_z}\end{aligned}$$

In these equations, μ and r_{eq} are the gravitational constant and equatorial radius of the moon, respectively and J_2 is the non-dimensional oblateness gravity coefficient. The coefficient used by the `lsocs` computer program corresponds to the GLGM-1 value of 2.037448533865259d-4.

Technical Discussion

This section summarizes the algorithms implemented in the `fb_1socs` software. It includes a summary of the equations of motion used during the TLI propulsive maneuver and the geocentric coast to lunar encounter.

Modified equinoctial equations of motion

The modified equinoctial orbital elements are a set of orbital elements that are useful for trajectory analysis and optimization. They are valid for circular, elliptic, and hyperbolic orbits. These equations exhibit no singularity for zero eccentricity and orbital inclinations equal to 0 and 90 degrees. However, two components of the orbital element set are singular for an orbital inclination of 180 degrees. For this application the equations are well-behaved as the spacecraft transitions from a circular or elliptical park orbit to hyperbolic flight conditions.

The relationship between direct modified equinoctial and classical orbital elements is defined by the following definitions

$$\begin{aligned}p &= a(1 - e^2) \\f &= e \cos(\omega + \Omega) \\g &= e \sin(\omega + \Omega) \\h &= \tan(i/2) \cos \Omega \\k &= \tan(i/2) \sin \Omega \\L &= \Omega + \omega + \theta\end{aligned}$$

where

$$\begin{aligned}p &= \text{semiparameter} \\a &= \text{semimajor axis} \\e &= \text{orbital eccentricity} \\i &= \text{orbital inclination} \\\omega &= \text{argument of periapsis} \\\Omega &= \text{right ascension of the ascending node} \\\theta &= \text{true anomaly} \\L &= \text{true longitude}\end{aligned}$$

The relationship between classical and modified equinoctial orbital elements is:

semimajor axis

$$a = \frac{p}{1 - f^2 - g^2}$$

orbital eccentricity

$$e = \sqrt{f^2 + g^2}$$

orbital inclination

$$i = 2 \tan^{-1} \left(\sqrt{h^2 + k^2} \right)$$

argument of periapsis

$$\omega = \tan^{-1} (g/f) - \tan^{-1} (k/h)$$

right ascension of the ascending node

$$\Omega = \tan^{-1} (k/h)$$

true anomaly

$$\theta = L - (\Omega + \omega) = L - \tan^{-1} (g/f)$$

The mathematical relationships between an inertial state vector and the corresponding modified equinoctial elements are summarized as follows:

position vector

$$\mathbf{r} = \begin{bmatrix} \frac{r}{s^2} (\cos L + \alpha^2 \cos L + 2hk \sin L) \\ \frac{r}{s^2} (\sin L - \alpha^2 \sin L + 2hk \cos L) \\ \frac{2r}{s^2} (h \sin L - k \cos L) \end{bmatrix}$$

velocity vector

$$\mathbf{v} = \begin{bmatrix} -\frac{1}{s^2} \sqrt{\frac{\mu}{p}} (\sin L + \alpha^2 \sin L - 2hk \cos L + g - 2fhk + \alpha^2 g) \\ -\frac{1}{s^2} \sqrt{\frac{\mu}{p}} (-\cos L + \alpha^2 \cos L + 2hk \sin L - f + 2ghk + \alpha^2 f) \\ \frac{2}{s^2} \sqrt{\frac{\mu}{p}} (h \cos L + k \sin L + fh + gk) \end{bmatrix}$$

where

$$\alpha^2 = h^2 - k^2$$

$$s^2 = 1 + h^2 + k^2$$

$$r = \frac{p}{w}$$

$$w = 1 + f \cos L + g \sin L$$

The system of first-order modified equinoctial equations of orbital motion are given by

$$\dot{p} = \frac{dp}{dt} = \frac{2p}{w} \sqrt{\frac{p}{\mu}} \Delta_t$$

$$\dot{f} = \frac{df}{dt} = \sqrt{\frac{p}{\mu}} \left[\Delta_r \sin L + [(w+1) \cos L + f] \frac{\Delta_t}{w} - (h \sin L - k \cos L) \frac{g \Delta_n}{w} \right]$$

$$\dot{g} = \frac{dg}{dt} = \sqrt{\frac{p}{\mu}} \left[-\Delta_r \cos L + [(w+1) \sin L + g] \frac{\Delta_t}{w} + (h \sin L - k \cos L) \frac{g \Delta_n}{w} \right]$$

$$\dot{h} = \frac{dh}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \cos L$$

$$\dot{k} = \frac{dk}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \sin L$$

$$\dot{L} = \frac{dL}{dt} = \sqrt{\mu p} \left(\frac{w}{p} \right)^2 + \frac{1}{w} \sqrt{\frac{p}{\mu}} (h \sin L - k \cos L) \Delta_n$$

where $\Delta_r, \Delta_t, \Delta_n$ are *non-two-body* perturbations in the radial, tangential and normal directions, respectively. For a lunar spacecraft, the radial direction is along the geocentric radius vector of the spacecraft measured positive in a direction away from the gravitational center, the tangential direction is perpendicular to this radius vector measured positive in the direction of orbital motion, and the normal direction is positive along the angular momentum vector of the spacecraft's orbit.

The equations of orbital motion can also be expressed in vector form as follows:

$$\dot{\mathbf{y}} = \frac{d\mathbf{y}}{dt} = \mathbf{A}(\mathbf{y}) \mathbf{P} + \mathbf{b}$$

where

$$\mathbf{A} = \begin{pmatrix} 0 & \frac{2p}{w} \sqrt{\frac{p}{\mu}} & 0 \\ \sqrt{\frac{p}{\mu}} \sin L & \sqrt{\frac{p}{\mu}} \frac{1}{q} \{(w+1) \cos L + f\} & -\sqrt{\frac{p}{\mu}} \frac{g}{w} \{h \sin L - k \cos L\} \\ -\sqrt{\frac{p}{\mu}} \cos L & \sqrt{\frac{p}{\mu}} \{(w+1) \sin L + g\} & \sqrt{\frac{p}{\mu}} \frac{f}{w} \{h \sin L - k \cos L\} \\ 0 & 0 & \sqrt{\frac{p}{\mu}} \frac{s^2 \cos L}{2w} \\ 0 & 0 & \sqrt{\frac{p}{\mu}} \frac{s^2 \sin L}{2w} \\ 0 & 0 & \sqrt{\frac{p}{\mu}} \{h \sin L - k \cos L\} \end{pmatrix}$$

and

$$\mathbf{b} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{\mu p} \left(\frac{w}{p}\right)^2 \end{bmatrix}^T$$

The total non-two-body acceleration vector is given by

$$\mathbf{P} = \Delta_r \hat{\mathbf{i}}_r + \Delta_t \hat{\mathbf{i}}_t + \Delta_n \hat{\mathbf{i}}_n$$

where $\hat{\mathbf{i}}_r$, $\hat{\mathbf{i}}_t$ and $\hat{\mathbf{i}}_n$ are unit vectors in the radial, tangential and normal directions. These unit vectors can be computed from the inertial position vector \mathbf{r} and velocity vector \mathbf{v} according to

$$\hat{\mathbf{i}}_r = \frac{\mathbf{r}}{|\mathbf{r}|}$$

$$\hat{\mathbf{i}}_n = \frac{\mathbf{r} \times \mathbf{v}}{|\mathbf{r} \times \mathbf{v}|}$$

$$\hat{\mathbf{i}}_t = \hat{\mathbf{i}}_n \times \hat{\mathbf{i}}_r = \frac{(\mathbf{r} \times \mathbf{v}) \times \mathbf{r}}{|\mathbf{r} \times \mathbf{v}| |\mathbf{r}|}$$

For *unperturbed* two-body motion, $\mathbf{P} = 0$ and the first five equations of motion are simply $\dot{p} = \dot{f} = \dot{g} = \dot{h} = \dot{k} = 0$. Therefore, for two-body motion these modified equinoctial orbital elements are constant. The true longitude is often called the *fast variable* of this orbital element set.

Non-spherical Earth gravity

The non-spherical gravitational acceleration vector can be expressed as

$$\mathbf{g} = g_N \hat{\mathbf{i}}_N - g_r \hat{\mathbf{i}}_r$$

where

$$\hat{\mathbf{i}}_N = \frac{\hat{\mathbf{e}}_N - (\hat{\mathbf{e}}_N^T \hat{\mathbf{i}}_r) \hat{\mathbf{i}}_r}{\|\hat{\mathbf{e}}_N - (\hat{\mathbf{e}}_N^T \hat{\mathbf{i}}_r) \hat{\mathbf{i}}_r\|}$$

and

$$\hat{\mathbf{e}}_N = [0 \quad 0 \quad 1]^T$$

In these equations the north direction component is indicated by subscript N and the radial direction component is subscript r .

The contributions due to the *zonal* gravity effects of J_2, J_3, J_4 are as follows:

$$g_N = -\frac{\mu \cos \phi}{r^2} \sum_{k=2}^4 \left(\frac{R_e}{r}\right)^k P_k' J_k$$

$$g_r = -\frac{\mu}{r^2} \sum_{k=2}^4 (k+1) \left(\frac{R_e}{r}\right)^k P_k J_k$$

where

- μ = gravitational constant
- r = geocentric distance of the spacecraft
- R_e = equatorial radius of the Earth
- ϕ = geocentric latitude
- J_k = zonal gravity coefficient
- P_k = k^{th} order Legendre polynomial

For a zonal only Earth gravity model, the east component is identically zero.

Finally, the zonal gravity perturbation contribution is

$$\mathbf{a}_g = \mathbf{Q}^T \mathbf{g}$$

where $\mathbf{Q} = [\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n]$.

For J_2 effects only, the three components are as follows:

$$\Delta_{J_2} = -\frac{3\mu J_2 R_e^2}{2r^4} \left[1 - \frac{12(h \sin L - k \cos L)^2}{(1 + h^2 + k^2)^2} \right]$$

$$\Delta_{J_{2e}} = -\frac{12\mu J_2 R_e^2}{r^4} \left[\frac{(h \sin L - k \cos L)(h \cos L + k \sin L)}{(1 + h^2 + k^2)^2} \right]$$

$$\Delta_{J_{2n}} = -\frac{6\mu J_2 R_e^2}{r^4} \left[\frac{(1 - h^2 - k^2)(h \sin L - k \cos L)}{(1 + h^2 + k^2)^2} \right]$$

Secondary Body Perturbations

The general vector equation for secondary body perturbations such as the Moon or planets is given by

$$\mathbf{t} = -\sum_{j=1}^n \mu_j \left[\frac{\mathbf{d}_j}{d_j^3} + \frac{\mathbf{s}_j}{s_j^3} \right]$$

In this equation, \mathbf{s}_j is the vector from the primary body to the secondary body j , μ_j is the gravitational constant of the secondary body and $\mathbf{d}_j = \mathbf{r} - \mathbf{s}_j$, where \mathbf{r} is the position vector of the spacecraft relative to the primary body.

The perturbation due to secondary bodies in the modified equinoctial coordinate system is given by

$$\mathbf{a}_{TB} = \mathbf{Q}^T \mathbf{t}$$

where $\mathbf{Q} = [\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n]$.

Cartesian equations of motion

This part of the trajectory analysis implements a *special perturbation* technique which numerically solves the vector system of second-order, nonlinear differential equations of motion of a spacecraft using the method of order reduction. The second-order equations of motion are given by

$$\ddot{\mathbf{a}}(\vec{r}, \vec{v}, t) = \ddot{\vec{r}}(\vec{r}, \vec{v}, t) = \ddot{\mathbf{a}}_g(\vec{r}) + \ddot{\mathbf{a}}_s(\vec{r}, t) + \ddot{\mathbf{a}}_m(\vec{r}, t)$$

where

- t = dynamical time
- \vec{r} = inertial position vector of the spacecraft
- \vec{v} = inertial velocity vector of the spacecraft
- $\ddot{\mathbf{a}}_g$ = acceleration due to the Earth's gravity
- $\ddot{\mathbf{a}}_s$ = acceleration due to the Sun
- $\ddot{\mathbf{a}}_m$ = acceleration due to the Moon

After order reduction, the system of six first-order differential equations is defined by

$$\dot{y}_1 = v_x = y_4$$

$$\dot{y}_2 = v_y = y_5$$

$$\dot{y}_3 = v_z = y_6$$

$$\dot{y}_4 = -\mu \frac{r_x}{r^3} \left\{ 1 + \frac{3 J_2 r_{eq}^2}{2 r^2} \left(1 - \frac{5 r_z^2}{r^2} \right) \right\} + a_{m_x} + a_{s_x} + a_{T_x}$$

$$\dot{y}_5 = -\mu \frac{r_y}{r^3} \left\{ 1 + \frac{3 J_2 r_{eq}^2}{2 r^2} \left(1 - \frac{5 r_z^2}{r^2} \right) \right\} + a_{m_y} + a_{s_y} + a_{T_y}$$

$$\dot{y}_6 = -\mu \frac{r_z}{r^3} \left\{ 1 + \frac{3 J_2 r_{eq}^2}{2 r^2} \left(3 - \frac{5 r_z^2}{r^2} \right) \right\} + a_{m_z} + a_{s_z} + a_{T_z}$$

Propulsive thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)} \hat{\mathbf{u}}_T$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = [u_{T_r} \ u_{T_t} \ u_{T_n}]^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system. *The components of this unit vector are the control variables for this problem.*

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g I_{sp}$ is also called the *exhaust velocity*. This differential equation and the modified equinoctial differential equations are included in the right-hand-side subroutine required by SOCS.

The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}t$ where m_{sc_i} is the initial mass of the spacecraft.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as follows:

$$u_{T_r} = \sin \theta$$

$$u_{T_t} = \cos \theta \cos \psi$$

$$u_{T_n} = \cos \theta \sin \psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

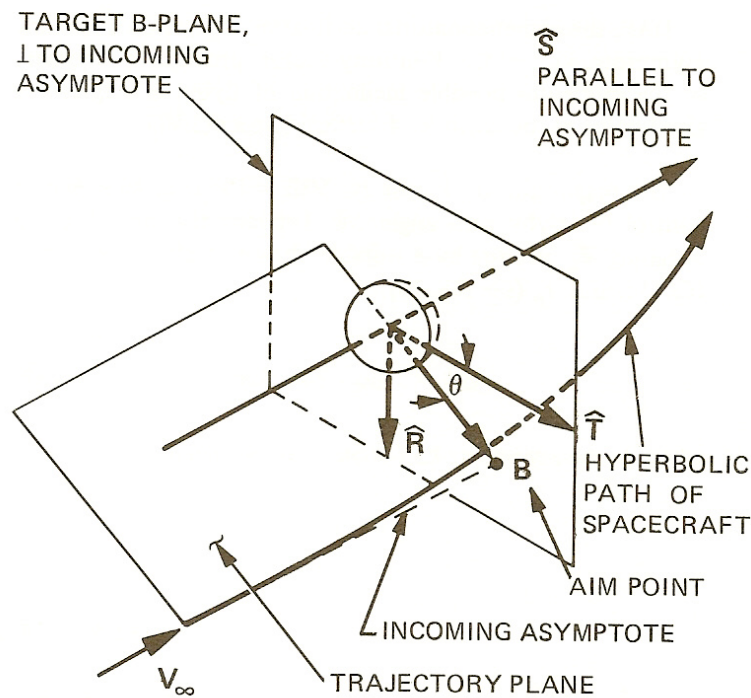
$$\theta = \sin^{-1}(u_{T_r})$$

$$\psi = \tan^{-1}(u_{T_n}, u_{T_t})$$

The pitch angle is positive above the “local horizontal” and the yaw angle is positive in the direction of the angular momentum vector.

The B-plane

The derivation of B-plane coordinates is described in the classic JPL reports, “A Method of Describing Miss Distances for Lunar and Interplanetary Trajectories” and “Some Orbital Elements Useful in Space Trajectory Calculations”, both by William Kizner. The following diagram illustrates the fundamental geometry of the B-plane coordinate system.



The arrival asymptote unit vector \hat{S} is given by

$$\hat{S} = \begin{Bmatrix} \cos \delta_{\infty} \cos \alpha_{\infty} \\ \cos \delta_{\infty} \sin \alpha_{\infty} \\ \sin \delta_{\infty} \end{Bmatrix}$$

where δ_∞ and α_∞ are the declination and right ascension of the asymptote of the incoming hyperbola.

The following computational steps summarize the calculation of the *predicted* B-plane vector from a moon-centered position vector \mathbf{r} and velocity vector \mathbf{v} at closest approach.

angular momentum vector

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}$$

radius rate

$$\dot{r} = \frac{\mathbf{r} \cdot \mathbf{v}}{|\mathbf{r}|}$$

semiparameter

$$p = \frac{h^2}{\mu}$$

semimajor axis

$$a = \frac{r}{\left(2 - \frac{rv^2}{\mu}\right)}$$

orbital eccentricity

$$e = \sqrt{1 - p/a}$$

true anomaly

$$\cos \theta = \frac{p - r}{er}$$

$$\sin \theta = \frac{\dot{r}h}{e\mu}$$

B-plane magnitude

$$B = \sqrt{p|a|}$$

fundamental vectors

$$\hat{\mathbf{z}} = \frac{r\mathbf{v} - \dot{r}\mathbf{r}}{h}$$

$$\hat{\mathbf{p}} = \cos \theta \hat{\mathbf{r}} - \sin \theta \hat{\mathbf{z}}$$

$$\hat{\mathbf{q}} = \sin \theta \hat{\mathbf{r}} + \cos \theta \hat{\mathbf{z}}$$

S vector

$$\mathbf{S} = -\frac{a}{\sqrt{a^2 + b^2}} \hat{\mathbf{p}} + \frac{b}{\sqrt{a^2 + b^2}} \hat{\mathbf{q}}$$

B vector

$$\mathbf{B} = \frac{b^2}{\sqrt{a^2 + b^2}} \hat{\mathbf{p}} + \frac{ab}{\sqrt{a^2 + b^2}} \hat{\mathbf{q}}$$

T vector

$$\mathbf{T} = \frac{(S_y^2, -S_x^2, 0)^T}{\sqrt{S_x^2 + S_y^2}}$$

R vector

$$\mathbf{R} = \mathbf{S} \times \mathbf{T} = (-S_z T_y, S_z T_x, S_x T_y - S_y T_x)^T$$

Geocentric-to-selenocentric coordinate transformation

This section describes the transformation of coordinates between the Earth mean equator and equinox of J2000 (EME2000) and lunar mean equator and IAU node of epoch coordinate systems. This transformation is used to compute the selenocentric orbital inclination and B-plane coordinates at lunar encounter.

A unit vector in the direction of the pole of the moon can be determined from

$$\hat{\mathbf{p}}_{Moon} = \begin{bmatrix} \cos \alpha_p \cos \delta_p \\ \sin \alpha_p \cos \delta_p \\ \sin \delta_p \end{bmatrix}$$

The right ascension and declination of the lunar pole in the EME2000 coordinate system are given by the following expressions

$$\begin{aligned} \alpha_p = & 269.9949 + 0.0031T - 3.8787 \sin E1 - 0.1204 \sin E2 \\ & + 0.0700 \sin E3 - 0.0172 \sin E4 + 0.0072 \sin E6 \\ & - 0.0052 \sin E10 + 0.0043 \sin E13 \end{aligned}$$

$$\begin{aligned} \delta_p = & 66.5392 + 0.0130T + 1.5419 \cos E1 + 0.0239 \cos E2 \\ & - 0.0278 \cos E3 + 0.0068 \cos E4 - 0.0029 \cos E6 \\ & + 0.0009 \cos E7 + 0.0008 \cos E10 - 0.0009 \cos E13 \end{aligned}$$

where T is the time in Julian centuries given by $T = (JD - 2451545.0) / 36525$ and JD is the TDB Julian Date.

The trigonometric arguments, in degrees, for these equations are

$$\begin{aligned}
E1 &= 125.045 - 0.0529921d \\
E2 &= 250.089 - 0.1059842d \\
E3 &= 260.008 + 13.0120009d \\
E4 &= 176.625 + 13.3407154d \\
E6 &= 311.589 + 26.4057084d \\
E7 &= 134.963 + 13.0649930d \\
E10 &= 15.134 - 0.1589763d \\
E13 &= 25.053 + 12.9590088d
\end{aligned}$$

where $d = JD - 2451545$ is the number of days since January 1.5, 2000. These equations are given in “Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000”, *Celestial Mechanics and Dynamical Astronomy*, **82**: 83-110, 2002.

The unit vector in the x-axis direction of this selenocentric coordinate system is given by

$$\hat{\mathbf{x}} = \hat{\mathbf{z}} \times \hat{\mathbf{p}}_{Moon}$$

where $\hat{\mathbf{z}} = [0 \ 0 \ 1]^T$. The unit vector in the y-axis direction can be determined using

$$\hat{\mathbf{y}} = \hat{\mathbf{p}}_{Moon} \times \hat{\mathbf{x}}$$

Finally, the components of the matrix that transforms coordinates from the EME2000 system to the moon-centered (selenocentric) mean equator and IAU node of epoch system are as follows:

$$\mathbf{M} = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \\ \hat{\mathbf{p}}_{Moon} \end{bmatrix}$$

Circularization delta-v

The impulsive delta-v required to circularize the spacecraft’s trajectory at closest approach to the moon can be computed from

$$\Delta v = v_p - \sqrt{\frac{\mu_m}{r_p}} = v_p - v_{lc}$$

where v_p is the velocity of the incoming hyperbola at periapsis, r_p is the periapsis radius at closest approach, and μ_m is the gravitational constant of the moon. For capture into an elliptical orbit at the moon, the impulsive delta-v is determined using

$$\Delta v = v_p - \sqrt{\frac{2\mu_m}{r_p} + \frac{\mu_m}{a}}$$

where a is the semimajor axis of the final elliptical orbit.

A note about targeting the lunar inclination

The range of orbital inclinations possible at closest approach to the moon is a function of the declination of the incoming hyperbola. This range is governed by the following constraint

$$i > |\delta_{\infty}|$$

where i is the selenocentric inclination of the final lunar orbit and δ_{∞} is the selenocentric declination of the incoming hyperbola.

References and Bibliography

“Lunar Trajectories”, NASA TN D-866, August 1961.

“Earth-Moon Trajectories”, JPL Technical Report No. 32-503, May 1, 1964.

“Three-Dimensional Lunar Trajectories”, V. A. Egorov, Mechanics of Space Flight Series, Israel Program for Scientific Translations, Jerusalem 1969.

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

“Optimal Low Thrust Trajectories to the Moon”, John T. Betts and Sven O. Erb, *SIAM Journal on Applied Dynamical Systems*, Vol. 2, No. 2, pp. 144-170, 2003.

“Integrated Algorithm for Lunar Transfer Trajectories Using a Pseudostate Technique”, R. V. Ramanan, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 25, No. 5, September-October 2002, pp. 946-952.

“Nonimpact Lunar Transfer Trajectories Using the Pseudostate Technique”, R. V. Ramanan and V. Adimurthy, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 28, No. 2, March-April 2005, pp. 217-225.

“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.

“Earth-Moon Trajectories, 1964-69”, R. J. Richard, V. C. Clarke, Jr., R. Y. Roth and W. E. Kirhofer, JPL Technical Report No. 32-503, May 1, 1964.

APPENDIX A

Compiling and Running the Software

This appendix describes how to compile and run the `fb_lsocs` computer program. This software was created using version 6.3.3 of SOCS and Compaq Visual Fortran.

A DOS/Windows version of `fb_lsocs` using Compaq Visual Fortran version 6.6C can be created with the following command:

```
f132 /arch:host fb_lsocs.f *.for c:\socs\socs633.lib advapi32.lib
```

This command assumes the SOCS library is located in the subdirectory `c:\socs`.

An input file created by the user can be run from the command line or a simple batch file with a statement similar to the following:

```
fb_lsocs lsocs1.in
```

If the software is executed without an input file on the command line, the computer program will display the following information screen and file name prompt:

```
*****
*           program fb_lsocs           *
*                                         *
*   finite-burn lunar trajectory       *
*           optimization with SOCS     *
*                                         *
*           December 19, 2005         *
*****
```

```
please input the name of the simulation definition file
```

The source code that reads the name of an input file included on the command line is

```
c   if present, use command line argument #1 for input file
    call getarg(1, inputfname$, istatus)
```

The source code that creates the file name input prompt is as follows:

```
c   clear screen

    isys = system("cls")

    if (istatus .eq. -1) then
c     *****
c     input filename not on command line
c     request name of simulation definition input file
c     *****

    print *, ' '
    print *, ' '
```

```

print *, ' *****'
print *, ' *           program fb_lsocs           *'
print *, ' *                                           *'
print *, ' *   finite-burn lunar trajectory   *'
print *, ' *           optimization with SOCS           *'
print *, ' *                                           *'
print *, ' *           December 19, 2005           *'
print *, ' *****'
print *, ' '
print *, ' '

print *,
&      'please input the name of the simulation definition file'

      read (*, *) inputfname$
end if

```

If your compiler does not accept input from a command line, you will have to modify this source code for your particular Fortran compiler. You may also choose to eliminate the code that accepts a command line input file. Please note also that your compiler may have a different command to clear the screen.

APPENDIX B

Contents of the Simulation Summary and CSV Files

This appendix is a brief summary of the information contained in the simulation summary screen displays and CSV data file produced by the fb_lsocs software.

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 (min) = orbital period in minutes. If the orbit is hyperbolic, the period is undefined and a value of 0 is displayed.
rx (km) = x-component of the eci position vector in kilometers
ry (km) = y-component of the eci position vector in kilometers
rz (km) = z-component of the eci position vector in kilometers
rmag (km) = geocentric position magnitude in kilometers
vx (kps) = x-component of the eci velocity vector in kilometers/second
vy (kps) = y-component of the eci velocity vector in kilometers/second
vz (kps) = z-component of the eci velocity vector in kilometers/second
vmag (kps) = velocity vector scalar magnitude in kilometers/seconds
deltav = scalar magnitude of the TLI maneuver in meters/seconds

The comma-separated-variable disk file is created by the odeprt subroutine and contains the following information:

time (hrs) = time since ignition in hours
rx (km) = x-component of eci position vector in kilometers
ry (km) = y-component of eci position vector in kilometers
rz (km) = z-component of eci position vector in kilometers
rmag (km) = geocentric radius magnitude in kilometers
vx (km/sec) = x-component of eci velocity vector in kilometers per second

vy (km/sec) = y-component of eci velocity vector in kilometers per second
vz (km/sec) = z-component of eci velocity vector in kilometers per second
vmag (km/sec) = scalar velocity vector in kilometers per second
semimajor axis (km) = semimajor axis in kilometers
eccentricity = orbital eccentricity (non-dimensional)
inclination (deg) = orbital inclination in degrees
arg of perigee (deg) = argument of perigee in degrees
raan (deg) = right ascension of the ascending node in degrees
true anomaly (deg) = true anomaly in degrees
fpa (deg) = flight path angle in degrees
seleno radius (km) = selenocentric radius of the spacecraft in kilometers
rm2sc-x (km) = x-component of selenocentric position vector in kilometers
rm2sc-y (km) = y-component of selenocentric position vector in kilometers
rm2sc-z (km) = z-component of selenocentric position vector in kilometers
rm2scm (km) = selenocentric position magnitude in kilometers
pmee = orbital semiparameter in kilometers
fmee = modified equinoctial orbital element = $\text{ecc} * \cos(\text{argper} + \text{raan})$
gmee = modified equinoctial orbital element = $\text{ecc} * \sin(\text{argper} + \text{raan})$
hmee = modified equinoctial orbital element = $\tan(i/2) * \cos(\text{raan})$
xkmee = modified equinoctial orbital element = $\tan(i/2) * \sin(\text{raan})$
xlmee = modified equinoctial orbital element = orbital true longitude in degrees
rmoon-x (km) = x-component of the moon's geocentric position vector in kilometers
rmoon-y (km) = y-component of the moon's geocentric position vector in kilometers
rmoon-z (km) = z-component of the moon's geocentric position vector in kilometers

APPENDIX C

Fortran Functions and Subroutines

This appendix is a brief summary of the major Fortran functions and subroutines included in the fb_lsocs computer program.

- fb_lsocs.f** - SOCS main executive program
- atan3.for** - four quadrant inverse tangent function
- csint.for** - cubic spline integration subroutine
- display.for** - subroutine that displays the simulation summary
- eci2mee.for** - convert eci position and velocity vectors to modified equinoctial orbital elements subroutine
- eci2orb.for** - convert eci position and velocity vectors to classical orbital elements subroutine
- findca.for** - subroutine to compute closest approach to the moon
- fpaobj.for** - selenocentric flight path angle subroutine
- fpasub.for** - subroutine that converts state vector to flight path angle
- gdate.for** - convert Julian date to calendar date subroutine
- geo_eqm.for** - first-order equations of motion subroutine
- linput.for** - read and echo a line of text from an input file subroutine
- mee2eci.for** - convert modified elements to eci state vector subroutine
- mm2000.for** - lunar coordinates transformation matrix subroutine
- odeigs.for** - SOCS initial guess subroutine
- odeinp.for** - SOCS simulation input subroutine
- odepf.for** - SOCS point functions subroutine
- odeprt.for** - SOCS print subroutine - creates comma-separated-variable file
- oderhs.for** - SOCS subroutine that evaluates the equations of motion and any algebraic equations
- orb2eci.for** - convert classical orbital elements to eci position and velocity vector subroutine
- readfpn.for** - read and echo floating point number from input file subroutine
- readint.for** - read and echo integer from input file subroutine
- readtext.for** - read and echo text from input file subroutine
- rkf78.for** - Runge-Fehlberg-Kutta (RK78) numerical integration subroutine

rkf78cn.for - evaluate RKF78 integration coefficients subroutine
rv2bp.for - convert position and velocity to b-plane coordinates subroutine
sv2000.for - lunar and solar ephemeris subroutine
ueci2umee.for - convert eci unit vector to modified unit vector subroutine
twobody2.for - two-body orbit propagation subroutine
utility.for - number and text manipulation functions and subroutines
uvector.for - unit vector subroutine
vcross.for - vector cross product subroutine
vdot.for - vector dot product subroutine
vecmag.for - vector scalar magnitude function
xmod.for - modulo 2 pi function

APPENDIX D

Example Fortran Subroutine

This appendix contains a Fortran 77 routine that illustrates typical programming conventions used in the fb_lsocs software. This subroutine is the point function routine required by the SOCS software.

```
      subroutine odepf(iphase, iphend, time, y, ny, p, np,
&                    ptf, nf, iferr)
c     lunar trajectory optimization point functions
c     *****
      implicit double precision (a-h, o-z)
      include 'socscom1.inc'
      dimension y(ny), ptf(nf), p(np), reci(3), veci(3)
      dimension hv(3), rmoon(3), vmoon(3), tmatrix(3, 3)
      dimension rtmp(3), vtmp(3), oev(6)
      dimension bplane(12), tv(3), rv(3)
      iferr = 0
c     local circular velocity (kilometers/second)
      vlc = sqrt(emu / req)
      if (iphase .eq. 1) then
c     -----
c     phase 1 - TLI propulsive maneuver
c     load modified equinoctial elements into local variables
c     -----
      pmee = y(1)
      fmee = y(2)
      gmee = y(3)
      hmee = y(4)
      xkmee = y(5)
      xlmee = y(6)
c     convert modified equinoctial elements to eci state vector
      call mee2eci(pmee, fmee, gmee, hmee, xkmee, xlmee, reci,
&                veci, smovrp, tani2s, cosl, sinl, wmee, radius,
&                hsmks, ssqrd)
      else
c     -----
```

```

c      phase 2 - coast to lunar encounter
c      load cartesian state vector into local variables
c      -----

      do i = 1, 3
        reci(i) = y(i)

        veci(i) = y(i + 3)
      end do

c      convert eci state vector to modified equinoctial orbital elements

      call eci2mee(reci, veci, pmee, fmee, gmee, hmee,
&                xkmee, xlmee)

      end if

c      extract current spacecraft mass (kilograms)

      xmass = y(7)

      if (iphase .eq. 1 .and. ioevl .eq. 2 .and. iphend .eq. -1) then
c      -----
c      park orbit inclination and raan
c      -----

c      compute classical orbital elements

      call eci2orb(emu, reci, veci, oev)

c      cosine(inclination)

      call vcross(reci, veci, hv)

      hmag = vecmag(hv)

      ptf(1) = hv(3) / hmag

c      RAAN (radians)

      ptf(2) = oev(5)

      end if

      if (iphase .eq. 1 .and. iphend .eq. +1) then
c      -----
c      "working" eci state vector at the
c      end of the TLI propulsive maneuver
c      -----

      do i = 1, 3
        ptf(i) = reci(i)

        ptf(i + 3) = veci(i)
      end do

      end if

      if (iphase .eq. 2 .and. iphend .eq. -1) then

```

```

c -----
c "working" eci state vector at the
c beginning of the coast to lunar encounter
c -----

do i = 1, 3
  ptf(i) = reci(i)

  ptf(i + 3) = veci(i)
end do

end if

if (iphase .eq. 2 .and. iphend .eq. +1) then
c -----
c selenocentric mission constraints at SOI
c -----

xjdsOI = xjdate0 + time / 86400.0d0

call sv2000(xjdsOI, 10, 3, rmoon, vmoon)

c state vector from the moon to the spacecraft

do i = 1, 3
  rtmp(i) = reci(i) - rmoon(i)

  vtmp(i) = veci(i) - vmoon(i)
end do

c state vector in mm2000 coordinate system

call mm2000 (xjdsOI, tmatrix)

call matxvtr(tmatrix, rtmp, rm2sc)

call matxvtr(tmatrix, vtmp, vm2sc)

c -----
c selenocentric distance point function
c -----

ptf(1) = vecmag(rm2sc)

c find closest approach

call findca(xjdsOI, rm2sc, vm2sc, icaerr)

if (icaerr .eq. 1) then
c error check - close approach not found

  iferr = 1

  return
end if

c compute b-plane coordinates

call rv2bp(xmmu, rca, vca, bplane, tv, rv, ibperr)

```

```

      if (itarget .eq. 1) then
c         periapsis radius point function (kilometers)
c         -----
          ptf(2) = bplane(5)
c
c         cosine of orbital inclination point function
c         -----
          call vcross(rca, vca, hv)
          hmag = vecmag(hv)
          ptf(3) = hv(3) / hmag
c
c         else
c         user-defined b-plane targets
c         -----
          ptf(2) = bplane(7) - bpuser(2)
          ptf(3) = bplane(8) - bpuser(1)
c
c         end if
c
c     end if
c
c     return
c     end

```