Program lunar_tcm

Translunar TCM Delta-V Optimization with SOCS

This document is the user's manual for a Fortran computer program called `lunar_tcm` 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, trans-lunar trajectory correction maneuver (TCM) problem. The software attempts to minimize the scalar magnitude of the impulsive TCM 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 in the geocentric phase of flight
- Earth non-spherical and lunar J_2 gravity models
- JPL DE421 lunar and solar ephemeris
- B-plane targeting 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 `lunar_tcm` 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 `lunar_tcm` 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.

```
****************************************************
** trans-lunar TCM trajectory optimization with SOCS
** n-body geocentric motion
** data file ==> tcm_socs1.in
** November 20, 2007
****************************************************

TCM epoch
2018 JUL 19 19:32:59.01641 UTC

************************************************
geocentric EME2000 orbital elements prior to TCM
************************************************

semimajor axis (kilometers)
0.220570073828D+06

orbital eccentricity (non-dimensional)
0.969738070789D+00

orbital inclination (degrees)
0.280960107795D+02

argument of perigee (degrees)
0.487471688823D+02

right ascension of the ascending node (degrees)
0.366067656510D+02

true anomaly (degrees)
0.314559641701D+01

**********************************************************
initial guess and bounds for geocentric TCM delta-v vector
**********************************************************

x-component of TCM velocity vector (meters/second)
0.0

y-component of TCM velocity vector (meters/second)
0.0

z-component of TCM velocity vector (meters/second)
```

```
0.0

lower bound for TCM delta-v components (meters/second)
-100.0

upper bound for TCM delta-v components (meters/second)
+100.0
```

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)
140.0
```

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

```
************************
trajectory perturbations
************************
```

```
name of Earth gravity model data file
egm96.dat

order of Earth gravity model (zonals)
8

degree of lunar gravity model (tesserals)
8

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

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

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
tcm_socs1.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

The following is the program output created by the `lunar_tcm` simulation for this example.

```
--------------------------------------
tcm_socs - translunar TCM optimization
--------------------------------------

input data file ==> tcm_socs1.in


tzero                0.000000000000000E+000 hours

tfinal                   90.241762167212727 hours


--------------------------------
TCM delta-v vector and magnitude
(geocentric EME2000)
--------------------------------

delta-vx            -4.425943188518626E-001 meters/second
delta-vy                 -2.748958543497322 meters/second
delta-vz                 -1.693102241080948 meters/second

deltav                    3.258720301544793 meters/second


pitch angle             -79.766185875275966 degrees

yaw angle               271.949406619600040 degrees


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

UTC epoch             2018 JUL 19 19:32:59.01641

TDB Julian date       2458319.315326388925314

     sma (km)          eccentricity      inclination (deg)      argper (deg)
 0.220570073825D+06    0.969738070789D+00    0.280960107795D+02    0.487471688823D+02

    raan (deg)        true anomaly (deg)      arglat (deg)        period (min)
 0.366067656510D+02    0.314559641699D+01    0.518927652993D+02    0.171822333599D+05

     rx (km)               ry (km)              rz (km)              rmag (km)
 0.544259155991D+03    0.618033703692D+04    0.247534969797D+04    0.667983083962D+04

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 -.103399314811D+02    -.778103920087D-01    0.325838868403D+01    0.108414636607D+02
```

```
-----------------------------
time and conditions after TCM
(geocentric EME2000)
-----------------------------

UTC epoch             2018 JUL 19 19:32:59.01641

TDB Julian date       2458319.315326388925314


     sma (km)             eccentricity       inclination (deg)      argper (deg)
 0.220394166294D+06    0.969713428350D+00    0.280941230019D+02    0.487860563612D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.366016542529D+02    0.311121804690D+01    0.518972744081D+02    0.171616828734D+05

     rx (km)                ry (km)               rz (km)              rmag (km)
 0.544259155991D+03    0.618033703692D+04    0.247534969797D+04    0.667983083962D+04

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 -.103403740754D+02    -.805593505522D-01    0.325669558179D+01    0.108413971397D+02


orbital energy         -1.808579819522805 (km/sec)**2


--------------------------------
time and conditions at lunar SOI
(geocentric EME2000)
--------------------------------

UTC epoch             2018 JUL 23 13:47:29.36033

TDB Julian date       2458323.075399812776595


     sma (km)             eccentricity       inclination (deg)      argper (deg)
 0.219352903352D+06    0.998778141093D+00    0.772872085685D+02    0.206215776728D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.719690169270D+02    0.179060751620D+03    0.199682329292D+03    0.170402048325D+05

     rx (km)                ry (km)               rz (km)              rmag (km)
 -.872932816397D+05    -.362763875780D+06    -.129792739340D+06    0.395049239685D+06

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 -.875384030822D-01    -.400620993267D+00    -.180702160909D+00    0.448122107486D+00


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

UTC epoch             2018 JUL 23 13:47:29.36033

TDB Julian date       2458323.075399812776595


     sma (km)             eccentricity       inclination (deg)      argper (deg)
 -.745477124319D+04    0.124791853410D+01    0.140133391049D+03    0.328307121125D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.194940703369D+03    0.228074246339D+03    0.196381367464D+03    0.000000000000D+00

     rx (km)                ry (km)               rz (km)              rmag (km)
```

```
 0.245695259127D+05    0.955070191692D+03    -.451953945952D+04    0.249999999914D+05

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 -.100616794842D+01    -.180101686510D+00    0.713179924617D-01    0.102464472573D+01


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

b-magnitude            5565.178555303199573 kilometers
b dot r                3559.117699922789143
b dot t               -4278.304985681002108
theta                   140.242963060714601 degrees
v-infinity              810.970279841494857 meters/second
r-periapsis            1848.175958675761876 kilometers
decl-asy                  3.234211465170965 degrees
rasc-asy                191.060963086808869 degrees


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

UTC epoch             2018 JUL 23 19:31:47.91648

TDB Julian date       2458323.314503471832722


     sma (km)            eccentricity       inclination (deg)       argper (deg)
 -.741496532353D+04    0.124787708611D+01    0.139999999973D+03    0.328344720616D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.194884345687D+03    0.360000000000D+03    0.328344720616D+03    0.000000000000D+00

     rx (km)               ry (km)               rz (km)              rmag (km)
 -.132224159397D+04    -.111601231832D+04    -.620030374509D+03    0.183799999804D+04

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 -.165212676650D+01    0.121305431701D+01    0.133981689085D+01    0.244870025311D+01


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

b-magnitude            5534.948644114345370 kilometers
b dot r                3549.561151801780397
b dot t               -4246.913246418255767
theta                   140.111193404530383 degrees
v-infinity              813.144140947894357 meters/second
r-periapsis            1837.999998037190608 kilometers
decl-asy                  3.265706329641945 degrees
rasc-asy                190.985199478887409 degrees

flight path angle   -1.141901620713837E-012 degrees


TCM-to-SOI time          90.241762172430754 hours

TCM-to-CA time           95.980249989777803 hours
                          3.999177082907408 days
```

```
------------------------------------
trajectory verification of SOCS
solution using numerical integration
------------------------------------

****************
TCM to SOI phase
****************


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

UTC epoch             2018 JUL 19 19:32:59.01641

TDB Julian date       2458319.315326388925314


     sma (km)            eccentricity       inclination (deg)      argper (deg)
 0.220394166294D+06    0.969713428350D+00    0.280941230019D+02    0.487860563612D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.366016542529D+02    0.311121804690D+01    0.518972744081D+02    0.171616828734D+05

     rx (km)               ry (km)               rz (km)             rmag (km)
 0.544259155991D+03    0.618033703692D+04    0.247534969797D+04    0.667983083962D+04

     vx (kps)             vy (kps)              vz (kps)             vmag (kps)
 -.103403740754D+02    -.805593505522D-01    0.325669558179D+01    0.108413971397D+02


orbital energy         -1.808579819522805 (km/sec)**2

--------------------------------
time and conditions at lunar SOI
(geocentric EME2000)
--------------------------------

UTC epoch             2018 JUL 23 13:47:29.36033

TDB Julian date       2458323.075399812776595


     sma (km)            eccentricity       inclination (deg)      argper (deg)
 0.219352906874D+06    0.998778141182D+00    0.772872355310D+02    0.206215752226D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.719690260138D+02    0.179060751611D+03    0.199682326833D+03    0.170402052428D+05

     rx (km)               ry (km)               rz (km)             rmag (km)
 -.872932879138D+05    -.362763877636D+06    -.129792738468D+06    0.395049242489D+06

     vx (kps)             vy (kps)              vz (kps)             vmag (kps)
 -.875384306199D-01    -.400621002895D+00    -.180702167314D+00    0.448122124055D+00


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

UTC epoch             2018 JUL 23 13:47:29.36033

TDB Julian date       2458323.075399812776595
```

```
      sma (km)           eccentricity       inclination (deg)      argper (deg)
 -.745477172097D+04    0.124791851169D+01    0.140133408583D+03    0.328307123956D+03

     raan (deg)        true anomaly (deg)       arglat (deg)         period (min)
 0.194940706574D+03    0.228074247680D+03    0.196381371636D+03    0.000000000000D+00

      rx (km)               ry (km)               rz (km)              rmag (km)
 0.245695197476D+05    0.955068556444D+03    -.451953776092D+04    0.249999935630D+05

      vx (kps)             vy (kps)              vz (kps)              vmag (kps)
 -.100616797540D+01    -.180101699260D+00    0.713179912058D-01    0.102464475437D+01


***************************
SOI to close approach phase
***************************


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

UTC epoch             2018 JUL 23 19:31:47.91648

TDB Julian date       2458323.314503471832722

      sma (km)           eccentricity       inclination (deg)      argper (deg)
 -.741496576394D+04    0.124787706603D+01    0.140000017679D+03    0.328344723498D+03

     raan (deg)        true anomaly (deg)       arglat (deg)         period (min)
 0.194884348987D+03    0.462070924888D-03    0.328345185569D+03    0.000000000000D+00

      rx (km)               ry (km)               rz (km)              rmag (km)
 -.132225151516D+04    -.111600516665D+04    -.620021971763D+03    0.183799995833D+04

      vx (kps)             vy (kps)              vz (kps)              vmag (kps)
 -.165212054459D+01    0.121306003873D+01    0.133981941102D+01    0.244870026863D+01


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

b-magnitude            5534.948723923113903 kilometers
b dot r                3549.559886101484608
b dot t               -4246.914408301316143
theta                   140.111211170786646 degrees
v-infinity              813.144116799827998 meters/second
r-periapsis            1837.999958295321676 kilometers
decl-asy                  3.265706182104615 degrees
rasc-asy                190.985200504892362 degrees

flight path angle    2.565121192843666E-004 degrees
```

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

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

To create and initial guess for the `lunar_tcm` software, the program 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. 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`.

(1) Performance index – minimize TCM 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 TCM impulsive delta-v vector are the optimization parameters used by SOCS to solve this trajectory problem.

(2) Point functions – position and velocity vector "matching" at TCM

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

$$\mathbf{r}_{TCM}\left(t_{TCM}\right) - \mathbf{r}_{po}\left(t_{TCM}\right) = 0$$

$$\mathbf{v}_{to}\left(t_{TCM}\right) - \left\{\mathbf{v}_{po}\left(t_{TCM}\right) + \Delta\mathbf{v}_{TCM}\right\} = 0$$

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 TCM 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

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

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

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 $\mathbf{B \bullet T}$ and $\mathbf{B \bullet R}$ 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 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

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

where **r** and **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

$$\dot{y}_1 = v_x \qquad \dot{y}_2 = v_y \qquad \dot{y}_2 = v_z$$

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

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

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

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 `lunar_tcm` 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 propagating the spacecraft's trajectory, computing the B-plane coordinates and calculating the geocentric-to-selenocentric coordinate transformation.

Geocentric equations of motion

The `lunar_tcm` 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

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

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

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

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

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

where

$$
\begin{aligned}
R &= \text{ radius of the Earth} \\
r &= \text{ geocentric distance of the spacecraft} \\
S_n^m, C_n^m &= \text{ harmonic coefficients} \\
\phi &= \text{ geocentric latitude of the spacecraft} = \sin^{-1}\left(z/r\right) \\
\lambda &= \text{ longitude of the spacecraft} = \alpha - \alpha_g \\
\alpha &= \text{ right ascension of the spacecraft} = \tan^{-1}\left(r_y/r_x\right) \\
\alpha_g &= \text{ right ascension of Greenwich}
\end{aligned}
$$

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:

$$
\begin{aligned}
&P_n^0\left(\sin\phi\right) = \frac{1}{n}\left[\left(2n-1\right)\sin\phi\, P_{n-1}^0\left(\sin\phi\right) - \left(n-1\right)P_{n-2}^0\left(\sin\phi\right)\right] \\
&P_n^n\left(\sin\phi\right) = \left(2n-1\right)\cos\phi\, P_{n-1}^{n-1}\left(\sin\phi\right), \qquad m \neq 0,\ m < n \\
&P_n^m\left(\sin\phi\right) = P_{n-2}^m\left(\sin\phi\right) + \left(2n-1\right)\cos\phi\, P_{n-1}^{m-1}\left(\sin\phi\right), \qquad m \neq 0,\ m = n
\end{aligned}
$$

where the first few associated Legendre functions are given by

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

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

The trigonometric arguments are determined from expansions given by

$$
\begin{aligned}
&\sin m\lambda = 2\cos\lambda\sin\left(m-1\right)\lambda - \sin\left(m-2\right)\lambda \\
&\cos m\lambda = 2\cos\lambda\cos\left(m-1\right)\lambda - \cos\left(m-2\right)\lambda \\
&m\tan\phi = \left(m-1\right)\tan\phi + \tan\phi
\end{aligned}
$$

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

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

where $\left[PN\right]$ 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

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

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

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

Point mass acceleration of the sun and moon

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

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

where

$\mu_m =$ 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

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

where

$\mu_s =$ 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 $f\left(q\right)$ function given by

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

where

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

The point-mass acceleration due to n gravitational bodies can now be expressed as

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

Spacecraft selenocentric equations of motion

The first-order, nonlinear differential equations of geocentric motion implemented in the `tli_socs` software are as follows:

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

where

$t =$ barycentric dynamical time

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

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

$\mathbf{a}_e =$ acceleration due to the Earth

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

$$\dot{y}_1 = v_x \qquad \dot{y}_2 = v_y \qquad \dot{y}_2 = v_z$$

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

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

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

where

$$r_x, r_y, r_z = \text{ inertial position vector components of the spacecraft}$$
$$v_x, v_y, v_z = \text{ inertial velocity vector components of the spacecraft}$$
$$a_{m_x}, a_{m_y}, a_{m_z} = \text{ acceleration components due to the Moon}$$
$$a_{s_x}, a_{s_y}, a_{s_z} = \text{ acceleration components due to the Sun}$$
$$r = \sqrt{r_x^2 + r_y^2 + r_z^2} = \text{ geocentric distance of the spacecraft}$$
$$\mu = \text{ gravitational constant of the Earth}$$
$$r_{eq} = \text{ equatorial radius of the Earth}$$
$$J_2 = \text{ oblateness gravity coefficient for the Earth}$$

Point mass acceleration of the Earth and sun

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

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

where

$$\mu_m = \text{ gravitational constant of the moon}$$
$$\vec{r}_{m-b} = \text{ position vector from the moon to the spacecraft}$$
$$\vec{r}_{e-m} = \text{ position vector from the Earth to the moon}$$

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

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

where

$$\mu_s = \text{ gravitational constant of the sun}$$
$$\vec{r}_{s-b} = \text{ position vector from the sun to the spacecraft}$$
$$\vec{r}_{e-s} = \text{ position vector from the Earth to the sun}$$

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

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

where

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

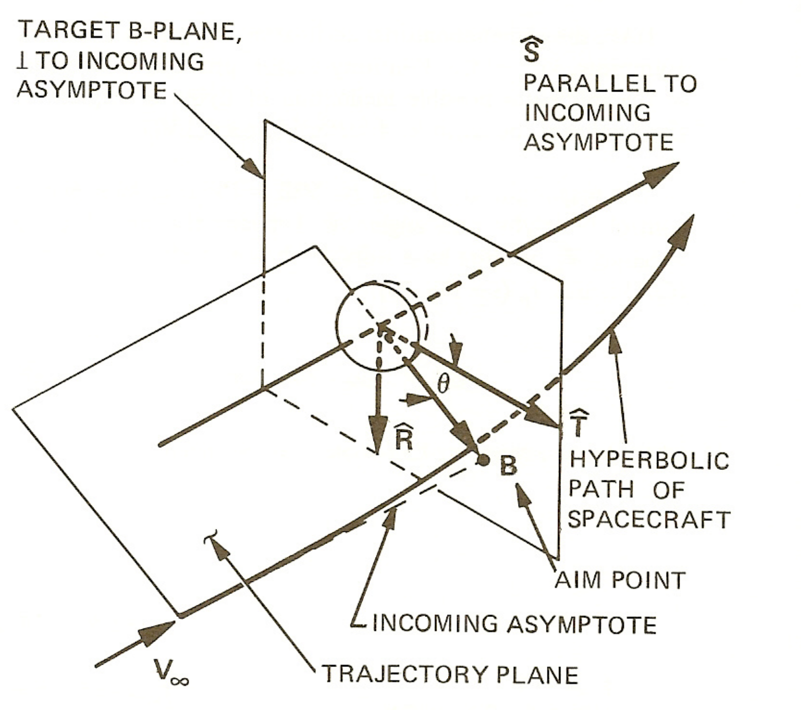
The third-body acceleration can now be expressed as

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

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 **r** and velocity vector **v** at closest approach.

angular momentum vector

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

radius rate

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

semiparameter

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

semimajor axis

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

orbital eccentricity

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

true anomaly

$$\cos\theta = \frac{p - r}{e\,r}$$

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

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

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

R vector

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

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 $T = \left(JD - 2451545.0\right)/36525$ and JD is the TDB Julian Date.

The trigonometric arguments, in degrees, for these equations are

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

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 $\hat{\mathbf{z}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. 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:

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

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.

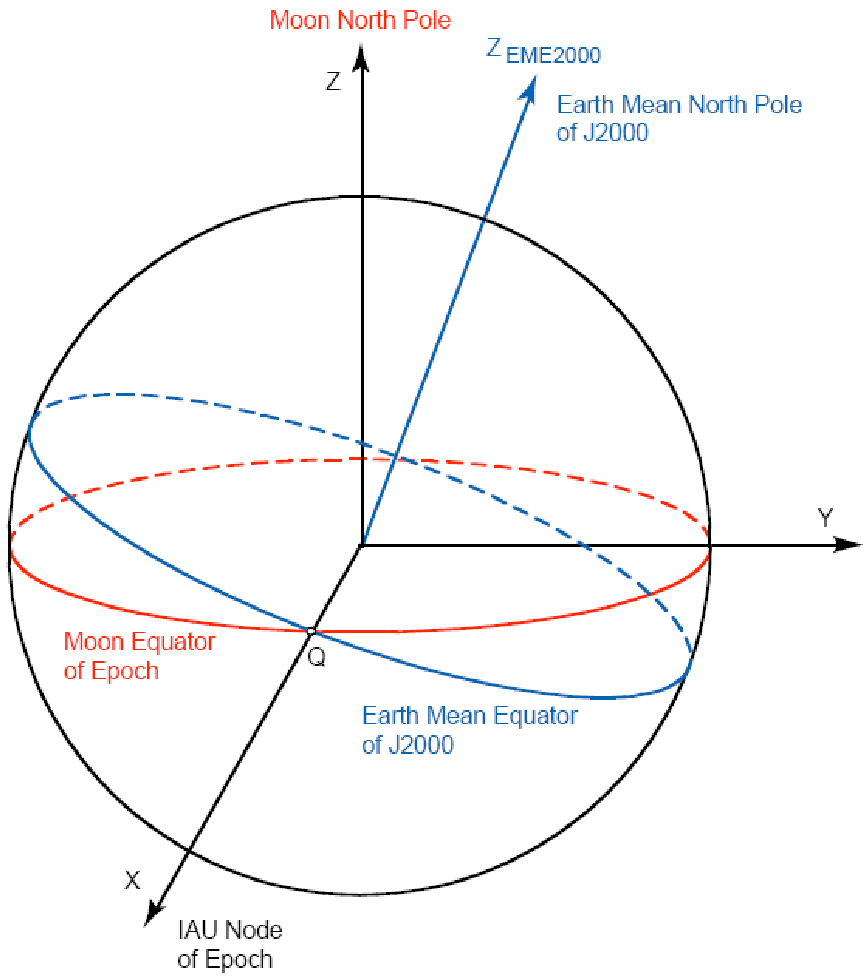


Figure 1. Moon mean equator and IAU node of epoch coordinate system

Maneuver pitch and yaw angles

The pitch and yaw angles can be determined from the components of the unit thrust vector in the radial-tangential-normal (RTN) system according to

$$\theta = \sin^{-1}\left(u_r\right)$$

$$\psi = \tan^{-1}\left(u_n, u_t\right)$$

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

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 RTN system $\hat{\mathbf{u}}_{RTN}$ is given by

$$\hat{\mathbf{u}}_{T_{ECI}} = \left[\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n\right]\hat{\mathbf{u}}_{T_{RTN}}$$

where

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

This relationship can also be expressed as

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

The transformation of the unit thrust vector in the ECI system to the RTN coordinate system is given by

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

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

$$i \ge |\delta_\infty|$$

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 `lunar_tcm` computer program. This software was created using version 6.4.7 of SOCS and Compaq Visual Fortran.

A DOS/Windows version of `lunar_tcm` using Compaq Visual Fortran version 6.6C and the JPL Spice libraries can be created with the following command:

```
fl32 /arch:host lunar_tcm.f *.for socs647.lib spicelib.lib support.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:

```
      lunar_tcm tcm_socs1.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 lunar_tcm         *'
      *  '                                *'
      *      translunar TCM delta-v       *'
      *      optimization with SOCS       *'
      *                                   *'
      *         November 20, 2007         *'
      *************************************'

      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 lunar_tcm         *'
    print *, '        *                                   *'
    print *, '        *      translunar TCM delta-v       *'
    print *, '        *      optimization with SOCS       *'
    print *, '        *                                   *'
    print *, '        *         November 20, 2007         *'
    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 `lunar_tcm` 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 TCM impulsive velocity vector in meters/second

deltav-y = y-component of the TCM impulsive velocity vector in meters/second

deltav-z = z-component of the TCM impulsive velocity vector in meters/second

deltav = scalar magnitude of the TCM 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 TCM

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 = ecc * cos(argper + raan)

gmee = modified equinoctial orbital element = ecc * sin(argper + raan)

hmee = modified equinoctial orbital element = tan(i/2) * cos(raan)

xkmee = modified equinoctial orbital element = tan(i/2) * sin(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 `lunar_tcm` computer program.

tcm_socs.f - SOCS main executive program

atan3.for - four quadrant inverse tangent function

display1.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

fpaobj.for – selenocentric flight path angle objective function 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

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 (RKF78) 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 `lunar_tcm` 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     evaluate point functions

c     ************************************

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

      include 'socscom1.inc'

      parameter (zero = 0.0d0, one = 1.0d0)

      dimension ydyn(nydyn), parm(nparm), ptf(nptf)

      dimension rmoon(3), vmoon(3), rtmp(3), vtmp(3)

      dimension reci(3), veci(3), hv(3), tmatrix(3, 3)

      dimension bplane(12), tv(3), rv(3)

      iferr = 0

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

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

      if (iphase .eq. 1 .and. iphend .eq. -1) then
c        ************************************************
c        "position & velocity match" at beginning of phase
c        ************************************************

c        position match

         ptf(1) = reci(1) - rtcm(1)

         ptf(2) = reci(2) - rtcm(2)

         ptf(3) = reci(3) - rtcm(3)

c        velocity match

         ptf(4) = veci(1) - (vtcm(1) + parm(1))

         ptf(5) = veci(2) - (vtcm(2) + parm(2))

         ptf(6) = veci(3) - (vtcm(3) + parm(3))
```

```
c        *************************
c        delta-v 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 = xjdtcm + 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
```