

Program tli_soc

Impulsive TLI Delta-V Optimization with SOCS

This document is the user's manual for a Fortran computer program called `tli_soc` 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 one impulse, translunar injection (TLI) trajectory optimization problem. The software attempts to minimize the scalar magnitude of the impulsive TLI delta-v vector while targeting to either a user-defined periapsis radius and orbital inclination or B-plane coordinates of the arrival hyperbola relative to the moon.

The important features of this scientific simulation are as follows:

- rectangular, cartesian equations of motion
- sun and moon point-mass perturbations
- Earth high fidelity and lunar J_2 gravity models
- JPL DE421 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 `tli_soc` 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 `tli_soc`s 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 DE421 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.

```
*****
** impulsive TLI delta-v trajectory optimization
** 3-body geocentric motion
** data: lsocsl.in program: tli_soc.s.f
** February 2, 2006
*****
```

The software allows the user to specify an initial guess for the calendar date 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 search boundary should be set to $\pm 1.0D-6$.

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

```
TLI calendar date initial guess (month, day, year)
10,1,2008

TLI TDB time initial guess (hours, minutes, seconds)
0, 0, 0

TLI TDB time search boundary (hours)
-24.0d0, +24.0d0
```

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. Please note that the transfer time is the elapsed time from the TLI maneuver to the lunar sphere-of-influence. For a fixed transfer time, both the lower and upper bounds should be set to the value of the transfer time initial guess.

```
transfer time initial guess (hours)
96.0

transfer time lower bound (hours)
84.0d0

transfer time upper bound (hours)
```

120.0d0

The next two inputs define the altitude and orbital inclination of the circular park orbit. The orbital inclination should be with respect to the EME2000 system.

```
*****
circular park orbit characteristics (EME2000)
*****
```

```
altitude (kilometers)
185.2
```

```
orbital inclination (degrees)
28.5
```

The type of TLI maneuver to use for the initial guess is defined by the next input. Please see the technical discussion for information about this program option.

```
type of TLI maneuver
(1 = ascending, 2 = descending)
2
```

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 mean lunar equator and IAU node of epoch.

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

```
b dot r target (kilometers)
-5572.4203567d0
```

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

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 series of inputs define the types of perturbations to include during the numerical solution of the spacecraft's equations of motion. The first input is the name of the data file that contains the un-normalized Earth gravity model coefficients.

```
*****  
trajectory perturbations  
*****  
  
name of Earth gravity model data file  
egm96.dat  
  
order of Earth gravity model (zonals)  
8  
  
degree of Earth gravity model (tesserals)  
8  
  
include solar point-mass perturbations (1 = yes, 0 = no)  
1  
  
include lunar point-mass perturbations (1 = yes, 0 = no)  
1
```

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

```
*****  
* initial guess/restart option *  
*****  
1 = two-body Lambert algorithm  
2 = 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  
tli_socsl.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  
tli_socsl.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  
tli_socsl.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 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 tli_socS simulation for this example. The first part of the solution display summarizes the characteristics of the optimized two-body initial guess.

optimized two-body initial guess

time and conditions prior to TLI
(geocentric EME2000)

calendar date October 1, 2008

TDB time 00:00:00.000

TDB Julian date 2454740.5000000000000000

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.656333630000D+04	0.148452954789D-15	0.285000000000D+02	0.000000000000D+00
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.325175381768D+03	0.110971936813D+03	0.110971936813D+03	0.881955589276D+02
rx (km)	ry (km)	rz (km)	rmag (km)
0.114731803240D+04	0.576277319725D+04	0.292429304217D+04	0.656333630000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.737334687378D+01	0.214332629567D+01	-.133089585966D+01	0.779303378153D+01

delta-v vector and magnitude

deltav-x	-2976.359977402429649	meters/second
deltav-y	865.247447579105824	meters/second
deltav-z	-537.207956507512563	meters/second
deltav	3145.785156230496341	meters/second

time and conditions after TLI
(geocentric EME2000)

calendar date October 1, 2008

TDB time 00:00:00.000

TDB Julian date 2454740.5000000000000000

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.220832334152D+06	0.970279097375D+00	0.285000000000D+02	0.110971224040D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.325175381768D+03	0.712772403522D-03	0.110971936813D+03	0.172128872738D+05
rx (km)	ry (km)	rz (km)	rmag (km)
0.114731803240D+04	0.576277319725D+04	0.292429304217D+04	0.656333630000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.103497068512D+02	0.300857374325D+01	-.186810381616D+01	0.109388189373D+02

TLI-to-SOI estimate = 86.227757915854454 hours

The second part of the 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 the user-defined SOI radius and the moment of closest approach to the moon. The pitch and yaw angles of the impulsive maneuver are defined with respect to a local-vertical, local-horizontal (LVLH) system.

```

-----
tli_socsl - impulsive descending TLI maneuver
-----

input data file ==> tli_socsl.in

two-body Lambert initial guess

tzero                -22.443319847382419 hours
tfinal               90.475153933478353 hours

TLI delta-v vector and magnitude

delta-vx             -2971.157179812312734 meters/second
delta-vy              867.906664525759652 meters/second
delta-vz             -534.393880761709738 meters/second

deltav               3141.116646238092926 meters/second

pitch angle          8.327136703242297E-002 degrees
yaw angle            1.785226877466450E-004 degrees

-----
time and conditions prior to TLI
(geocentric EME2000)
-----

calendar date        September 30, 2008
TDB time             01:33:24.049
TDB Julian date      2454739.564861672930419

      sma (km)          eccentricity      inclination (deg)      argper (deg)
0.656333630000D+04    0.497349407171D-15    0.285000000000D+02    0.000000000000D+00

      raan (deg)        true anomaly (deg)      arglat (deg)          period (min)
0.325175381768D+03    0.110971936813D+03    0.110971936813D+03    0.881955589276D+02

      rx (km)           ry (km)                   rz (km)                rmag (km)
0.114731803240D+04    0.576277319725D+04    0.292429304217D+04    0.656333630000D+04

      vx (kps)          vy (kps)                   vz (kps)                vmag (kps)
-.737334687378D+01    0.214332629567D+01    -.133089585966D+01    0.779303378153D+01

-----
time and conditions after TLI
(geocentric EME2000)
-----

calendar date        September 30, 2008

```

TDB time 01:33:24.049

TDB Julian date 2454739.564861672930419

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.209002408385D+06	0.968596844467D+00	0.284999742460D+02	0.110923193798D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.325175522580D+03	0.486192665626D-01	0.110971813064D+03	0.158484453350D+05
rx (km)	ry (km)	rz (km)	rmag (km)
0.114731803240D+04	0.576277319725D+04	0.292429304217D+04	0.656333630000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.103445040536D+02	0.301123296020D+01	-.186528974042D+01	0.109341480633D+02

orbital energy -1.907157169052667 (km/sec)**2

time and conditions at lunar SOI
(geocentric EME2000)

calendar date October 4, 2008

TDB time 18:28:30.554

TDB Julian date 2454744.269798080436885

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.202983577930D+06	0.998678784966D+00	0.122912703784D+03	0.148013155776D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.236365606783D+03	0.179696494460D+03	0.327709650236D+03	0.151687951165D+05
rx (km)	ry (km)	rz (km)	rmag (km)
-.909500969314D+05	-.347089647325D+06	-.180034139303D+06	0.401441695301D+06
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.573822549296D-01	-.131660076071D+00	-.388564061540D-01	0.148784808065D+00

time and conditions at lunar SOI
(selenocentric - mean lunar equator and IAU node of epoch)

calendar date October 4, 2008

TDB time 18:28:30.554

TDB Julian date 2454744.269798080436885

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.824411721973D+04	0.122218672289D+01	0.902723347440D+02	0.145724943422D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.351199349525D+03	0.226765425570D+03	0.124903689925D+02	0.000000000000D+00
rx (km)	ry (km)	rz (km)	rmag (km)
0.241170097899D+05	-.375979649312D+04	0.540682650307D+04	0.250000000742D+05

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.981041846517D+00	0.152062141692D+00	-.368903297309D-01	0.993441943941D+00

time and conditions at lunar closest approach
(selenocentric - mean lunar equator and IAU node of epoch)

calendar date	October 5, 2008
TDB time	00:21:03.599
TDB Julian date	2454744.514624984934926

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.820109602690D+04	0.122411638755D+01	0.899999998425D+02	0.145584967968D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.351113891784D+03	0.203555499614D-12	0.145584967968D+03	0.000000000000D+00
rx (km)	ry (km)	rz (km)	rmag (km)
-.149808667881D+04	0.234222001243D+03	0.103880720730D+04	0.183800001552D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.136010776854D+01	0.212649345096D+00	-.200938775772D+01	0.243572413193D+01

b-plane coordinates of incoming hyperbola
(selenocentric - mean lunar equator and IAU node of epoch)

b-magnitude	5790.118591743281286 kilometers
b dot r	-5790.118591743281286
b dot t	1.591802071132520E-005
theta	270.000000157515842 degrees
v-infinity	773.189861546666521 meters/second
r-periapsis	1838.000015516909116 kilometers
decl-asy	-8.077347617909355E-001 degrees
rasc-asy	171.113891785941149 degrees
flight path angle	1.280388843655004E-013 degrees
TLI-to-SOI time	112.918473780155182 hours
TLI-to-CA time	118.794319488108158 hours 4.949763312004507 days

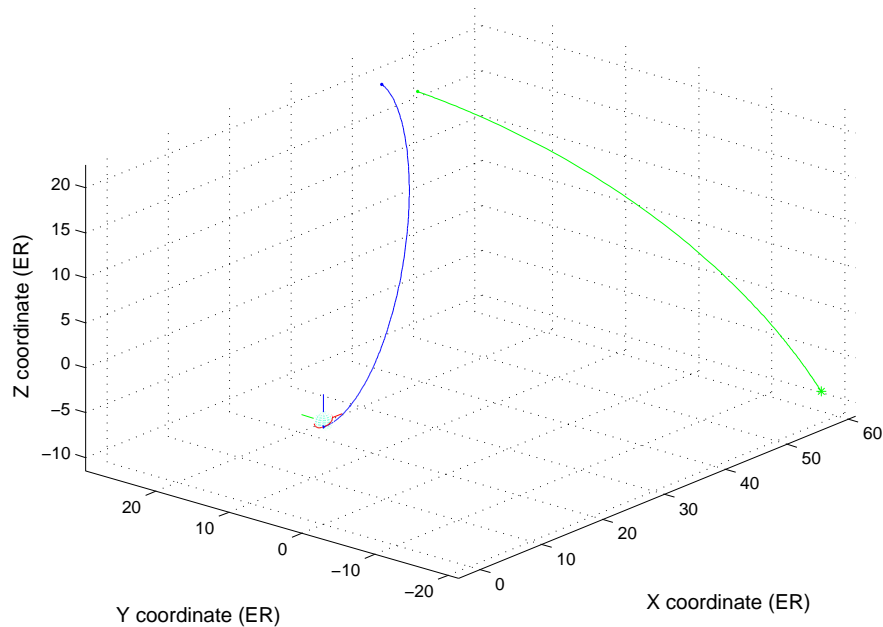
In this display, the TLI-to-SOI time is the elapsed time from translunar injection until the spacecraft reaches the lunar sphere-of-influence. The TLI-to-CA time is the elapsed time from translunar injection until closest approach to the moon which corresponds to periapsis of the encounter hyperbola.

The tli_socsv 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.

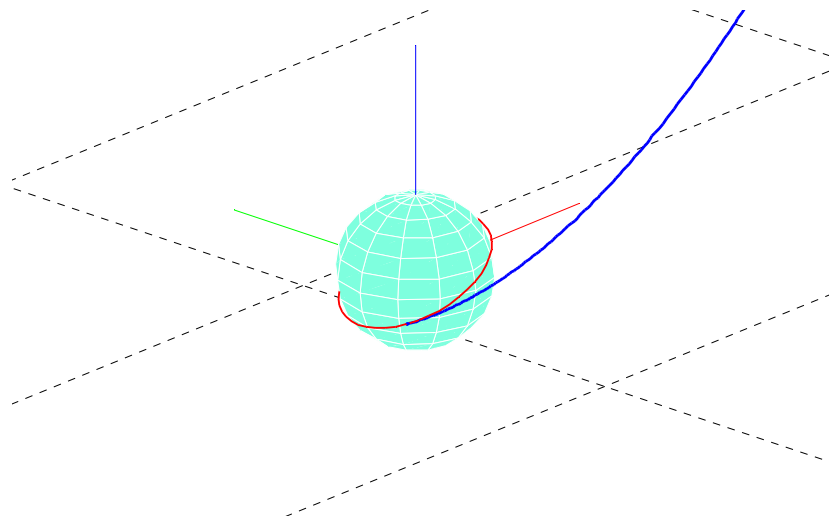
The `tli_soc`s software package also includes a MATLAB script called `lplot.m` that can be used to create trajectory graphic displays using these three data files. The interactive graphic features of MATLAB allow the user to rotate and zoom the displays. These capabilities allow the user to interactively 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 `lunar1` 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 the TLI 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 green and the z-axis blue.

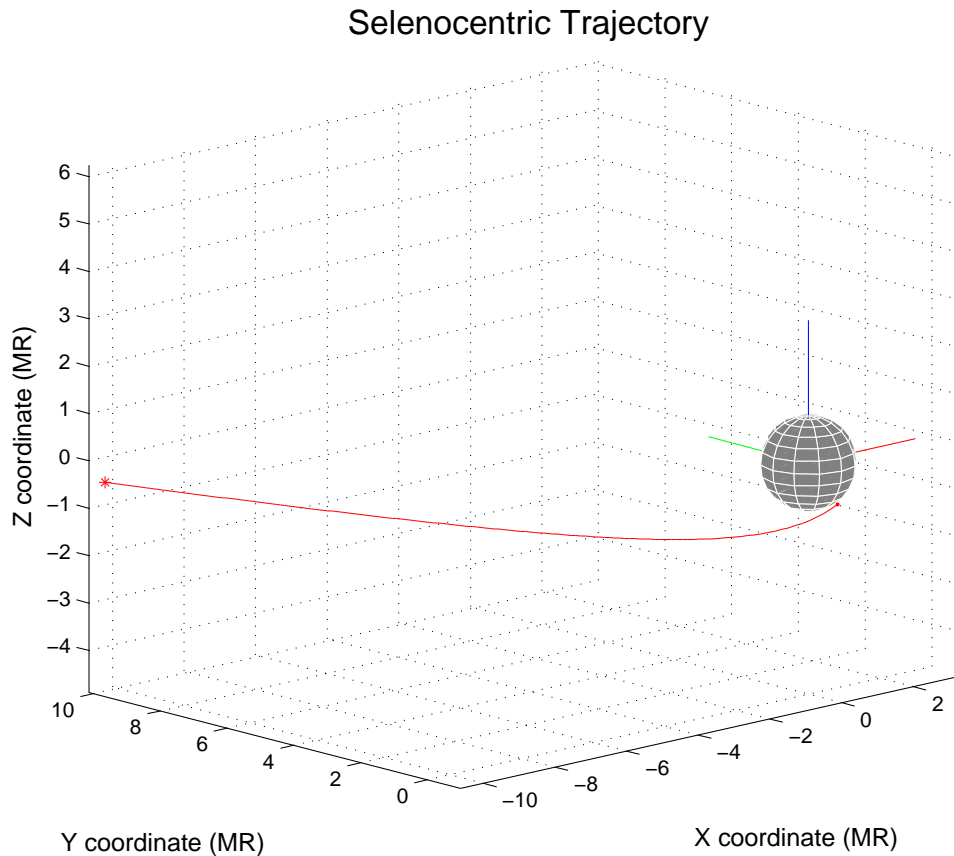
Geocentric Transfer Trajectory



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 moon's sphere-of-influence (SOI). The coordinate units are lunar radii (MR). The entry into the SOI is marked with an asterisk and the display is labeled with a selenocentric, inertial coordinate system. The x-axis is red, the y-axis green and the z-axis blue.



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.

Initial Guess

There are two types of initial guess algorithms used in the `tli_soc`s software. The first numerical method uses an initial guess for the components of the impulsive TLI Δv vector determined from the solution of the two-body lunar Lambert problem. Using this initial guess, the software then numerically integrates the geocentric spacecraft equations of motion until the spacecraft reaches the user-defined SOI selenocentric distance. The time that the vehicle reaches the lunar SOI is determined using Brent's root-finding algorithm embedded within a Runge-Kutta-Fehlberg variable step size integrator. Finally, the position and velocity vectors at each grid point are determined by SOCS by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2`.

The binary data file initialization, option 2, uses state vectors read from the file contents created by a previous execution of `tli_soc`s. Both initial guess options use the perturbations defined by the user in the main input file.

The TLI delta-v vector is determined by using the SOCS dense nonlinear programming routine (`hdnlpd`) to find the park orbit true anomaly that minimizes the magnitude of the required TLI delta-v. Here's the Fortran source code that performs this optimization.

```

c      initial guess for park orbit true anomaly at TLI
      xbar(1) = 0.0d0

c      bounds on true anomaly
      xlwr(1) = -pi
      xupr(1) = +pi

c      SOCS NLP setup
      istatv(1) = 0

      call hhsnlp('ioflag = 5')
      call hhsnlp('nitmax = 5000')
      call hhsnlp('maxnfe = 500000')

      call hdnlpd(lunfunc, xbar, xlwr, xupr, istatv,
&                vecnu, ndim, fbar, cbar, clwr, cupr,
&                istatc, veclam, mcon, hold, nhold,
&                ihold, nihold, needed, iernlp)

```

Here's the source code that determines the two-body, Lambert TLI delta-v magnitude.

```

      subroutine lunfunc(xbar, ndim, mcon, quant, iferr)

c      two body lunar trajectory objective function

c      input

c      control variable

c      x(1) = current value for true anomaly at TLI (radians)

c      output

c      fx = delta-v objective function

c      *****

      implicit double precision (a-h, o-z)

      include 'socscom1.inc'

      dimension xbar(ndim), quant(mcon + 1)

      dimension statev(12, 11), rmoon(3), vmoon(3)

```

```

    iferr = 0

    direct = 1.0d0

    revmax = 0.0d0

    do i = 1, 3
        deltav(i) = 0.0d0
    end do

c   current park orbit true anomaly (radians)

    oevpo(6) = xbar(1)

c   compute state vector of park orbit

    call orb2eci(emu, oevpo, rpo, vpo)

c   compute position vector of the moon at arrival

    xjdate = xjdate0 + tof / 86400.0d0

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

c   solve lambert's problem

    call lambfunc(emu, rpo, rmoon, tof,
&                direct, revmax, statev, nsol)

c   transfer orbit state vector at TLI

    do i = 1, 3
        rto(i) = statev(i, 1)

        vto(i) = statev(i + 3, 1)
    end do

c   compute delta-v components

    do i = 1, 3
        deltav(i) = vto(i) - vpo(i)
    end do

c   scalar objective function (kps)

    quant(mcon + 1) = vecmag(deltav)

    return
end

```

During this optimization, the TLI calendar date, time and all the remaining orbital elements of the park orbit are fixed. The value for the right ascension of the ascending node (RAAN) is computed using the algorithm described in the next section.

Park orbit RAAN

For a given TLI calendar date, there are two possible locations on the initial park orbit at which to perform the propulsive maneuver. One opportunity occurs during the ascending part of the park orbit and the other during the descending motion. The park orbit RAAN Ω_p at these two locations can be

determined from spherical trigonometry relationships involving the park orbit inclination and the geocentric right ascension and declination of the moon at encounter. The equations implemented in this computer program are as follows:

ascending

$$\Omega_p = -180^\circ + \alpha_m + \sin^{-1} \left(\frac{\tan \delta_m}{\tan i_p} \right)$$

descending

$$\Omega_p = \alpha_m - \sin^{-1} \left(\frac{\tan \delta_m}{\tan i_p} \right)$$

where

α_m = right ascension of the moon at encounter

δ_m = declination of the moon at encounter

i_p = park orbit inclination

These opportunities are valid whenever $|\delta_m| \leq i_p$ and $i_p \neq 90$ or 180 degrees.

(1) TLI maneuver time and transfer time bounds

The software allows the user to specify an initial guess for the TLI maneuver calendar date and bounds on the actual date found during the optimization process. For any guess for launch time t_L and user-defined TLI time search bound Δt_L , the TLI time t is constrained as follows:

$$t_{TLI} - \Delta t_L \leq t \leq t_{TLI} + \Delta t_L$$

The same type of inequality is enforced for the lunar transfer time. For a fixed TLI time or transfer time, the search bounds are set to $\pm 1.0D-6$.

(2) Performance index – minimize TLI delta-v

The objective function or performance index J for this simulation is the scalar magnitude of the impulsive TLI maneuver. For this classic trajectory optimization problem, this index is simply

$$J = \Delta V$$

where ΔV is the scalar magnitude of the TLI delta-v. The value of the `maxmin` indicator in SOCS tells the software whether the user is minimizing or maximizing the performance index.

The components of the TLI impulsive delta-v vector are the optimization parameters used by SOCS to solve this trajectory problem.

(3) Point functions – position and velocity vector “matching” at TLI

For any TLI maneuver time t_{TLI} , the optimal solution satisfies the following state vector boundary conditions (equality constraints) at the TLI maneuver time:

$$\begin{aligned}\mathbf{r}_{to}(t_{TLI}) - \mathbf{r}_{po}(t_{TLI}) &= 0 \\ \mathbf{v}_{to}(t_{TLI}) - \{ \mathbf{v}_{po}(t_{TLI}) + \Delta\mathbf{v}_{TLI} \} &= 0\end{aligned}$$

where \mathbf{r}_{po} and \mathbf{v}_{po} are the geocentric, inertial position and velocity vectors of the park orbit at the TLI time t_{TLI} , \mathbf{r}_{to} and \mathbf{v}_{to} are the geocentric, inertial position and velocity vectors of the lunar transfer trajectory at the maneuver time, and $\Delta\mathbf{v}_{TLI}$ is the *impulsive* delta-v vector required at trans-lunar injection. These point functions constrain the position vector of the transfer orbit to the park orbit position vector, and the transfer orbit velocity vector to the sum of the park orbit velocity vector and the optimized TLI delta-v vector.

(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

$$\begin{aligned}r_p - r_{ca} &= 0 \\ \cos i - \hat{\mathbf{h}}_z &= 0\end{aligned}$$

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 mean lunar equator 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 mean lunar equator and IAU node of epoch 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

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

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 `tli_soc`s computer program corresponds to the GLGM-1 value of `2.037448533865259d-4`.

Technical Discussion

This section provides additional details about the numerical algorithms implemented in this computer program. The computational methods discussed here include solving the two body Lambert problem, propagating the spacecraft's trajectory, computing the B-plane coordinates and calculating the geocentric-to-selenocentric coordinate transformation.

Solving the two body Lambert problem

Lambert's problem is concerned with the determination of an orbit that passes between two positions within a specified time-of-flight. This classic astrodynamics problem is also known as the orbital two-point boundary value problem (TPBVP).

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

$$tof = tof(r_i + r_f, c, a)$$

From the following form of Kepler's equation

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

we can write

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

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

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

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

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

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

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

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

With the two substitutions given by

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

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

the time equation becomes

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

From the elliptic relationships given by

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

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

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

and some more manipulation, we have the following equations:

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

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

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

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

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

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

With the use of the half angle formulas given by

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

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

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

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

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

Spacecraft geocentric equations of motion

The `tli_socs` computer program implements a *special perturbation* technique which numerically integrates the vector system of second-order, nonlinear differential equations of motion of a spacecraft given by

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

where

t = barycentric dynamical time

\mathbf{r} = inertial position vector of the spacecraft

\mathbf{a}_g = acceleration due to the Earth's gravity

\mathbf{a}_m = acceleration due to the Moon

\mathbf{a}_s = acceleration due to the Sun

This computer program uses a *spherical harmonic* representation of the Earth’s geopotential function given by

$$\Phi(r, \phi, \lambda) = \frac{\mu}{r} + \frac{\mu}{r} \sum_{n=1}^{\infty} C_n^0 \left(\frac{R}{r} \right)^n P_n^0(u) + \frac{\mu}{r} \sum_{n=1}^{\infty} \sum_{m=1}^n \left(\frac{R}{r} \right)^n P_n^m(u) [S_n^m \sin m\lambda + C_n^m \cos m\lambda]$$

where ϕ is the geocentric latitude of the spacecraft, λ is the geocentric east longitude of the spacecraft and $r = |\mathbf{r}| = \sqrt{x^2 + y^2 + z^2}$ is the geocentric distance of the spacecraft. In this expression the S 's and C 's are *unnormalized* harmonic coefficients of the geopotential, and the P 's are associated Legendre polynomials of degree n and order m with argument $u = \sin \phi$.

The software calculates the spacecraft’s acceleration due to the Earth’s gravity field with a vector equation derived from the gradient of the potential function expressed as $\mathbf{a}_g(\mathbf{r}, t) = \nabla \Phi(\mathbf{r}, t)$.

This acceleration vector is a combination of pure two-body or *point mass* gravity acceleration and the gravitational acceleration due to higher order nonspherical terms in the Earth’s geopotential. In terms of the Earth’s geopotential Φ , the inertial rectangular cartesian components of the spacecraft’s acceleration vector are as follows:

$$\ddot{x} = \left(\frac{1}{r} \frac{\partial \Phi}{\partial r} - \frac{z}{r^2 \sqrt{x^2 + y^2}} \frac{\partial \Phi}{\partial \phi} \right) x - \left(\frac{1}{x^2 + y^2} \frac{\partial \Phi}{\partial \lambda} \right) y$$

$$\ddot{y} = \left(\frac{1}{r} \frac{\partial \Phi}{\partial r} - \frac{z}{r^2 \sqrt{x^2 + y^2}} \frac{\partial \Phi}{\partial \phi} \right) y + \left(\frac{1}{x^2 + y^2} \frac{\partial \Phi}{\partial \lambda} \right) x$$

$$\ddot{z} = \left(\frac{1}{r} \frac{\partial \Phi}{\partial r} \right) z + \left(\frac{\sqrt{x^2 + y^2}}{r^2} \frac{\partial \Phi}{\partial \phi} \right)$$

The three partial derivatives of the geopotential with respect to r, ϕ, λ are given by

$$\frac{\partial \Phi}{\partial r} = -\frac{1}{r} \left(\frac{\mu}{r} \right) \sum_{n=2}^N \left(\frac{R}{r} \right)^n (n+1) \sum_{m=0}^n (C_n^m \cos m\lambda + S_n^m \sin m\lambda) P_n^m(\sin \phi)$$

$$\frac{\partial \Phi}{\partial \phi} = \left(\frac{\mu}{r} \right) \sum_{n=2}^N \left(\frac{R}{r} \right)^n \sum_{m=0}^n (C_n^m \cos m\lambda + S_n^m \sin m\lambda) [P_n^{m+1}(\sin \phi) - m \tan \phi P_n^m(\sin \phi)]$$

$$\frac{\partial \Phi}{\partial \lambda} = \left(\frac{\mu}{r} \right) \sum_{n=2}^N \left(\frac{R}{r} \right)^n \sum_{m=0}^n m (S_n^m \cos m\lambda - C_n^m \sin m\lambda) P_n^m(\sin \phi)$$

where

R = radius of the Earth

r = geocentric distance of the spacecraft

S_n^m, C_n^m = harmonic coefficients

ϕ = geocentric latitude of the spacecraft = $\sin^{-1}(z/r)$

λ = longitude of the spacecraft = $\alpha - \alpha_g$

α = right ascension of the spacecraft = $\tan^{-1}(r_y/r_x)$

α_g = right ascension of Greenwich

Right ascension is measured positive east of the vernal equinox, longitude is measured positive east of Greenwich, and latitude is positive above the Earth's equator and negative below.

For $m = 0$, the coefficients are called *zonal* terms, when $m = n$ the coefficients are *sectorial* terms, and for $n > m \neq 0$ the coefficients are called *tesseral* terms.

The Legendre polynomials with argument $\sin \phi$ are computed using recursion relationships given by:

$$P_n^0(\sin \phi) = \frac{1}{n} [(2n-1) \sin \phi P_{n-1}^0(\sin \phi) - (n-1) P_{n-2}^0(\sin \phi)]$$

$$P_n^m(\sin \phi) = (2n-1) \cos \phi P_{n-1}^{m-1}(\sin \phi), \quad m \neq 0, m < n$$

$$P_n^m(\sin \phi) = P_{n-2}^m(\sin \phi) + (2n-1) \cos \phi P_{n-1}^{m-1}(\sin \phi), \quad m \neq 0, m = n$$

where the first few associated Legendre functions are given by

$$P_0^0(\sin \phi) = 1, \quad P_1^0(\sin \phi) = \sin \phi, \quad P_1^1(\sin \phi) = \cos \phi$$

and $P_i^j = 0$ for $j > i$.

The trigonometric arguments are determined from expansions given by

$$\sin m\lambda = 2 \cos \lambda \sin(m-1)\lambda - \sin(m-2)\lambda$$

$$\cos m\lambda = 2 \cos \lambda \cos(m-1)\lambda - \cos(m-2)\lambda$$

$$m \tan \phi = (m-1) \tan \phi + \tan \phi$$

The true-of-date position vector required in the previous equations is computed according to

$$\mathbf{r}_{TOD} = [PN] \mathbf{r}_{EME2000}$$

where $[PN]$ is the combined precession-nutation matrix.

The east longitude required in the gravity model calculations is computed from the x and y components of the true-of-date position vector according to

$$\lambda = \tan^{-1}(r_y, r_x) - \alpha_g$$

where α_g is the apparent right ascension of Greenwich at the time of interest.

The true-of-date gravity vector is converted to the EME2000 system for use in the equations of motion using the transpose of the combined precession-nutation matrix as follows

$$\mathbf{a}_{EME2000} = [PN]^T \mathbf{a}_{TOD}$$

Point mass acceleration of the sun and moon

The acceleration contribution of the moon represented by a *point mass* is given by

$$\bar{\mathbf{a}}_m(\vec{r}, t) = -\mu_m \left(\frac{\vec{r}_{m-b}}{|\vec{r}_{m-b}|^3} + \frac{\vec{r}_{e-m}}{|\vec{r}_{e-m}|^3} \right)$$

where

μ_m = gravitational constant of the moon

\vec{r}_{m-b} = position vector from the moon to the spacecraft

\vec{r}_{e-m} = position vector from the Earth to the moon

Likewise, the acceleration contribution of the sun represented by a *point mass* is given by

$$\vec{a}_s(\vec{r}, t) = -\mu_s \left(\frac{\vec{r}_{s-b}}{|\vec{r}_{s-b}|^3} + \frac{\vec{r}_{e-s}}{|\vec{r}_{e-s}|^3} \right)$$

where

μ_s = gravitational constant of the sun

\vec{r}_{s-b} = position vector from the sun to the spacecraft

\vec{r}_{e-s} = position vector from the Earth to the sun

To avoid numerical problems, use is made of Richard Battin's $f(q)$ function given by

$$f(q_k) = q_k \left[\frac{3 + 3q_k + q_k^2}{1 + (\sqrt{1 + q_k})^3} \right]$$

where

$$q_k = \frac{\mathbf{r}^T (\mathbf{r} - 2\mathbf{s}_k)}{\mathbf{s}_k^T \mathbf{s}_k}$$

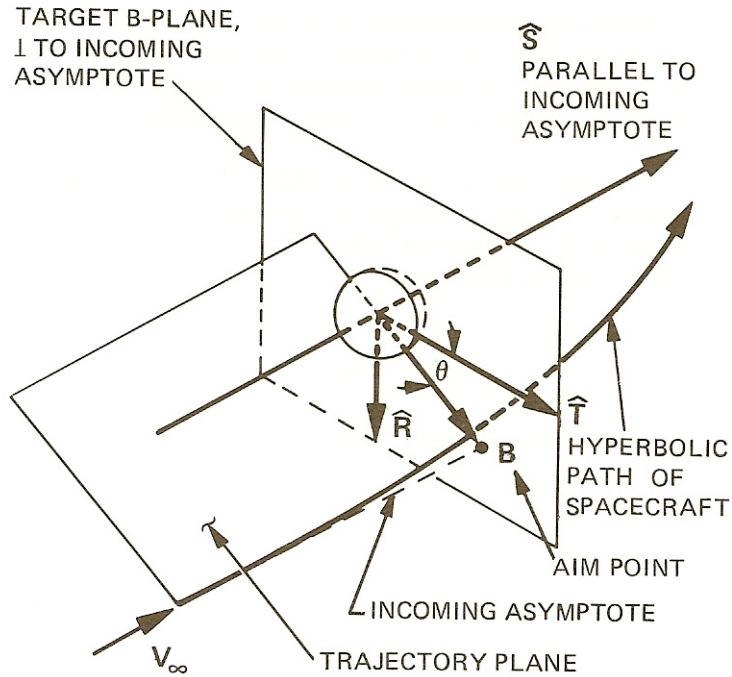
The third-body acceleration can now be expressed as

$$\ddot{\mathbf{r}} = -\sum_{k=1}^n \frac{\mu_k}{d_k^3} [\mathbf{r} + f(q_k) \mathbf{s}_k]$$

In this computer program the heliocentric coordinates of the sun and moon are computed using the JPL Development Ephemeris DE421. These coordinates are provided in the Earth mean equator and equinox of J2000 coordinate system (EME2000).

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 $\hat{\mathbf{S}}$ is given by

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

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 mean lunar 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 lunar equator and IAU node of epoch system are as follows:

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

The following figure illustrates the orientation of this coordinate system relative to the Earth's mean equator and equinox of J2000 (EME2000). The fundamental plane of this inertial system is the lunar mean equator and the fundamental x-axis is the IAU node of epoch. The y-axis is advanced 90 degrees along the lunar equator from the x-axis, and the z-axis is perpendicular to the mean equator of the moon. The term mean indicates that precession has been accounted for, but not the effect of nutation. The x-axis or Q-vector is formed from the cross product of the Earth's mean pole of J2000 and the Moon's north pole relative to EME2000.

This illustration was extracted from JPL D-32296, "Lunar Constants and Models Document" dated September 23, 2005.

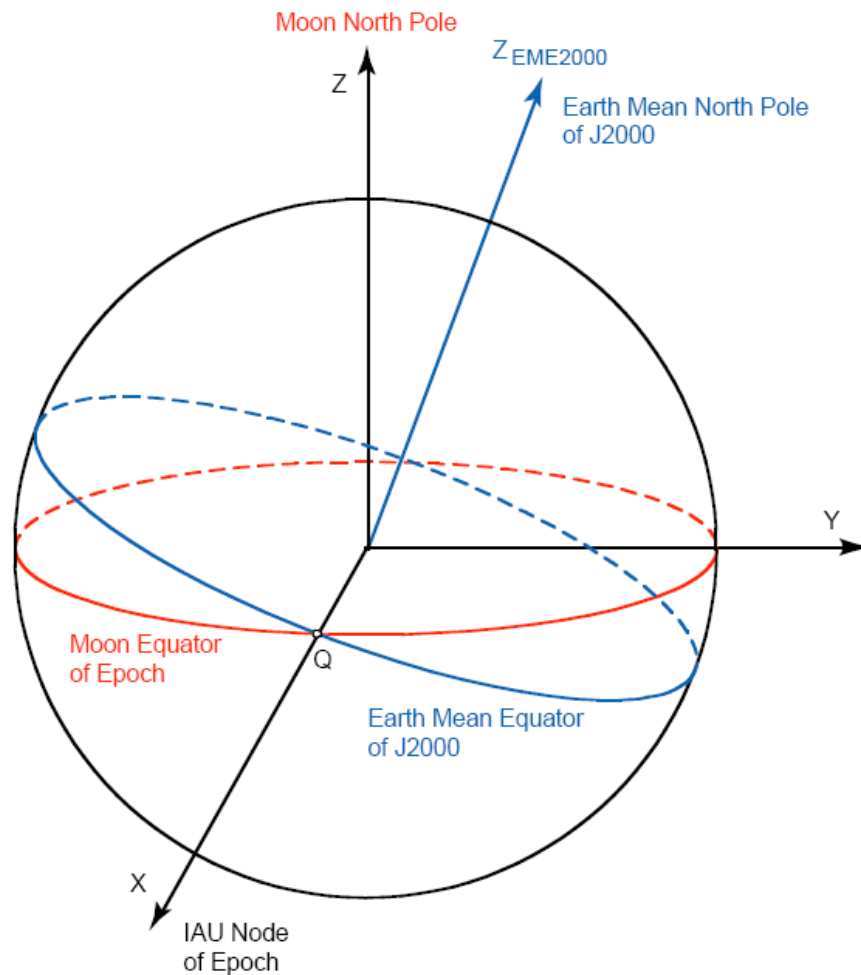


Figure 1. Moon mean equator and IAU node of epoch coordinate system

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 ellipse.

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

$$i \geq |\delta_\infty|$$

where 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 `tli_soc`s computer program. This software was created using version 6.4.3 of SOCS and Compaq Visual Fortran.

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

```
fl32 /arch:host tli_soc.f *.for c:\soc\socs643.lib advapi32.lib
```

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

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:

```
tli_soc tli_soc1.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 tli_soc           *
*                                     *
*           TLI delta-v               *
*           optimization with SOCS    *
*                                     *
*           February 2, 2006          *
*****
```

```
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 tli_socs          *'
print *, ' *          *          *'
print *, ' *          TLI delta-v          *'
print *, ' *          optimization with SOCS      *'
print *, ' *          *          *'
print *, ' *          February 2, 2006          *'
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 `tli_soc`s 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-x = x-component of the TLI impulsive velocity vector in meters/second
deltav-y = y-component of the TLI impulsive velocity vector in meters/second
deltav-z = z-component of the TLI impulsive velocity vector in meters/second
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 relative to the user's TLI initial guess 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 tli_socsl computer program.

- tli_socsl.f** - SOCS main executive program
- atan3.for** - four quadrant inverse tangent function
- 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
- findsoi.for** - subroutine that computes time of entry to the lunar SOI
- findca.for** - subroutine that computes time of closest approach to the moon
- 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
- glambert.for** - two-body Lambert subroutine
- linput.for** - read and echo a line of text from an input file subroutine
- lunfunc.for** - subroutine that computes two-body delta-v objective function
- 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
root.for - real root of a nonlinear equation subroutine
rv2bp.for - convert position and velocity to b-plane coordinates subroutine
sel_eqm.for - selenocentric equations of motion subroutine
soiobj.for - sphere-of-influence objective function subroutine
sv2000.for - lunar and solar ephemeris 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 tli_socs software. This subroutine is the point function routine required by the SOCS software.

```
      subroutine odepf(iphase, iphend, time, ydyn, nydyn, parm,
&                   nparm, ptf, nptf, iferr)
c
c   evaluate point functions
c
c   *****
c
c   implicit double precision (a-h, o-z)
c
c   include 'socscom1.inc'
c
c   parameter (zero = 0.0d0, one = 1.0d0)
c
c   dimension ydyn(nydyn), parm(nparm), ptf(nptf)
c
c   dimension rmoon(3), vmoon(3), rtmp(3), vtmp(3)
c
c   dimension reci(3), veci(3), hv(3), tmatrix(3, 3)
c
c   dimension bplane(12), tv(3), rv(3)
c
c   iferr = 0
c
c   do i = 1, 3
c     reci(i) = ydyn(i)
c
c     veci(i) = ydyn(i + 3)
c   end do
c
c   if (iphase .eq. 1 .and. iphend .eq. -1) then
c     *****
c     "position & velocity match" at beginning of phase
c     *****
c
c     position match
c
c     ptf(1) = reci(1) - rpo(1)
c
c     ptf(2) = reci(2) - rpo(2)
c
c     ptf(3) = reci(3) - rpo(3)
c
c     velocity match
c
c     ptf(4) = veci(1) - (vpo(1) + parm(1))
c
c     ptf(5) = veci(2) - (vpo(2) + parm(2))
c
c     ptf(6) = veci(3) - (vpo(3) + parm(3))
c
c     *****
```

```

c      launch delta-v contribution to objective function
c      *****

      dvm1 = sqrt(parm(1)**2 + parm(2)**2 + parm(3)**2)

      ptf(7) = dvm1

end if

if (iphase .eq. 1 .and. iphend .eq. +1) then
c      *****
c      radius at SOI, periapsis, inclination or b-plane constraints
c      *****

      xjdate = xjdate0 + time / 86400.0d0

      call sv2000(xjdate, 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 (xjdate, tmatrix)

      call matxvtr(tmatrix, rtmp, rm2sc)

      call matxvtr(tmatrix, vtmp, vm2sc)

c      selenocentric distance point function (kilometers)

      ptf(1) = vecmag(rm2sc)

c      find closest approach

      call findca(xjdate, 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      cosine of orbital inclination point function
c      -----

      call vcross(rca, vca, hv)

      hmag = vecmag(hv)

      ptf(3) = hv(3) / hmag

else
c      user-defined b-plane targets
c      -----

      ptf(2) = bplane(7) - bpuser(2)

      ptf(3) = bplane(8) - bpuser(1)

      end if

end if

return
end

```