

Program tlto_ocs

Finite-burn Trans-lunar Trajectory Optimization

This document is the user's manual for a Fortran computer program called `tlto_ocs` that uses optimal control software distributed by Applied Mathematical Analysis to solve the classic finite-burn, trans-lunar injection (TLI) trajectory optimization problem. The software attempts to maximize the spacecraft mass at the end of the TLI propulsive maneuver while targeting to a user-defined periapsis radius and orbital inclination relative to the moon. Since the TLI is a continuous thrust maneuver, maximizing the spacecraft mass is equivalent to minimizing the propellant required.

The important features of this scientific simulation are as follows:

- single, continuous thrust propulsive maneuver followed by a coast to lunar close approach
- combination of modified equinoctial and cartesian equations of motion
- valid for circular and elliptical park orbits
- Earth J_2 gravity model and sun and moon point-mass gravity
- JPL DE421 lunar and solar ephemeris
- B-plane coordinates at lunar close approach

The `AMA_OC` software suite 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 the `AMA_OC` software can be found in the book, "Practical Methods for Optimal Control and Estimation Using Nonlinear Programming" by John. T. Betts, SIAM, 2010.

The `tlto_ocs` software consists of Fortran routines that perform the following tasks;

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

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

Program execution

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:

```
tlto_ocs tlto1.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 tlto_ocs           *
*                                       *
*   finite-burn lunar trajectory      *
*           optimization               *
*                                       *
*           March 17, 2011             *
*****

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

The user should respond to this prompt with the name of a compatible input data file including the filename extension.

Input file format and contents

The `tlto_ocs` 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 Barycentric Dynamical 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.

```
*****
** finite-burn translunar trajectory optimization
** two phase, 3-body geocentric motion
** input file for tlto_ocs
** tlto1.in - March 8, 2011
*****
```

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

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

For a fixed maneuver time, the lower and upper bounds should be set to zero.

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

```
initial calendar date (month, day, year)
10, 13, 2008

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

lower bound for initial time guess (hours)
-24.0

upper bound for initial time guess (hours)
+24.0
```

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

```
initial spacecraft mass (kilograms)
1000.0

thrust magnitude (newtons)
5000.0

specific impulse (seconds)
450.0
```

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

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

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

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

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

```
initial guess for thrust duration (seconds)
3000.0

lower bound for thrust duration (seconds)
1.0

upper bound for thrust duration (seconds)
10000.0
```

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

```

*****
initial park orbit
*****

semimajor axis (kilometers)
6563.3363

orbital eccentricity (non-dimensional)
0.0d0

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
0.0d0

right ascension of the ascending node (degrees)
282.787

true anomaly (degrees)
305.233

```

This next integer input allows the user to define the type of initial park orbit constraints to use during the simulation. Option 1 will constrain all elements of the park orbit except the true longitude to the values input by the user. Option 2 will constrain the semimajor axis, eccentricity and orbital inclination. The RAAN and true longitude will be bounded.

```

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

```

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

```

transfer time initial guess (hours)
96.0

transfer time lower bound (hours)
84.0d0

transfer time upper bound (hours)
108.0d0

```

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
(lunar mean equator and IAU node of epoch)
-----

periapsis radius (kilometers)
1838.0

orbital inclination (degrees)
90.0

```

The next three inputs define the types of perturbations to include during the numerical solution of the spacecraft's equations of motion. The solar and lunar perturbations are modeled as point-mass bodies.

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

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

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

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

The next program input is the user-defined radius of the lunar sphere-of-influence (SOI) used by the software during the trajectory optimization. Typical values for this parameter are between 64,000 and 20,000 kilometers.

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

This next input defines the type of initial guess to use. Please see the technical discussion section for information about how the first option is modeled. Option 2 requires either a binary restart file created from a previous run using either initial guess option 1 or an updated binary restart file. This feature is described in the next two sections.

```
*****
* initial guess/restart option *
*****
1 = numerical integration
2 = binary data file
-----
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of initial guess/restart input data file
tlto1.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of yes to this item will update the binary file used to initialize the simulation.

```
*****
* binary restart file option *
*****

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

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by

AMA_OC. Options 2 and 3 allow the user to specify either the number of nodes or time step size used to create the data file.

```
*****
* type of comma-delimited solution data file *
*****
1 = OC-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
```

The name of the solution data file is defined in this next line. Please consult Appendix A for a description of the information written to this file.

```
name of solution output file
tlto1.csv
```

The next series of program inputs are algorithm control options and parameters for the AMA_OC 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
-----
1
```

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

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

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

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

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

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

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

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

```
maximum number of mesh refinement iterations
20
```

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

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

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

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

The level of output from the AMA_OC NLP algorithm is controlled with the following integer input. Additional information about these algorithm items can be found in the AMA_OC user's manual.

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

The level of output from the AMA_OC 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 AMA_OC 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 AMA_OC user's manual. To ignore this special output control, input the simple character string no.

```
*****
user-defined output
-----
```

```
input no to ignore
-----
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of AMA_OC configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the tlto_ocs software. Please consult Appendix E.

```
*****
* optimal control configuration options
*****

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
tlto1_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
tlto1_config.txt
```

Optimal Control Solution

The following is the program output created by the tlto_ocs scientific simulation for this example. This display summarizes the characteristics of the optimized TLI maneuver and the orbital transfer conditions before and after the propulsive maneuver. It also displays the conditions at the lunar SOI, the closest approach to the moon, and the corresponding B-plane characteristics. The delta-v magnitude is determined from a cubic spline integration of the thrust acceleration evaluated at the grid points determined by the AMA_OC software.

```
-----
program tlto_ocs
-----

input data file ==> tlto1.in

numerical integration initial guess

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

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

calendar date          October 12, 2008

TDB time                17:30:18.860

TDB Julian date        2454752.22938496

      sma (km)          eccentricity      inclination (deg)      argper (deg)
0.656333630000D+04    0.248233137449D-15    0.285000000002D+02    0.000000000000D+00

      raan (deg)        true anomaly (deg)      arglat (deg)          period (hrs)
0.288041391117D+03    0.285698028908D+03    0.285698028908D+03    0.146992598213D+01
```

rx (km)	ry (km)	rz (km)	rmag (km)
-.472983715871D+04	-.340824686342D+04	-.301494231403D+04	0.656333630000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.408542510472D+01	-.655960539581D+01	0.100610848572D+01	0.779303378153D+01

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

calendar date October 12, 2008
TDB time 17:37:49.024
TDB Julian date 2454752.23459518

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.188592901679D+06	0.964964312884D+00	0.285135532542D+02	0.302819212999D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.287991628544D+03	0.184565053021D+02	0.321275718301D+03	0.226410463496D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.191050323432D+04	-.618085897383D+04	-.202431074552D+04	0.677870741788D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.849302767938D+01	-.561049532896D+01	0.344687472134D+01	0.107466330701D+02

final mass 489.956599726842 kilograms
propellant mass 510.043400273158 kilograms
thrust duration 450.163540015994 seconds
delta-v 3148.40213255454 meters/second

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

calendar date October 12, 2008
TDB time 17:37:49.024
TDB Julian date 2454752.23459518

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.188592901693D+06	0.964964312887D+00	0.285135532542D+02	0.302819212999D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.287991628544D+03	0.184565053022D+02	0.321275718301D+03	0.226410463522D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.191050323432D+04	-.618085897384D+04	-.202431074553D+04	0.677870741788D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.849302767939D+01	-.561049532896D+01	0.344687472133D+01	0.107466330701D+02

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

```

calendar date      October 16, 2008
TDB time          20:19:59.385
TDB Julian date   2454756.34721510

   sma (km)      eccentricity      inclination (deg)      argper (deg)
0.182741011508D+06  0.991360644759D+00  0.787957481842D+02  0.336573217908D+03

   raan (deg)    true anomaly (deg)      arglat (deg)      period (hrs)
0.236510628329D+03  0.179365248430D+03  0.155938466338D+03  0.215954628509D+03

   rx (km)      ry (km)      rz (km)      rmag (km)
0.205941002561D+06  0.259386093079D+06  0.144524220245D+06  0.361358675081D+06

   vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.780197771732D-01  0.131667352020D+00  -.382877947133D-01  0.157763533309D+00

```

```

-----
time and conditions at lunar sphere-of-influence
(selenocentric lunar mean equator & IAU node of epoch)
-----

```

```

calendar date      October 16, 2008
TDB time          20:19:59.385
TDB Julian date   2454756.34721510

   sma (km)      eccentricity      inclination (deg)      argper (deg)
-.620162043993D+04  0.129578694816D+01  0.902140904168D+02  0.315764985231D+03

   raan (deg)    true anomaly (deg)      arglat (deg)
0.325952802357D+03  0.230078896017D+03  0.185843881248D+03

   rx (km)      ry (km)      rz (km)      rmag (km)
-.206014388084D+05  0.139320301916D+05  -.254543770383D+04  0.249999999868D+05

   vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.899321600623D+00  -.607355493081D+00  -.716351247635D-01  0.108756224071D+01

```

```

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

```

```

calendar date      October 17, 2008
TDB time          01:48:33.193
TDB Julian date   2454756.57538418

   sma (km)      eccentricity      inclination (deg)      argper (deg)
-.616494773235D+04  0.129813715848D+01  0.899999999984D+02  0.315640433302D+03

   raan (deg)    true anomaly (deg)      arglat (deg)
0.325911410381D+03  0.671733148725D-11  0.315640433302D+03

   rx (km)      ry (km)      rz (km)      rmag (km)
0.108830738216D+04  -.736523448998D+03  -.128505418087D+04  0.183799999912D+04

   vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.143361909576D+01  -.970216777116D+00  0.177020213307D+01  0.247592404937D+01

```

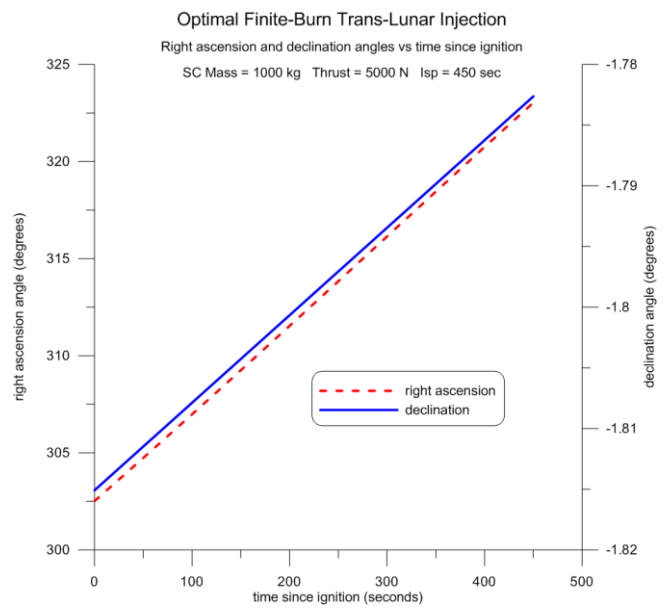
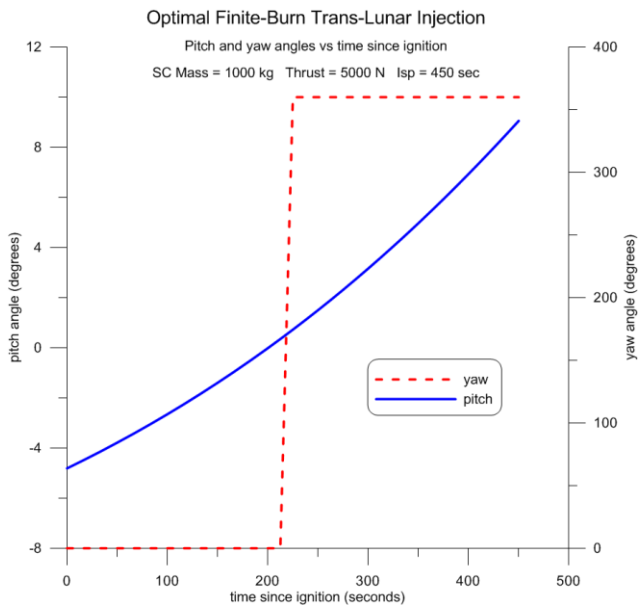
 b-plane coordinates of incoming hyperbola
 (selenocentric lunar mean equator & IAU node of epoch)

b-magnitude	5102.99831961645	kilometers
b dot r	5102.99831961645	
b dot t	1.411669927620096E-007	
theta	89.999999984150	degrees
v-infinity	891.779325708272	meters/second
r-periapsis	1837.99999911889	kilometers
decl-asy	-4.74351401653958	degrees
rasc-asy	325.911410380783	degrees
selenocentric flight path angle	3.778800121152344E-012	degrees
TLI to SOI duration	98.8279235847294	hours
TLI to closest approach duration	104.303981289268	hours

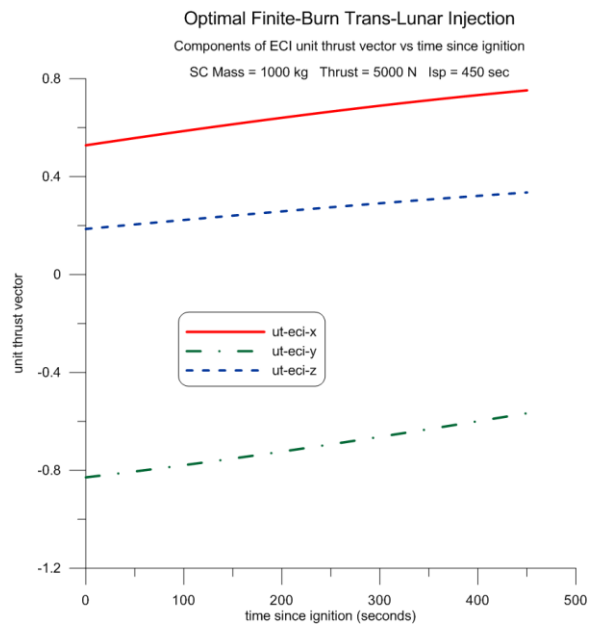
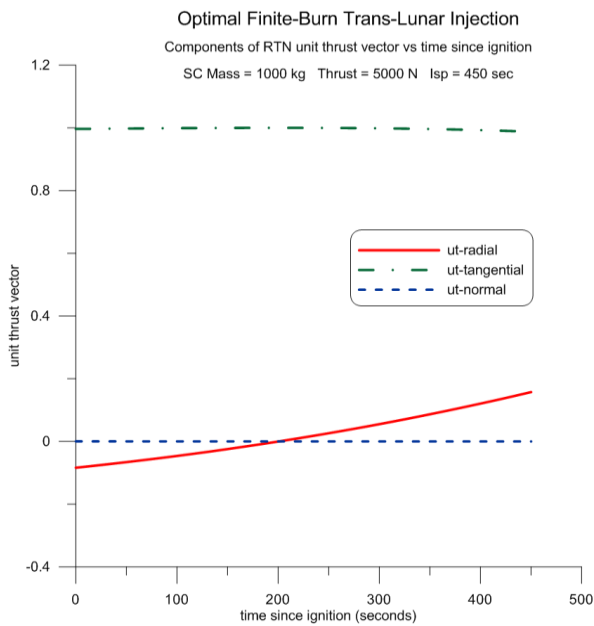
 coordinates of the moon at closest approach
 (geocentric - Earth mean equator and equinox of J2000)

calendar date	October 17, 2008		
TDB time	01:48:33.193		
TDB Julian date	2454756.57538418		
sma (km)	eccentricity	inclination (deg)	argper (deg)
0.390158366338D+06	0.674926768675D-01	0.272665922799D+02	0.638657394966D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.352326618096D+03	0.357606032273D+03	0.614717717691D+02	0.673705148186D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.210154683568D+06	0.258400912819D+06	0.146450011941D+06	0.363845611172D+06
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.881929978714D+00	0.579796000319D+00	0.235457397385D+00	0.108138988127D+01
declination	23.7349293042326	degrees	
right ascension	50.8789129511233	degrees	

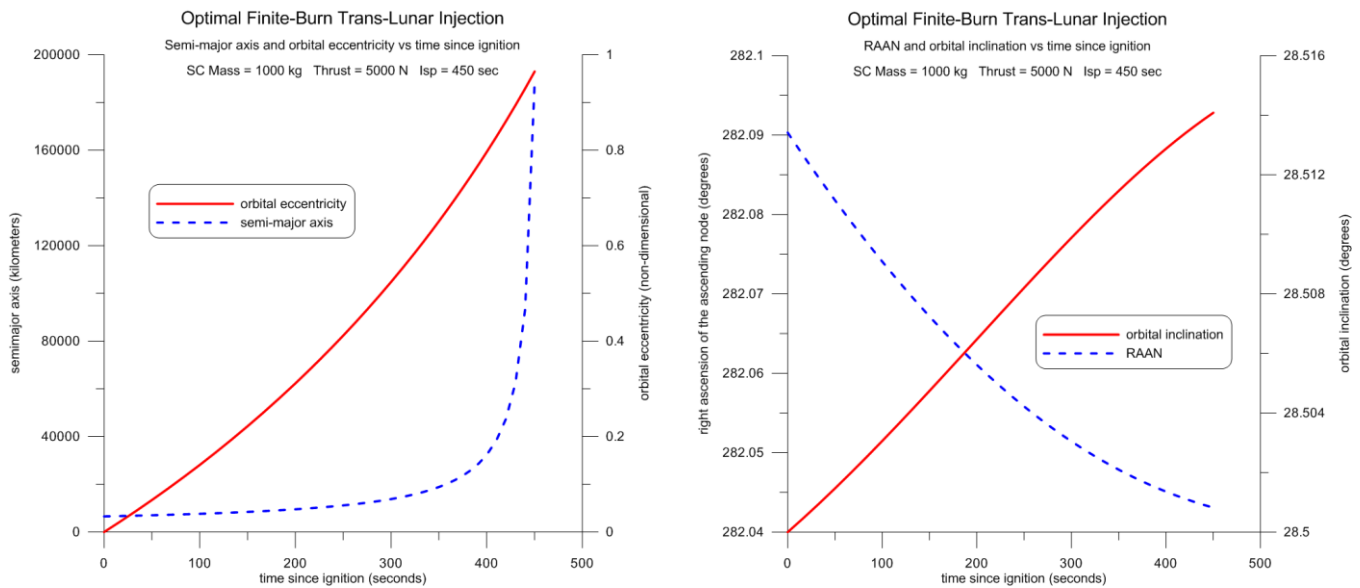
The following plots illustrate the behavior of the pitch, yaw, right ascension and declination angles during the propulsive maneuver.



These next two plots illustrate the behavior of the individual components of the RTN and ECI unit thrust vector during the maneuver.



The final two plots display the orbital evolution of the semimajor axis, eccentricity, RAAN and orbital inclination during the propulsive maneuver.



The `tlto_ocs` 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 prior to the orbital maneuver. The second file contains the state vectors and orbital elements of the geocentric transfer orbit with the name specified by the user in the main input data file. The third file is named `soiorb.csv` and contains the state vector of the selenocentric orbit at the lunar sphere-of-influence.

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

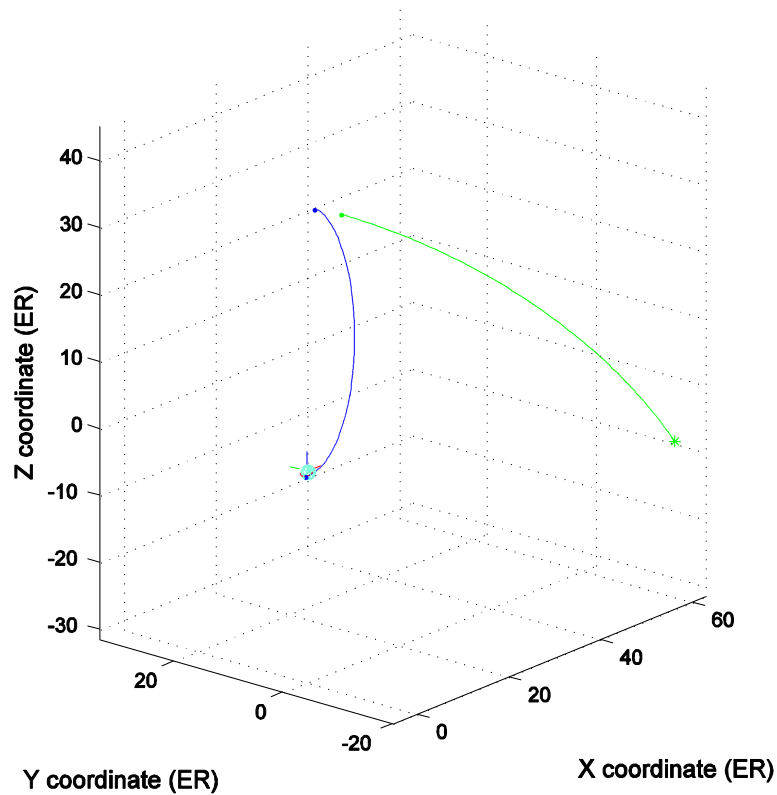
The `lplot` MATLAB script will also create two color encapsulated postscript (eps) files (with TIFF preview) of the screen images and save them to disk with the names `tlto_ocs1.eps` and `tlto_ocs2.eps`. These file names can be changed by editing the `lplot` script. The user can also change the perspective by editing the `view` command on lines 273 and 363 of the script. The `lplot` script and support functions were created with version 7.10 of MATLAB.

Important note!!

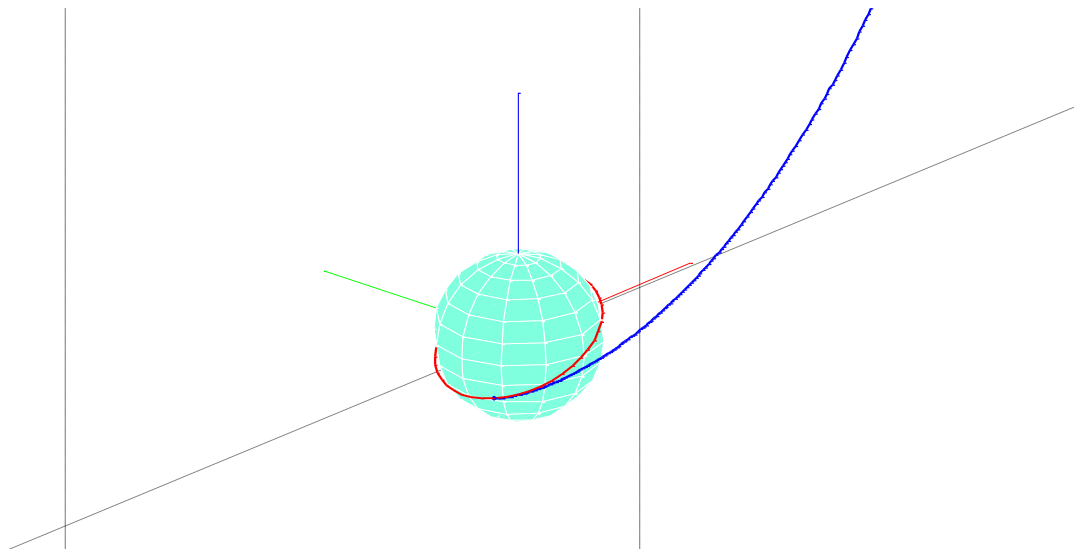
You must delete the first or “header” line of the user-defined solution file in order for the `lplot` script to work properly. This script uses the MATLAB `csvread` function to read the data file which can only contain comma-separated-variable numerical data.

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

Geocentric Transfer Trajectory

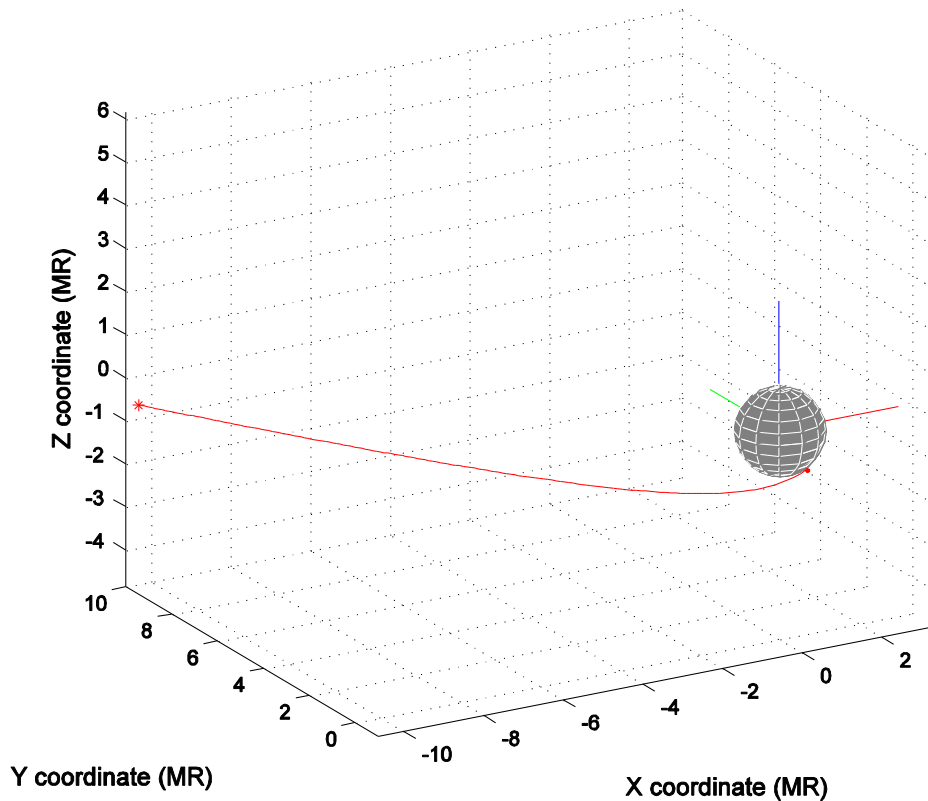


The following is a “zoomed” plot of the park orbit and initial portion of the lunar transfer trajectory.



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

Selenocentric Trajectory



Verification of the Optimal Control Solution

The optimal control solution determined by the *AMA_OC* software can be verified by numerically integrating the orbital equations of motion with the computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the *AMA_OC* optimal unit thrust vector solution. This part of the *tlto_ocs* computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is the program output of the final solution computed using this *explicit* numerical integration method.

```

=====
verification of the optimal control solution
=====

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

calendar date           October 12, 2008

TDB time                12:23:02.202

TDB Julian date        2454752.01599771

      sma (km)          eccentricity          inclination (deg)          argper (deg)
0.188695402725D+06    0.964983336589D+00    0.285140854737D+02    0.305283237994D+03

```

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.282043079806D+03	0.184533812789D+02	0.323736619273D+03	0.226595071153D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.230526001370D+04	-.608048204053D+04	-.191406227632D+04	0.677865178141D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.808777151646D+01	-.608847613587D+01	0.360702729784D+01	0.107467313955D+02

final mass	489.955135359308	kilograms
propellant mass	510.044864640692	kilograms
thrust duration	450.164832464579	seconds
delta-v	3148.41177933899	meters/second

time and conditions at lunar sphere-of-influence
(selenocentric lunar mean equator & IAU node of epoch)

calendar date	October 16, 2008
TDB time	15:13:24.191
TDB Julian date	2454756.13430777

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.612954437457D+04	0.129929593988D+01	0.902127380537D+02	0.314825045633D+03
raan (deg)	true anomaly (deg)	arglat (deg)	
0.323036880297D+03	0.230224045375D+03	0.185049091009D+03	
rx (km)	ry (km)	rz (km)	rmag (km)
-.198931037162D+05	0.149806868219D+05	-.220021165588D+04	0.249999496889D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.869849399415D+00	-.654199387101D+00	-.864507054695D-01	0.109182807265D+01

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

calendar date	October 16, 2008
TDB time	20:40:57.546
TDB Julian date	2454756.36177716

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.609392164284D+04	0.130160914303D+01	0.899993197974D+02	0.314703823629D+03
raan (deg)	true anomaly (deg)	arglat (deg)	
0.322998839177D+03	0.431376589251D-02	0.314708137395D+03	
rx (km)	ry (km)	rz (km)	rmag (km)
0.103262074110D+04	-.778187761014D+03	-.130625335880D+04	0.183798248733D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.140642762321D+01	-.105983799625D+01	0.174305035884D+01	0.247780543850D+01

```

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

b-magnitude          5077.32430462839      kilometers
b dot r              5077.32430426728
b dot t              6.055521462977254E-002
theta                89.9993166561721      degrees
v-infinity           896.961219257957      meters/second
r-periapsis          1837.98248438323      kilometers
decl-asy             -5.49584224450270     degrees
rasc-asy             322.998773730916     degrees

selenocentric flight path angle  2.439526773656330E-003  degrees

```

Creating an initial guess

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

The software allows the user to input either a delta-v magnitude or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. The user should also provide lower and upper bounds for the total thrust duration. These three inputs should be in units of seconds.

For trans-lunar injection from circular and near-circular orbits, the delta-v magnitude can be estimated from *A Computer Program for Preliminary Trans-Lunar Mission Analysis* described in Appendix D. This algorithm calculates the *impulsive* delta-v required to perform the TLI maneuver. The `lguess` computer program described in Appendix D will also provide a park orbit right ascension of the ascending node (RAAN) and the park orbit true anomaly at the TLI impulse.

For the numerical integration initial guess option, the software models the maneuver using tangential thrusting. For this case, the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = [0 \ 1 \ 0]^T$. Please note that this type of steering method creates a *coplanar* initial guess that increases the spacecraft's orbital energy.

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

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

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

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

In these equations

m_i = initial mass

m_p = propellant mass

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

I_{sp} = specific impulse

ΔV = impulsive velocity increment

F = thrust

g = acceleration of gravity

Since a finite burn maneuver will require a thrust duration longer than this impulsive estimate, the user can increase this value by about 10% and use it for the initial guess required by `tlto_ocs`.

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

Binary restart data files can also be used to initialize a `tlto_ocs` simulation. A typical scenario is

1. Create a binary restart file from a converged and optimized simulation
2. Modify the original input file with slightly different spacecraft characteristics, propulsive parameters or perhaps final mission targets and/or constraints
3. Use the previously created binary restart file as the initial guess for the new simulation

This technique works well provided the two simulations are not dramatically different. Sometimes it is necessary to make successive small changes in the mission definition and run multiples simulations to eventually reach the final desired solution.

Problem Setup

This section describes several methods that can be used to create an initial guess for `tlto_ocs`. The *AMA_OC* software attempts to obtain problem feasibility before trying to solve the optimal control problem. If the optimal control algorithm cannot find a feasible solution, the software will print warnings and the user will have to provide a better initial guess.

(1) Point functions – initial orbit and mass constraints

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

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

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

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

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

$$p_L = p_U = p_i$$

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

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

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

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

is enforced, where e is the park orbit eccentricity.

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

$$h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

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

$$-360^\circ \leq L_i \leq +360^\circ$$

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

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

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

The spacecraft mass at the initial time is constrained to the user-defined initial value.

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

(2) *Performance index – maximize final spacecraft mass*

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

$$J = m_f$$

The value of the `maxmin` indicator in the `AMA_OC` code tells the software whether the user is minimizing or maximizing the performance index. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass.

(3) *Path constraint – unit thrust vector scalar magnitude*

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

$$|\mathbf{u}_T| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

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

At the user-defined 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 scalar 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.

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

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

where v_x, v_y, v_z are the components of the spacecraft's inertial velocity vector.

In these equations, μ_m and r_{eq_m} are the gravitational constant and equatorial radius of the moon, respectively and J_{2_m} is the non-dimensional oblateness gravity coefficient. The coefficient used by this computer program corresponds to the GLGM-1 value of $2.037448533865259d-4$. The additional terms in the equations of motion, $a_{s_x}, a_{s_y}, a_{s_z}$ and $a_{e_x}, a_{e_y}, a_{e_z}$ represent the EME2000 ECI components of the point-mass gravitational acceleration due to the sun and Earth, respectively.

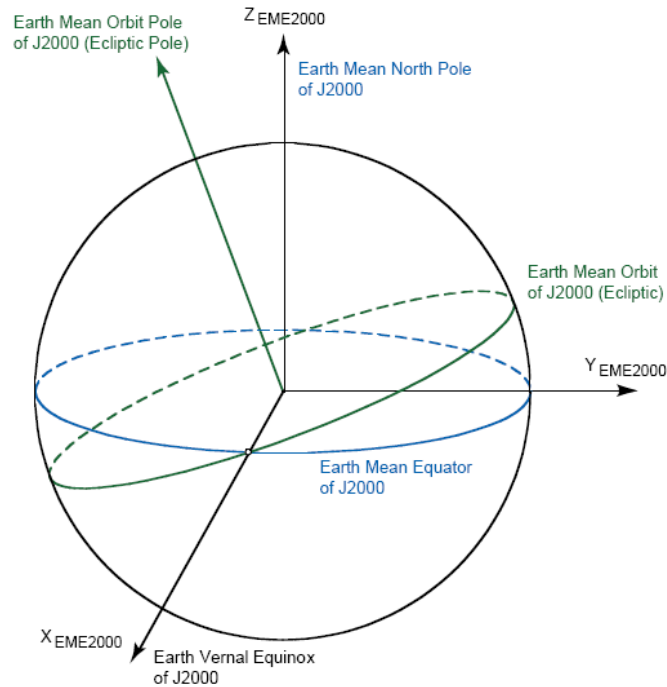
Technical Discussion

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

The spacecraft's orbital motion is modeled with respect to the Earth mean equator and equinox of J2000 (EME2000) coordinate system. The following figure illustrates the geometry of the EME2000 coordinate system. The origin of this Earth-centered-inertial (ECI) inertial coordinate system is the geocenter and the fundamental plane is the Earth's mean equator. The z-axis of this system is normal to the Earth's mean equator at epoch J2000, the x-axis is parallel to the vernal equinox of the Earth's mean orbit at epoch J2000, and the y-axis completes the right-handed coordinate system. The epoch J2000 is the Julian Date 2451545.0 which corresponds to January 1, 2000, 12 hours Terrestrial Time (TT).

Terrestrial Time (TT) is the time scale that would be kept by an ideal clock on the geoid - approximately, sea level on the surface of the Earth. Since its unit of time is the SI (atomic) second, TT is independent of the variable rotation of the Earth. TT is meant to be a smooth and continuous "coordinate" time scale independent of Earth rotation. In practice TT is derived from International Atomic Time (TAI), a time scale kept by real clocks on the Earth's surface

Barycentric Dynamical Time (TDB) is the time scale that would be kept by an ideal clock, free of gravitational fields, co-moving with the solar system barycenter. It is always within 2 milliseconds of TT, the difference caused by relativistic effects. TDB is the time scale now used for investigations of the dynamics of solar system bodies.



Modified equinoctial equations of motion (phase 1 – TLI propulsive maneuver)

The modified equinoctial orbital elements are a set of orbital elements that are useful for trajectory analysis and optimization. They are valid for circular, elliptic, and hyperbolic orbits. These equations exhibit no singularity for zero eccentricity and orbital inclinations equal to 0 and 90 degrees. However, two components of the orbital element set are singular for an orbital inclination of 180 degrees.

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

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

where

- p = semiparameter
- a = semimajor axis
- e = orbital eccentricity
- i = orbital inclination
- ω = argument of periapsis
- Ω = right ascension of the ascending node
- θ = true anomaly
- L = true longitude

The relationship between classical and modified equinoctial orbital elements is:

semimajor axis	$a = \frac{p}{1 - f^2 - g^2}$
orbital eccentricity	$e = \sqrt{f^2 + g^2}$
orbital inclination	$i = 2 \tan^{-1}(\sqrt{h^2 + k^2})$
argument of periapsis	$\omega = \tan^{-1}(g/f) - \tan^{-1}(k/h)$
right ascension of the ascending node	$\Omega = \tan^{-1}(k/h)$
true anomaly	$\theta = L - (\Omega + \omega) = L - \tan^{-1}(g/f)$

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

position vector

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

velocity vector

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

where

$$\begin{aligned} \alpha^2 &= h^2 - k^2 & s^2 &= 1 + h^2 + k^2 \\ r &= \frac{p}{w} & w &= 1 + f \cos L + g \sin L \end{aligned}$$

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

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

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

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

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

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

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

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

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

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

where

$$\mathbf{b} = \left[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \sqrt{\mu p} \left(\frac{w}{p} \right)^2 \right]^T$$

and

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

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

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

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

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

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

Non-spherical Earth gravity

The non-spherical gravitational acceleration vector can be expressed as

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

where

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

and

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

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

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

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

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

where

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

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

Finally, the zonal gravity perturbation contribution is $\mathbf{a}_g = \mathbf{Q}^T \mathbf{g}$, where $\mathbf{Q} = [\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n]$.

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

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

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

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

Secondary Body Perturbations

The general vector equation for secondary body point-mass perturbations such as the Moon, Sun or planets is given by

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

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

The perturbation due to secondary bodies in the modified equinoctial coordinate system is given by $\mathbf{a}_{TB} = \mathbf{Q}^T \mathbf{t}$, where $\mathbf{Q} = [\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n]$.

Cartesian equations of motion (Phase 2 – coast to lunar closest approach)

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

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

where

t = dynamical time

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

$\bar{\mathbf{a}}_g$ = acceleration due to the Earth's gravity $\bar{\mathbf{a}}_s$ = acceleration due to the Sun

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

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

$$\begin{aligned} \dot{y}_1 &= v_x = y_4 & \dot{y}_2 &= v_y = y_5 & \dot{y}_3 &= v_z = y_6 \\ \dot{y}_4 &= -\mu \frac{r_x}{r^3} \left\{ 1 + \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(1 - \frac{5r_z^2}{r^2} \right) \right\} + a_{m_x} + a_{s_x} + a_{T_x} \\ \dot{y}_5 &= -\mu \frac{r_y}{r^3} \left\{ 1 + \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(1 - \frac{5r_z^2}{r^2} \right) \right\} + a_{m_y} + a_{s_y} + a_{T_y} \\ \dot{y}_6 &= -\mu \frac{r_z}{r^3} \left\{ 1 + \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(3 - \frac{5r_z^2}{r^2} \right) \right\} + a_{m_z} + a_{s_z} + a_{T_z} \end{aligned}$$

where v_x, v_y, v_z are the components of the spacecraft's inertial velocity vector.

Propulsive thrust

The acceleration due to propulsive thrust can be expressed as

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

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

The components of this unit vector are the control variables for this problem.

The propellant mass flow rate is determined from

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

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g I_{sp}$ is also called the *exhaust velocity*.

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

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

$$u_{T_r} = \sin \theta \quad u_{T_t} = \cos \theta \cos \psi \quad u_{T_n} = \cos \theta \sin \psi$$

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

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

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

The relationship between a unit thrust vector in the ECI coordinate system $\hat{\mathbf{u}}_{T_{ECI}}$ and the corresponding unit thrust vector in the modified equinoctial system $\hat{\mathbf{u}}_{T_{MEE}}$ is given by

$$\hat{\mathbf{u}}_{T_{ECI}} = [\hat{\mathbf{i}}_r \ \hat{\mathbf{i}}_t \ \hat{\mathbf{i}}_n] \hat{\mathbf{u}}_{T_{MEE}}$$

where

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

This relationship can also be expressed as

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

In these equations, \mathbf{r} is the inertial position vector and \mathbf{v} is the inertial velocity vector of the spacecraft.

In the `tlto_ocs` computer program, the components of the inertial unit thrust vector are defined in terms of the right ascension α and the declination angle δ as follows:

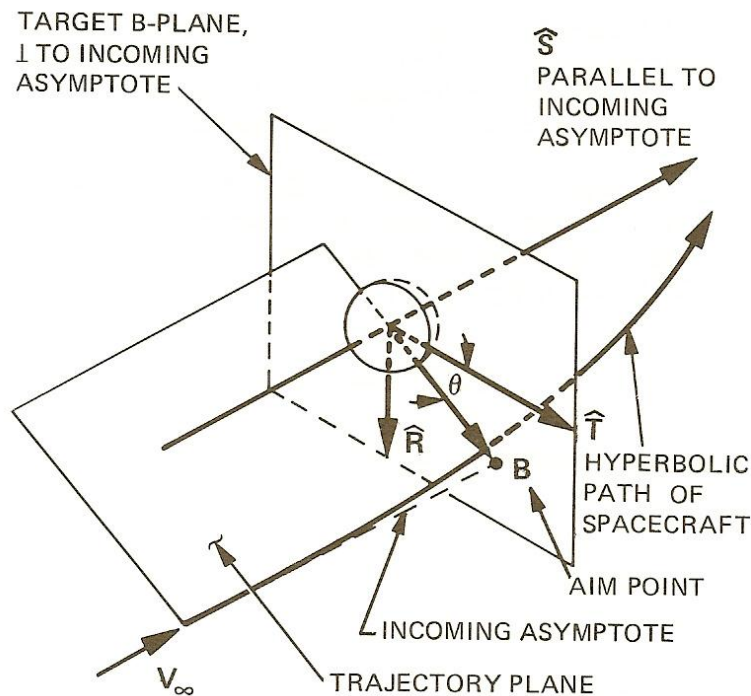
$$u_{T_{ECI_x}} = \cos \alpha \cos \delta \quad u_{T_{ECI_y}} = \sin \alpha \cos \delta \quad u_{T_{ECI_z}} = \sin \delta$$

Finally, the right ascension and declination angles can be determined from the components of the ECI unit thrust vector according to

$$\alpha = \tan^{-1}(u_{T_{ECI_y}}, u_{T_{ECI_x}}) \quad \delta = \sin^{-1}(u_{T_{ECI_z}})$$

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

semiparameter $p = h^2 / \mu$

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

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

true anomaly $\cos \theta = \frac{p - r}{er}$ $\sin \theta = \frac{\dot{r}h}{e\mu}$ $\rightarrow \theta = \tan^{-1}(\sin \theta, \cos \theta)$

B-plane magnitude $B = \sqrt{p|a|}$

fundamental vectors

$$\hat{\mathbf{z}} = \frac{r\mathbf{v} - \dot{r}\mathbf{r}}{h} \quad \hat{\mathbf{p}} = \cos \theta \hat{\mathbf{r}} - \sin \theta \hat{\mathbf{z}} \quad \hat{\mathbf{q}} = \sin \theta \hat{\mathbf{r}} + \cos \theta \hat{\mathbf{z}}$$

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

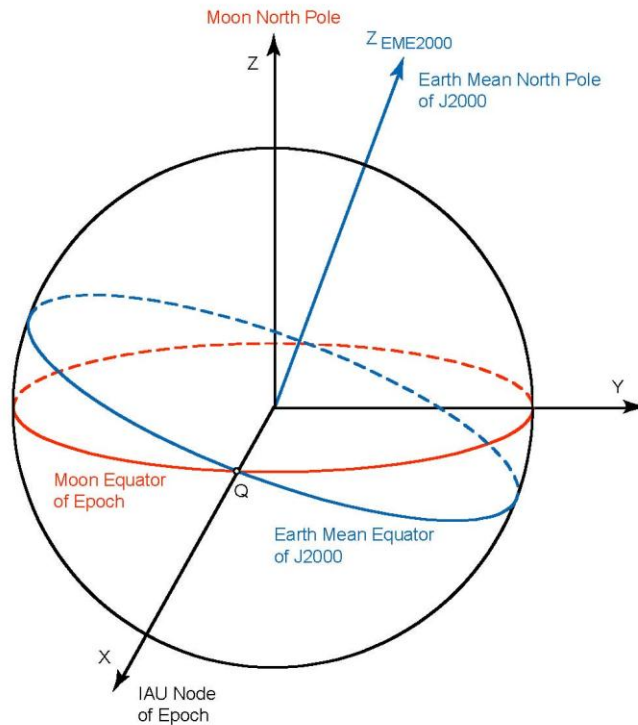
R vector

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

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.

The following diagram illustrates the orientation of the lunar mean equator and IAU node of epoch coordinate frame.



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 Working Group on Cartographic Coordinates and Rotational Elements: 2009”, *Celestial Mechanics and Dynamical Astronomy*, **109**: 101-135, 2011.

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

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

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

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

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

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

Circularization delta-v

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

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

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

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

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

A note about targeting the lunar inclination

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

$$i > |\delta_\infty|$$

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

Algorithm Resources

“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.

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

APPENDIX A

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 `tlto_ocs` 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.
period (min) = orbital period.
rx (km) = x-component of the eci position vector in kilometers
ry (km) = y-component of the eci position vector in kilometers
rz (km) = z-component of the eci position vector in kilometers
rmag (km) = geocentric position magnitude in kilometers
vx (kps) = x-component of the eci velocity vector in kilometers/second
vy (kps) = y-component of the eci velocity vector in kilometers/second
vz (kps) = z-component of the eci velocity vector in kilometers/second
vmag (kps) = velocity vector scalar magnitude in kilometers/seconds
deltav = scalar magnitude of the TLI maneuver in meters/seconds

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

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

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

APPENDIX B

Fortran Functions and Subroutines

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

tlto_ocs.f	- main executive program
atan3.for	- four quadrant inverse tangent function
csint.for	- cubic spline integration subroutine
display.for	- subroutine that displays the simulation summary
eci2mee.for	- convert eci position and velocity vectors to modified equinoctial orbital elements subroutine
eci2orb.for	- convert eci position and velocity vectors to classical orbital elements subroutine
findca.for	- subroutine to compute closest approach to the moon
fpaobj.for	- selenocentric flight path angle subroutine
fpasub.for	- subroutine that converts state vector to flight path angle
gdate.for	- convert Julian date to calendar date subroutine
geo_eqm.for	- first-order equations of motion subroutine
linput.for	- read and echo a line of text from an input file subroutine
mee2eci.for	- convert modified elements to eci state vector subroutine
mm2000.for	- lunar coordinates transformation matrix subroutine
odeigs.for	- initial guess subroutine
odeinp.for	- simulation input subroutine
odepf.for	- point functions subroutine
odeprt.for	- print subroutine - creates comma-separated-variable file
oderhs.for	- 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
rv2bp.for - convert position and velocity to b-plane coordinates subroutine
sv2000.for - lunar and solar ephemeris subroutine
ueci2umee.for - convert eci unit vector to modified unit vector subroutine
twobody2.for - two-body orbit propagation subroutine
utility.for - number and text manipulation functions and subroutines
uvector.for - unit vector subroutine
vcross.for - vector cross product subroutine
vdot.for - vector dot product subroutine
vecmag.for - vector scalar magnitude function
xmod.for - modulo 2 pi function

APPENDIX C

Example Fortran Subroutine

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

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

```

c      load cartesian state vector into local variables
c      -----

      reci(1) = y(1)

      reci(2) = y(2)

      reci(3) = y(3)

      veci(1) = y(4)

      veci(2) = y(5)

      veci(3) = y(6)

c      convert eci state vector to modified equinoctial orbital elements

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

      end if

c      extract current spacecraft mass (kilograms)

      xmass = y(7)

c      *****
c      compute point functions for state vector continuity at phase boundary
c      *****

      if (iphase .eq. 1 .and. iphend .eq. +1) then

c      -----
c      "working" eci state vector at the
c      end of the TLI propulsive maneuver
c      -----

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

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

      end if

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

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

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

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

      end if

c      *****
c      compute mission constraint point functions

```

```

c *****
if (iphase .eq. 1 .and. ioevl .eq. 2 .and. iphend .eq. -1) then
c -----
c park orbit inclination and raan prior to TLI maneuver
c -----
c compute classical orbital elements
c call eci2orb(emu, reci, veci, oev)
c cosine(inclination)
c call vcross(reci, veci, hv)
c hmag = vecmag(hv)
c ptf(1) = hv(3) / hmag
c RAAN (radians)
c ptf(2) = oev(5)
end if
if (iphase .eq. 2 .and. iphend .eq. +1) then
c -----
c selenocentric mission constraints
c -----
c xjdate = xjdate0 + time / 86400.0d0
c call sv2000(xjdate, 10, 3, rmoon, vmoon)
c state vector from the moon to the spacecraft
c do i = 1, 3
c   rtmp(i) = reci(i) - rmoon(i)
c   vtmp(i) = veci(i) - vmoon(i)
c end do
c state vector in mm2000 coordinate system
c call mm2000 (xjdate, tmatrix)
c call matxvtr(tmatrix, rtmp, rm2sc)
c call matxvtr(tmatrix, vtmp, vm2sc)
c -----
c selenocentric distance point function
c -----
c ptf(1) = vecmag(rm2sc)
c -----
c find closest approach to the moon
c -----

```

```

    call findca(xjdate, rm2sc, vm2sc, icaerr)
    if (icaerr .eq. 1) then
c       error check - close approach not found
        iferr = 1
        return
    end if
c       selenocentric periapsis radius point function (kilometers)
        ptf(2) = vecmag(rca)
c       -----
c       cos(orbital inclination) point function
c       -----
        call vcross(rca, vca, hv)
        hmag = vecmag(hv)
        ptf(3) = hv(3) / hmag
    end if
    return
end

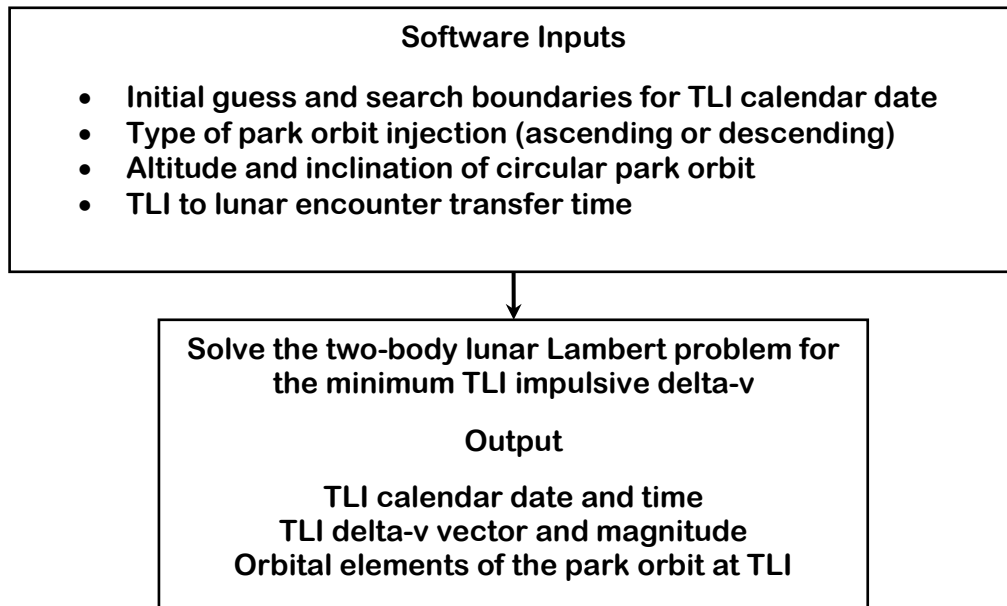
```

APPENDIX D

A Computer Program for Preliminary Trans-Lunar Mission Analysis

This appendix describes a Fortran computer program called `lguess` that can be used to perform a preliminary performance assessment of trans-lunar trajectories. The software can also be used to create an initial guess for the `tlto_ocs` computer program. This algorithm assumes that trans-lunar injection (TLI) occurs *impulsively* from a circular Earth park orbit. The software solves for the minimum TLI delta-v using a *two-body* Lambert solution for the transfer trajectory from the Earth park orbit to the center of the moon.

The program inputs, major computational step and outputs for this implementation are as follows:



This computer program uses the BOBYQA algorithm written by M.J.D. Powell to solve this classic trajectory optimization problem. The lunar coordinates required by the software are computed using the JPL DE421 ephemeris. The source code for the `lguess` computer program was created using the Intel Visual Fortran compiler, version 11.1.

Input data file

The `lguess` computer program is “data-driven” by a simple text file created by the user. This section describes a typical input data file. In the following discussion the actual input file contents are in *courier* font and all explanations are in *times italic* font.

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

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

```

*****
* input file for program lguess
* lguess1.in - December 7, 2010
*****

```

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

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

The first five inputs define the user-defined TLI calendar date and the lower and upper bounds, respectively. Be sure to include all four digits of the calendar year.

IMPORTANT: The TLI calendar date is a control variable in the NLP formulation and must always have a lower and upper bound.

```

TLI calendar date (month, day, year)
9,15,2008

lower bound for TLI calendar date search (hours)
0.0

upper bound for TLI calendar date search (hours)
+24.0

```

The next input is the user-defined number for the TLI-to-lunar encounter transfer time, in hours.

```

TLI-to-lunar encounter transfer time (hours)
110.0

```

The next two numbers define the fixed values for the park orbit altitude and orbital inclination.

```

*****
circular park orbit characteristics
*****

altitude (kilometers)
185.32

orbital inclination (degrees)
28.5

```

This next integer input defines the type of TLI maneuver to perform. The software uses this indicator to compute the park orbit RAAN. Please see the Technical Discussion later in this document for information about how the park orbit RAAN is determined.

```

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

```

Running the software

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:

```
lguess lguess1.in
```

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

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

At this point the user should input the name of a valid input file, including the filename extension.

Program example

The following is the solution created by the computer program for this example. The results are presented in the Earth mean equator and equinox of J2000 coordinate system (EME2000). The trajectory characteristics are given before and after the impulsive TLI maneuver. The geocentric orbital elements and the state vectors of the spacecraft and the moon at encounter are also displayed.

```
program lguess

minimum TLI delta-v - two-body Lambert solution
-----

DE421 ephemeris

descending TLI maneuver

transfer time          110.000000000000          hours

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

calendar date          September 15, 2008

TDB time                13:28:05.752

TDB Julian date        2454725.06117768

      sma (km)          eccentricity          inclination (deg)          argper (deg)
0.656345630000D+04    0.207703680281D-15    0.285000000000D+02    0.000000000000D+00

      raan (deg)        true anomaly (deg)          arglat (deg)          period (hrs)
0.357104409591D+03    0.242909717395D+03    0.242909717395D+03    0.146996629514D+01

      rx (km)           ry (km)           rz (km)           rmag (km)
-.324455523486D+04    -.497771531863D+04    -.278821988671D+04    0.656345630000D+04

      vx (kps)          vy (kps)          vz (kps)          vmag (kps)
0.677158909898D+01    -.346530416667D+01    -.169337334111D+01    0.779296254099D+01

impulsive TLI delta-v vector and magnitude
(geocentric EME2000 coordinates)
-----

delta-vx                2720.83248728905          meters/second
delta-vy                -1392.36667258734          meters/second
delta-vz                -680.401233802791          meters/second

deltav                  3131.22343721745          meters/second

time and conditions after TLI
```

(geocentric EME2000 coordinates)

calendar date September 15, 2008
TDB time 13:28:05.752
TDB Julian date 2454725.06117768

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.187780714768D+06	0.965047229115D+00	0.285000000000D+02	0.242909681798D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.357104409591D+03	0.355961509164D-04	0.242909717395D+03	0.224949463452D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.324455523486D+04	-.497771531863D+04	-.278821988671D+04	0.656345630000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.949242158627D+01	-.485767083926D+01	-.237377457491D+01	0.109241859782D+02
energy	-2.12269104413893	(km/sec)**2	

time and conditions at lunar encounter
(geocentric EME2000 coordinates)

calendar date September 20, 2008
TDB time 03:28:05.752
TDB Julian date 2454729.64451101

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.187780714768D+06	0.965047229115D+00	0.285000000000D+02	0.242909681798D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.357104409591D+03	0.179731146959D+03	0.626408287569D+02	0.224949463452D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.183855964261D+06	0.278989583980D+06	0.156328383523D+06	0.368885845565D+06
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.155895536718D+00	0.106160944175D+00	0.532911953610D-01	0.195993663009D+00

coordinates of the moon at encounter
(geocentric EME2000 coordinates)

calendar date September 20, 2008
TDB time 03:28:05.752
TDB Julian date 2454729.64451101

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.386687523939D+06	0.460363401223D-01	0.274675327593D+02	0.667488782745D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.352453157500D+03	0.440281280327D-04	0.667489223026D+02	0.664735266590D+03

rx (km)	ry (km)	rz (km)	rmag (km)
0.183855964261D+06	0.278989583980D+06	0.156328383523D+06	0.368885845565D+06
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.919440261341D+00	0.497446347203D+00	0.193581222756D+00	0.106315424672D+01
declination	25.0737984493215	degrees	
right ascension	56.6148560499457	degrees	

The final two items of the program output are the geocentric right ascension and declination of the moon at encounter, in the EME2000 system.

The specific orbital energy displayed by the software is calculated using the expression $E = v^2 - 2\mu/r$. TDB is the barycentric dynamic time which is the fundamental time argument for the JPL lunar ephemeris used by the code.

A brief guide to the other items displayed by the software is as follows;

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.

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

delta-vx = x-component of the TLI impulsive velocity vector in meters/second

delta-vy = y-component of the TLI impulsive velocity vector in meters/second

delta-vz = z-component of the TLI impulsive velocity vector in meters/second

deltav = scalar magnitude of the TLI maneuver in meters/seconds

Technical Discussion

This section describes several of the algorithms implemented in the `lguess` computer program.

Nonlinear programming problem

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

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

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

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

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

where \mathbf{p} is a vector of problem *parameters* that is not time dependent.

The initial dynamic variables at time t_0 are defined by $\boldsymbol{\psi}_0 \equiv \boldsymbol{\psi}[\mathbf{y}(t_0), \mathbf{u}(t_0), t_0]$ and the terminal conditions at the final time t_f are defined by $\boldsymbol{\psi}_f \equiv \boldsymbol{\psi}[\mathbf{y}(t_f), \mathbf{u}(t_f), t_f]$. These conditions are called the *boundary values* of the trajectory problem. The problem may also be subject to *path constraints* of the form $\mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), t] = 0$.

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

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

while satisfying all the user-defined mission constraints.

During the two-body trajectory optimization, the control variables are the TLI calendar date and the true anomaly of the TLI maneuver. The objective function or performance index is the scalar magnitude of the TLI delta-v vector.

In addition to the bounds on the TLI calendar date mentioned earlier, the true anomaly during the two-body optimization is bounded according to

$$-180^\circ \leq \theta \leq +180^\circ$$

The final boundary conditions are the components of the moon's inertial position vector at encounter.

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:

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

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

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

With the use of the half angle formulas given by

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

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.

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 $|\delta_m| \leq i_p$.

Additional information about these equations can be found in Appendix A of the journal reference, “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.

APPENDIX E

Typical AMA_OC Configuration File

The `tlto_ocs` computer program can read and use a user-defined configuration file. A description of each element in this file can be found in the **INSOCX** routine in section 6.2, *Subprograms for Optimal Control*, and the **INSNLP** routine in Section 2.2, *Subprograms for Optimization* of the *AMA_OC* user's manual. Please note that the `tlto_ocs` software can read and use a subset of the information in this file. For example, a subset configuration file might contain only the following information;

```
ODETOL=0.1D-06
INSNLP:IOFLAG=5
SOCOUT=I4K4
```

The following is a typical “full version” configuration file created during the execution of the `tlto_ocs` software.

```
AEQTOL=0.1000000000000000D-02
DTAUX=0.0000000000000000D+00
OBJCTL=0.1000000000000000D-04
ODETOL=0.1000000011686097D-06
PGDCTL=0.1000000000000000D-02
PRTMSD=0.1490116119384766D-07
PRTMXD=0.1000000000000000D-02
PRTSFD=0.1000000000000000D-04
QDRTOL=0.1000000000000000D-02
RESTOL=0.1000000000000000D-04
SMLTOL=0.1490116119384766D-10
TOLJSD=0.1000000000000000D-05
TOLM5A=0.1490116119384766D-07
TOLM5R=0.1490116119384766D-07
IDSCPH=0
IDSCND=0
IDSCVR=0
IDSCFN=0
IDTSFD=-1
IPFAUX=0
IPFSFD=0
IPRSFD=1
IPGRD=0
IPNLP=10
IPODE=0
IPUAUX=0
IPUOCP=6
IRSTRT=2
ISCALE=0
ISFHES=41
ISFINP=42
ISFRST=43
ISFSCL=44
ITSWCH=2
M5DTYP=0
MITODE=20
MTSWCH=-1
MXDATA=0
MXPARM=10
MXPCON=20
MXSTAT=20
MXTERM=50
NPTAUX=100
NSSWCH=-1
SOCOUT=A0B0C0D0E0F0G0H0I0J2K0L0M0N0O0P0Q0R0S1T0U0V0W0X0Y0Z0
SPRTHS=SPARSE
NLPALG=SNLPMN
NLPOMR=M
KEYDPL=.lueiLUE
RHSTMP=RHSTMPLT
```

RSTFIL=tlto1.rsin
SCLFIL=scalewgt.fil
INSNLP:ALFLWR=0.000000000000000D+00
INSNLP:ALFUPR=0.100000000000000D+01
INSNLP:CONTOL=0.1490116119384766D-07
INSNLP:EPSRLF=0.1490116119384766D-07
INSNLP:OBJTOL=0.9999999747378752D-05
INSNLP:PGDTOL=0.100000000000000D-04
INSNLP:SLPTOL=0.900000000000000D+00
INSNLP:SFZTOL=0.100000000000000D-01
INSNLP:TOLFIL=0.200000000000000D+01
INSNLP:TOLKTC=0.1110953834938985D+26
INSNLP:TOLPVT=0.100000000000000D-02
INSNLP:IHESHN=0
INSNLP:IOFLAG=5
INSNLP:IOFLIN=-1
INSNLP:IOFMFR=0
INSNLP:IOFPAT=0
INSNLP:IOFSHR=0
INSNLP:IOFSRC=0
INSNLP:IPUDRF=0
INSNLP:IPUFZF=0
INSNLP:IPUMF1=11
INSNLP:IPUMF2=12
INSNLP:IPUMF3=13
INSNLP:IPUMF4=14
INSNLP:IPUMF5=15
INSNLP:IPUMF6=16
INSNLP:IPUMF7=17
INSNLP:IPUNLP=6
INSNLP:IPUSTF=0
INSNLP:IRELAX=1
INSNLP:ITDRQP=-1
INSNLP:ITFZQP=-1
INSNLP:IT1MAX=20
INSNLP:JACPRM=0
INSNLP:LYNFNC=0
INSNLP:LYNOUT=0
INSNLP:LYNPLT=0
INSNLP:LYNPNT=101
INSNLP:LYNVAR=0
INSNLP:MAXLYN=5
INSNLP:MAXNFE=50000
INSNLP:MNSAME=2
INSNLP:NEWTON=0
INSNLP:NITMAX=1000
INSNLP:NITMIN=0
INSNLP:NORMAL=0
INSNLP:ALGOPT=FM
INSNLP:KTOPTN=SMALL
INSNLP:QPOPTN=SPARSE
INSNLP:BIGCON=-0.100000000000000D+01
INSNLP:FEATOL=0.100000000000000D-01
INSNLP:PMULWR=0.100000000000000D+00
INSNLP:PTHOL=0.100000000000000D+02
INSNLP:RHOLWR=0.100000000000000D+03
INSNLP:IMAXMU=10
INSNLP:MUCALC=3
INSNLP:MXQPIT=1