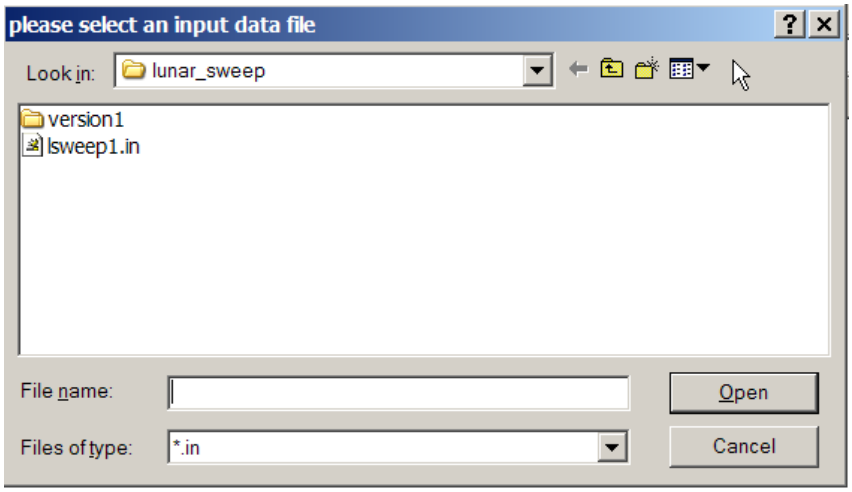
# A MATLAB Script for Parametric Analysis of Minimum TLI Delta-V Lunar Transfer Trajectories – OTB Version

This document is the user's guide for a MATLAB script called `tli_sweep_otb` that can be used to perform a parametric analysis of two-body and "perturbed" lunar transfer trajectories. The software assumes that trans-lunar injection (TLI) occurs *impulsively* from a circular Earth park orbit. The software initially solves for the minimum TLI delta-v using a two-body Lambert solution for the transfer trajectory from Earth park orbit to the center of the Moon. The perturbed solution uses this two-body answer as an initial guess for solving the two-point boundary-value problem subject to perturbations due to a non-spherical Earth gravity model and optionally, the point-mass gravity of the Sun.

This MATLAB script uses a nonlinear programming (NLP) method from the Optimization Toolbox for both optimization tasks required for the lunar transfer problem implemented in this script. The solar and lunar coordinates required by this script are computed using the JPL DE430 ephemeris.

## Interacting with the script

When the `tli_sweep_otb` script is started, the software will display the following screen which allows the user to select a data file for processing.




The file type defaults to names with a `*.in` filename extension. However, you can select any `tli_sweep_otb` compatible ASCII data file.

## Input data file format and contents

The `tli_sweep_otb` MATLAB script is "data-driven" by a simple text file created by the user. This section describes a typical input data file. In the following discussion the actual input file contents are in *`courier`* bold 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.*

```
*****************************************
* data file for tli_sweep_otb.m MATLAB script
* tli_sweep1.in     July 16, 2021
*****************************************
```

*The first input defines the type of orbital motion to model during the trajectory optimization.*

```
type of optimization
 1 = two-body orbital motion
 2 = perturbed orbital motion
-----------------------------
2
```

*The next inputs define the calendar date of the TLI maneuver. Be sure to include all four digits of the calendar year.*

```
initial calendar date (month, day, year)
9, 1, 2013
```

*The next input specifies the type of TLI maneuver. Please see the Technical Discussion later in this document for an explanation of this maneuver.*

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

*The next two inputs define the value of the altitude and orbital inclination of the initial circular Earth park orbit.*

```
park orbit altitude (kilometers)
185.32

park orbit inclination (degrees)
28.5
```

*The duration of the lunar transfer trajectory in hours is set by this next input.*

```
transfer time (hours)
84.0
```

*The name of the Earth gravity model data file is specified in this next statement.*

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

*The next two inputs define the order and degree of the gravity model to use during the simulation.*

```
order of the Earth gravity model (zonals)
-----------------------------------------
8

degree of the Earth gravity model (tesserals)
---------------------------------------------
8
```

The next integer input determines if the simulation should include the point-mass gravity of the Sun during the trajectory optimization.

```
include point-mass gravity of the Sun (1 = yes, 0 = no)
-------------------------------------------------------
1
```

*The total simulation duration and time step size of the parametric sweep are specified using these next two inputs.*

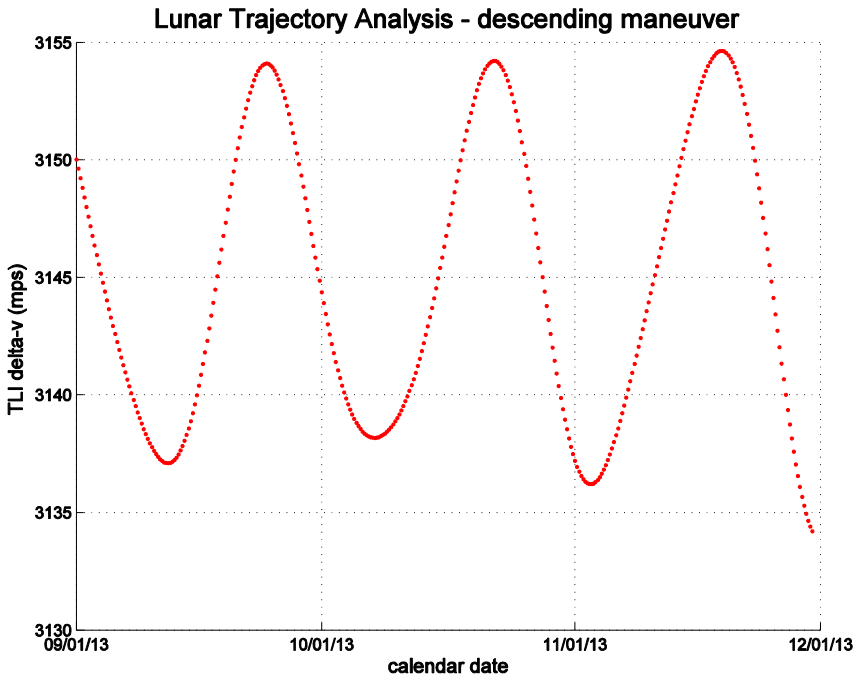
```
simulation duration (days)
90

simulation step size (days)
0.25
```

*The final input is the name of the data file created by the software.*

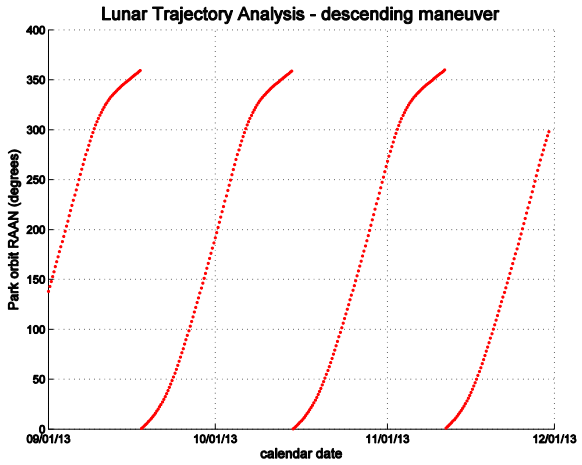
```
name of summary data file
tli_sweep1.txt
```

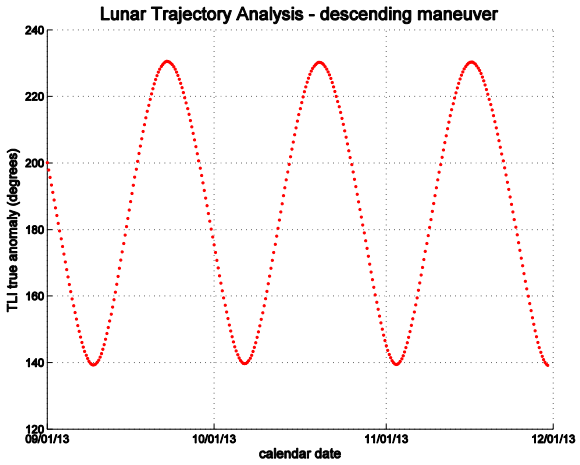
## Script example

This section contains plots of the behavior of the TLI characteristics for a two-body, descending node transfer for a period of three months at a step size of 0.25 days. The initial calendar date and time is 0 hours UTC on August 1, 2013 and the transfer time from TLI until lunar encounter is 84 hours. The first plot shows the magnitude of the TLI impulsive delta-v as a function of the calendar date.

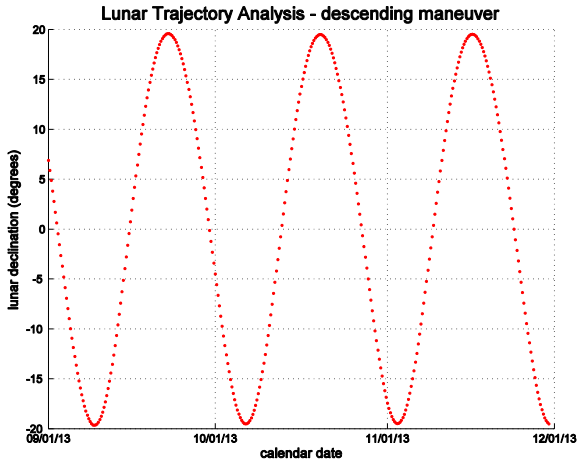


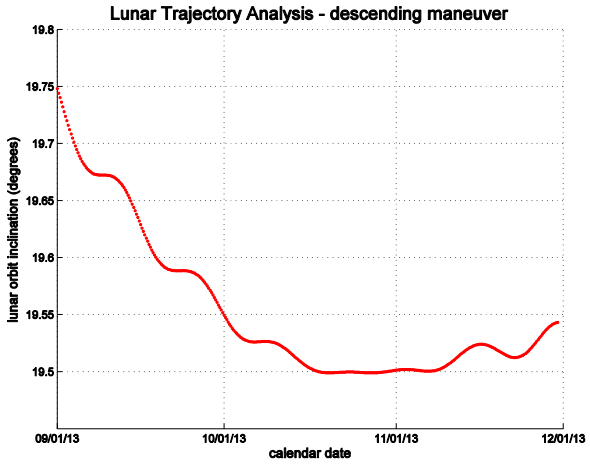

The next two plots illustrate the variation of the RAAN of the circular park orbit and the true anomaly location of the TLI impulse as a function of the TLI calendar date.




The final two plots illustrate the geocentric declination of the Moon at the encounter time and the evolution of the geocentric orbital inclination of the Moon.



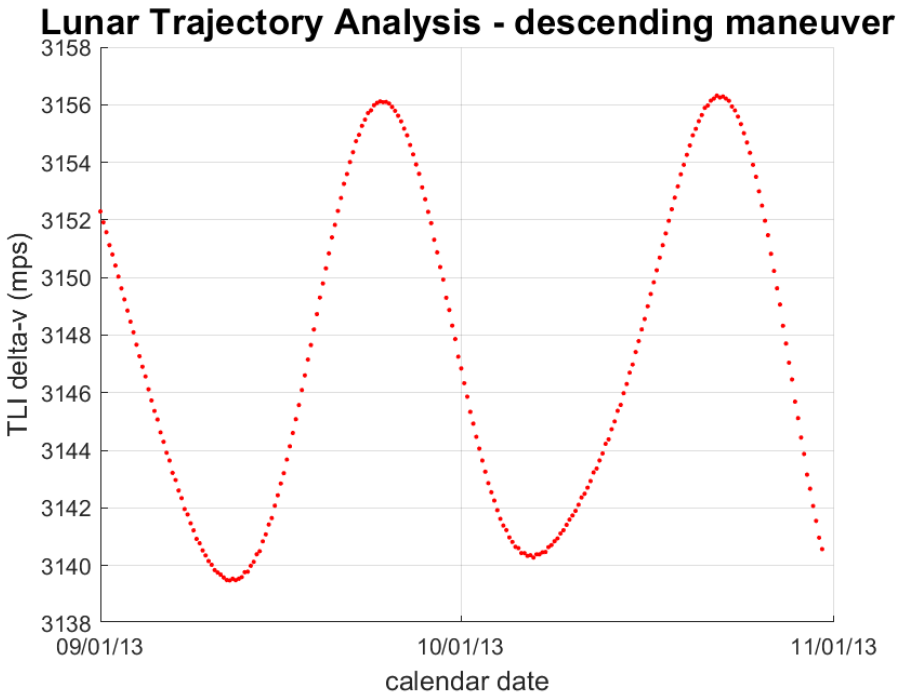

These plots are saved to disk files in tiff format.

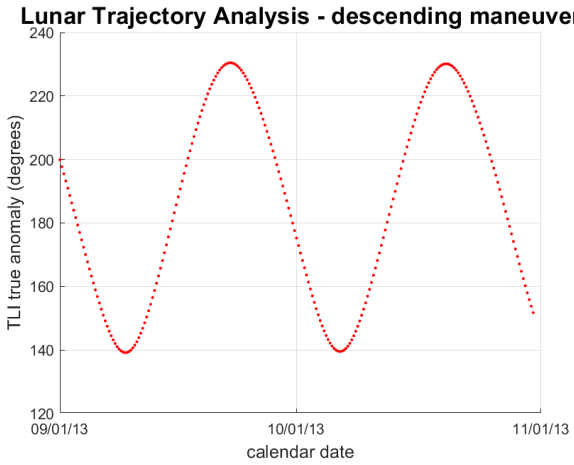
The following printout illustrates the first five days of the ASCII data file created by the software.

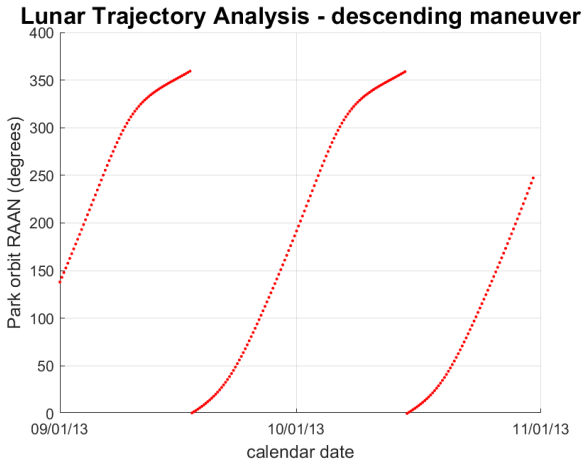
| time (days) | delta-v (mps) | RAAN (deg) | tanom (deg) | C3 (km/sec)^2 | rasc moon (deg) | decl moon (deg) |
|---|---|---|---|---|---|---|
| 0.0000 | 3150.02069561 | 138.00961114 | 200.07071238 | -1.71164825 | 150.84989646 | 6.88023307 |
| 0.2500 | 3149.62288247 | 142.90173568 | 197.87523718 | -1.72035461 | 153.82937906 | 5.87662682 |
| 0.5000 | 3149.21971487 | 147.81512273 | 195.65007537 | -1.72917784 | 156.81142964 | 4.85291194 |
| 0.7500 | 3148.81263300 | 152.74946810 | 193.39886425 | -1.73808640 | 159.79784314 | 3.81167716 |
| 1.0000 | 3148.40299341 | 157.70471293 | 191.12519626 | -1.74705060 | 162.79049282 | 2.75560191 |
| 1.2500 | 3147.99206325 | 162.68098569 | 188.83265144 | -1.75604271 | 165.79132040 | 1.68745656 |
| 1.5000 | 3147.58101586 | 167.67854192 | 186.52483009 | -1.76503704 | 168.80232542 | 0.61010247 |
| 1.7500 | 3147.17092753 | 172.69770104 | 184.20538601 | -1.77401005 | 171.82555366 | -0.47350832 |
| 2.0000 | 3146.76277581 | 177.73877949 | 181.87806107 | -1.78294035 | 174.86308441 | -1.56033374 |
| 2.2500 | 3146.35743907 | 182.80201957 | 179.54672145 | -1.79180873 | 177.91701649 | -2.64724273 |
| 2.5000 | 3145.95569746 | 187.88751284 | 177.21539618 | -1.80059813 | 180.98945285 | -3.73101696 |
| 2.7500 | 3145.55823538 | 192.99511743 | 174.88831860 | -1.80929358 | 184.08248354 | -4.80835315 |
| 3.0000 | 3145.16564515 | 198.12436837 | 172.56997157 | -1.81788214 | 187.19816692 | -5.87586616 |
| 3.2500 | 3144.77843210 | 203.27438007 | 170.26513666 | -1.82635276 | 190.33850903 | -6.93009315 |
| 3.5000 | 3144.39702090 | 208.44374061 | 167.97894796 | -1.83469617 | 193.50544086 | -7.96749878 |
| 3.7500 | 3144.02176298 | 213.63039747 | 165.71695097 | -1.84290469 | 196.70079368 | -8.98448171 |
| 4.0000 | 3143.65294511 | 218.83153504 | 163.48516626 | -1.85097206 | 199.92627228 | -9.97738276 |
| 4.2500 | 3143.29079881 | 224.04344488 | 161.29015789 | -1.85889324 | 203.18342634 | -10.94249466 |
| 4.5000 | 3142.93551064 | 229.26139084 | 159.13910508 | -1.86666416 | 206.47362019 | -11.87607389 |
| 4.7500 | 3142.58723310 | 234.47947224 | 157.03987532 | -1.87428149 | 209.79800126 | -12.77435468 |
| 5.0000 | 3142.24609597 | 239.69049079 | 155.00109562 | -1.88174242 | 213.15746788 | -13.63356545 |

*Perturbed orbital motion graphics and data*

The following plots illustrate the park orbit true anomaly, RAAN and TLI delta-v magnitude for the perturbed orbital motion option for the first 60 days of the previous example. This plot includes the effect of the point-mass gravity of the Sun along with an 8 by 8 Earth gravity model.




Here is the data for the first two days for the perturbed orbital motion simulation.

| time (days) | delta-v (mps) | RAAN (deg) | tanom (deg) | C3 (km/sec)^2 | rasc moon (deg) | decl moon (deg) |
|---|---|---|---|---|---|---|
| 0.0000 | 3152.29546089 | 138.06172651 | 199.92780258 | -1.66186086 | 150.84989646 | 6.88023307 |
| 0.2500 | 3151.91467875 | 142.96363268 | 197.72425000 | -1.67019539 | 153.82937906 | 5.87662682 |
| 0.5000 | 3151.57490656 | 147.88653492 | 195.49396659 | -1.67763274 | 156.81142964 | 4.85291194 |
| 0.7500 | 3151.12556692 | 152.83196524 | 193.23308390 | -1.68746702 | 159.79784314 | 3.81167716 |
| 1.0000 | 3150.80095143 | 157.79541866 | 190.95651149 | -1.69457229 | 162.79049282 | 2.75560191 |
| 1.2500 | 3150.41701164 | 162.78253567 | 188.65588224 | -1.70297503 | 165.79132040 | 1.68745656 |
| 1.5000 | 3150.03483212 | 167.79046940 | 186.34308673 | -1.71133941 | 168.80232542 | 0.61010247 |
| 1.7500 | 3149.62396902 | 172.82072977 | 184.01432373 | -1.72033089 | 171.82555366 | -0.47350832 |
| 2.0000 | 3149.24062914 | 177.86988898 | 181.68669750 | -1.72872026 | 174.86308441 | -1.56033374 |

Here is a description of the items captured in this data file.

**time (days)** = elapsed time from initial TLI calendar date in days

**delta-v (mps)** = magnitude of impulsive TLI maneuver in meters per second

**RAAN (deg)** = park orbit right ascension of the ascending node in degrees

**tanom (deg)** = park orbit true anomaly of the TLI maneuver in degrees

**C3 (km/sec)^2** = twice the specific orbital energy in kilometers^2/second^2

**rasc moon (deg)** = geocentric right ascension of the Moon in degrees at arrival

**decl moon (deg)** = geocentric declination of the Moon in degrees at arrival

## Technical discussion

This section provides additional details about the numerical algorithms used in this computer program. The computational methods discussed here include solving the two body Lambert problem, the method used for propagating the spacecraft's geocentric trajectory, the algorithms used for enforcing mission constraints. A brief discussion of trajectory optimization is also provided along with the MATLAB source code for several important functions.

*Solving the two body Lambert problem*

The algorithm used in this MATLAB script 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.

A summary of the Lambert problem is described in Appendix A.

*Nonlinear programming problem*

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

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

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

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

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

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

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

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

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

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

while satisfying all the user-defined mission constraints.

During the two-body trajectory optimization, the main control variable is the park orbit true anomaly at the time of the TLI maneuver. The objective function or performance index is the scalar magnitude of the TLI delta-v vector. The final boundary conditions are the Earth-centered-inertial (ECI) components of the Moon's inertial position vector at encounter. These equality boundary conditions can be expressed as $\mathbf{r}_p - \mathbf{r}_t = 0$ where $\mathbf{r}_p$ is the position vector of the Moon predicted by the software and $\mathbf{r}_t$ is the "targeted" or desired lunar position vector. The targeted position vector is determined by evaluating the JPL ephemeris at the final encounter time with the Moon.

During the perturbed orbital motion optimization, the control variables are the park orbit true anomaly at TLI, the park orbit right ascension of the ascending node (RAAN) and the ECI components of the TLI delta-v vector. The objective function or performance index is the scalar magnitude of the TLI delta-v vector. The final boundary conditions are the Earth-centered-inertial (ECI) components of the Moon's inertial position vector at encounter. This script option uses the park orbit RAAN and TLI maneuver true anomaly computed by the two-body solution as initial guesses for the perturbed motion optimization.

*Park orbit true anomaly*

An initial guess for the park orbit true anomaly at the time of the impulsive TLI maneuver is obtained iteratively. This iteration involves setting the park orbit true anomaly equal to the argument of perigee of the TLI maneuver of the Earth-to-Moon elliptical transfer orbit. This process is repeated until the change in true anomaly between successive iterations is small. The argument of perigee of the transfer orbit is determined by re-solving the two-body Lambert problem which is a function of the current park state vector and the position and velocity vectors of the Moon at encounter.

During the two-body numerical optimization, the true anomaly control variable is bounded according to $\theta_1 - 30^\circ \le \theta \le \theta_1 + 30^\circ$ where $\theta_1$ is the true anomaly computed from the initial guess iteration.

During the *perturbed* motion numerical optimization, the true anomaly is bounded according to $\theta_{TB} - 10^\circ \le \theta \le \theta_{TB} + 10^\circ$ and the park orbit RAAN is bounded according to $\Omega_{TB} - 10^\circ \le \Omega \le \Omega_{TB} + 10^\circ$ where $\theta_{TB}$ is the true anomaly and $\Omega_{TB}$ the RAAN computed by the two-body solution.

*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, $\Omega_p$ 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.

In the two-body optimization, the park orbit RAAN is held fixed during the numerical optimization according to the RAAN option selected by the user.

The equations implemented in this MATLAB script are as follows:

*ascending maneuver*

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

*descending maneuver*

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

where

$\alpha_m =$ right ascension of the Moon at encounter

$\delta_m =$ declination of the Moon at encounter

$i_p =$ park orbit inclination

*Modeling the perturbed orbital motion*

For the perturbed orbital motion option, the `tli_sweep_otb` MATLAB script implements a *special perturbation* technique which numerically integrates the vector system of second-order, nonlinear differential equations of motion given by

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

where

$t =$ UTC time

$\mathbf{r} =$ inertial position vector

$\mathbf{a}_g =$ acceleration due to Earth gravity

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

*Geocentric acceleration due to non-spherical Earth gravity*

The software 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 $\phi$ is the geocentric latitude, $\lambda$ is the geocentric east longitude and $r=|\vec{r}|=\sqrt{x^2+y^2+z^2}$ is the geocentric distance. In this expression the *S's* and *C's* are 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 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 non-spherical terms in the Earth's geopotential. In terms of the Earth's geopotential $\Phi$, the inertial rectangular cartesian components of the 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 $r,\phi,\lambda$ 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

$R =$ radius of the Earth

$r =$ geocentric distance

$S_n^m, C_n^m =$ harmonic coefficients

$\phi =$ geocentric declination $= \sin^{-1}\left(\dfrac{z}{r}\right)$

$\lambda =$ longitude $= \alpha - \alpha_g$

$\alpha =$ right ascension $= \tan^{-1}\left(\dfrac{y}{x}\right)$

$\alpha_g =$ right ascension of Greenwich

The right ascension is measure positive east of the vernal equinox, longitude is measured positive east of Greenwich, and declination is positive above the Earth's equator and negative below.

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

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

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

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

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

where the first few associated Legendre functions are given by

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

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

The trigonometric arguments are determined from expansions given by

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

The gravity model data file used in this simulation is a simple space delimited ASCII data file. The following is a portion of a typical gravity model data file. In this file, column one is the degree index, column two is the model order index, and columns three and four are the corresponding *un-normalized* gravity coefficients (zonals and tesserals, respectively).

```
2   0  -0.10826300D-02   0.00000000D+00
3   0   0.25321531D-05   0.00000000D+00
4   0   0.16109876D-05   0.00000000D+00
```

```
 5   0   0.23578565D-06   0.00000000D+00
 6   0  -0.54316985D-06   0.00000000D+00
 7   0   0.33237640D-06   0.00000000D+00
 8   0   0.17721040D-06   0.00000000D+00
 9   0   0.14459876D-06   0.00000000D+00
10   0   0.23339780D-06   0.00000000D+00
```

Gravity model coefficients are often published in *normalized* form. The relationship between normalized $\bar{C}_{l,m}, \bar{S}_{l,m}$ and un-normalized gravity coefficients $C_{l,m}, S_{l,m}$ is given by the following expression:

$$\begin{Bmatrix} \bar{C}_{l,m} \\ \bar{S}_{l,m} \end{Bmatrix} = \left[ \frac{1}{(2-\delta_{m0})(2l+1)} \frac{(l+m)!}{(l-m)!} \right]^{1/2} \begin{Bmatrix} C_{l,m} \\ S_{l,m} \end{Bmatrix}$$

where $\delta_{m0}$ is equal to 1 if *m* is zero and equal to zero if *m* is greater than zero.

*Geocentric acceleration due to the point-mass gravity of the Sun*

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

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

where

$\mu_s =$ gravitational constant of the Sun

$\mathbf{r}_{s-sc} =$ position vector from the Sun to the trajectory

$\mathbf{r}_{e-s} =$ position vector from the Earth to the Sun

The first-order system of equations required by this computer program can be created from the second-order system by the method of *order reduction*. With the following definitions,

$$y_1 = r_x \qquad y_2 = r_y \qquad y_3 = r_z \qquad y_4 = v_x \qquad y_5 = v_y \qquad y_6 = v_z$$

where $v_x, v_y, v_z$ are *x*, *y* and *z* components of the velocity vector, the first-order system of differential equations is given by,

$$\dot{y}_1 = v_x \qquad \dot{y}_2 = v_y \qquad \dot{y}_3 = v_z$$

$$\dot{y}_4 = -\mu_e \frac{r_x}{r^3} + a_{x-s} \quad \dot{y}_5 = -\mu_e \frac{r_y}{r^3} + a_{y-s} \quad \dot{y}_6 = -\mu_e \frac{r_z}{r^3} + a_{z-s}$$

In these equations, $\mu_e$ is the gravitational constant of the Earth and $a_{x-s}$, $a_{y-s}$ and $a_{z-s}$ are the *x*, *y* and *z* point-mass gravitational contributions of the Sun.

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

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

In these equations, $\mathbf{s}_k$ is the vector from the primary body to the secondary body, $\mu_k$ is the gravitational constant of the secondary body and $\mathbf{d}_k = \mathbf{r}-\mathbf{s}_k$, where $\mathbf{r}$ is the position vector relative to the primary body. The derivation of the $f\left(q\right)$ functions is described in Section 8.4 of *An Introduction to the Mathematics and Methods of Astrodynamics*, Revised Edition", by Richard H. Battin, AIAA Education Series, 1999.

In this computer program the geocentric coordinates of the sun and moon are based on the JPL Development Ephemeris DE430. These coordinates are evaluated in the Earth mean equator and equinox of J2000 coordinate system (EME2000). 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). Appendix B describes the relationship between these time systems.

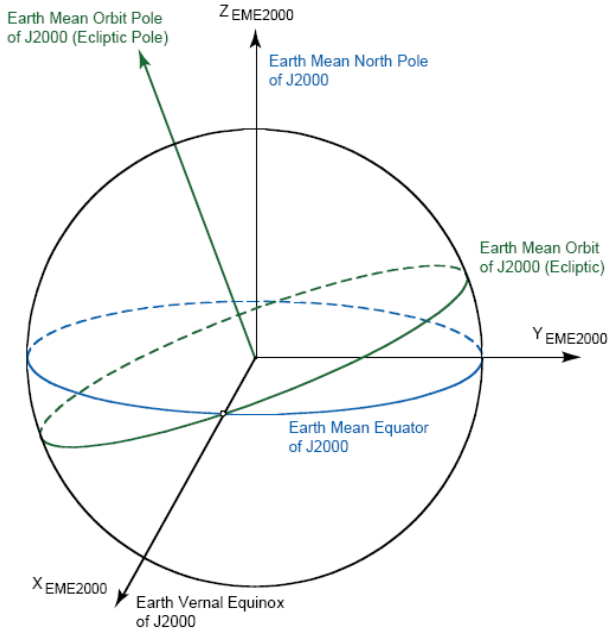



Figure 1. Earth mean equator and equinox of J2000 coordinate system

The EME2000 coordinates of the sun and moon are transformed to the true-of-date system using a MICE *frame definition*. The following are the contents of this frame definition file.

```
\begintext

Earth true-of-date frame definition

\begindata

FRAME_EARTH_TOD               =  398600
FRAME_398600_NAME             =  'EARTH_TOD'
FRAME_398600_CLASS            =  5
FRAME_398600_CLASS_ID         =  398600
FRAME_398600_CENTER           =  399
FRAME_398600_RELATIVE         = 'J2000'
FRAME_398600_DEF_STYLE        = 'PARAMETERIZED'
FRAME_398600_FAMILY           = 'TRUE_EQUATOR_AND_EQUINOX_OF_DATE'
FRAME_398600_PREC_MODEL       = 'EARTH_IAU_1976'
FRAME_398600_NUT_MODEL        = 'EARTH_IAU_1980'
FRAME_398600_ROTATION_STATE   = 'INERTIAL'
```

This frame definition uses the IAU 1976 model for precession and the IAU 1980 model for nutation.

## 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.

"Lunar Flight Handbook", NASA SP-34, Parts 1, 2, 3, 1963.

"User's Guide for SNOPT Version 7, A Fortran Package for Large-Scale Nonlinear Programming", Philip E. Gill, Walter Murray and Michael A. Saunders, April 2007.

Lunar Free-Return Trajectory Analysis with MATLAB
https://www.mathworks.com/matlabcentral/fileexchange/73659-lunar-free-return-trajectory-analysis-with-matlab-otb

A MATLAB Script for Propagating Trajectories from the Earth to the Moon
https://www.mathworks.com/matlabcentral/fileexchange/43067-a-matlab-script-for-propagating-trajectories-from-the-earth-to-the-moon

# Appendix A

## Numerical Solutions of Lambert's Problem

Lambert's problem is concerned with the determination of a two-body 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 functionally as $tof = tof\left(r_i + r_f, c, a\right)$.

From the following form of Kepler's equation

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

we can write

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

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 sun and difference identities:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## Gooding's solution of Lambert's problem

The algorithm used in this script 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.

## Gedeon's solution of Lambert's problem

Another practical numerical method for solving Lambert's problem is described in "A Practical Note on the Use of Lambert's Equation" by Geza Gedeon, *AIAA Journal*, Volume 3, Number 1, 1965, pages 149-150. 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. Additional information can also be found in G. S. Gedeon, "Lambertian Mechanics", Proceedings of the 12th International Astronautical Congress, Vol. I, 172-190.

The *elliptic* form of the general Lambert Theorem is

$$t = \sqrt{\frac{a^3}{\mu}}\left[\left(1-k\right)m\pi + k\left(\alpha - \sin\alpha\right) \mp \left(\beta - \sin\beta\right)\right]$$

where *k* may be either +1 (posigrade) or –1 (retrograde), and *m* is the number of revolutions about the central body.

The Gedeon algorithm introduces the following parameter

$$z = \frac{s}{2a}$$

and solves the problem with a Newton-Raphson procedure. In this equation, *a* is the semimajor axis of the transfer orbit and

$$s = \frac{r_1 + r_2 + c}{2}$$

This algorithm also makes use of the following constant

$$w = \pm\sqrt{1 - \frac{c}{s}}$$

The function to be solved iteratively is given by:

$$N\left(z\right) = \frac{1}{z\left|z\right|^{1/2}2^{1/2}}\left\{\frac{1-k}{2}m\pi + k\left[\left|z\right|^{1/2} - \left|z\right|^{1/2}\left(1-z\right)^{1/2}\right] - \left[w\left|z\right|^{1/2} - w\left|z\right|^{1/2} - w\left|z\right|^{1/2}\left(1-w^2z\right)^{1/2}\right]\right\}$$

The Newton-Raphson algorithm also requires the derivative of this equation given by

$$N'(z) = \frac{dN}{dz} = \frac{1}{|z|2^{1/2}} \left\{ \frac{k}{(1-z)^{1/2}} - \frac{w^3}{\left(1-w^2 z\right)^{1/2}} - \frac{3N(z)}{2^{1/2}} \right\}$$

The iteration for $z$ is as follows,

$$z_{n+1} = z_n - \frac{N(z_n)}{N'(z_n)}$$

## Perturbed motion solutions of Lambert's problem

*Shooting method with state transition matrix updates*

An initial guess for this algorithm is created by first solving the two-body form of Lambert's problem. At each *shooting* iteration, the initial delta-velocity vector is updated according to

$$\Delta\mathbf{V} = \left[\Phi_{12}\right]^{-1} \Delta\mathbf{r}$$

where the error in the final position vector $\Delta\mathbf{r}$ is determined from the difference between the two-body final position vector $\mathbf{r}_{tb}$ and the final position vector predicted by numerical integration $\mathbf{r}_{\text{int}}$ of the orbital equations of motion as follows

$$\Delta\mathbf{r} = \mathbf{r}_{tb} - \mathbf{r}_{\text{int}}$$

The new initial velocity vector can now be calculated from

$$\mathbf{V}_{n+1} = \mathbf{V}_n + \Delta\mathbf{V}$$

The sub-matrix $\Phi_{12}$ of the full state transition matrix is as follows:

$$\Phi_{12} = \left[\frac{\partial\mathbf{r}}{\partial\mathbf{V}_0}\right] = \begin{bmatrix} \partial x/\partial\dot{x}_0 & \partial x/\partial\dot{y}_0 & \partial x/\partial\dot{z}_0 \\ \partial y/\partial\dot{x}_0 & \partial y/\partial\dot{y}_0 & \partial y/\partial\dot{z}_0 \\ \partial z/\partial\dot{x}_0 & \partial z/\partial\dot{y}_0 & \partial z/\partial\dot{z}_0 \end{bmatrix}$$

This sub-matrix consists of the partial derivatives of the rectangular cartesian components of the final position vector with respect to the initial velocity vector.

*Nonlinear programming solution of Lambert's problem*

In this classic trajectory optimization problem, the components of the initial and final delta-v vectors are the *control variables* and the scalar magnitude of the flyby or rendezvous $\Delta V$ is the *objective function* or *performance index*. The NLP implementation uses the two-body solution for Lambert's problem as its initial guess.

For the flyby problem, this method attempts to match all three components of the position vector. For the rendezvous problem, the NLP attempts to match all three components of both the target position and velocity vectors. These mission requirements are formulated as equality constraints.

"A Practical Note on the Use of Lambert's Equation" Geza Gedeon, *AIAA Journal*, Volume 3, Number 1, 1965, pages 149-150.

*An Introduction to the Mathematics and Methods of Astrodynamics*, Richard H. Battin, AIAA Education Series, 1987.

*Analytical Mechanics of Space Systems*, Hanspeter Schaub and John L. Junkins, AIAA Education Series, 2003.

*Orbital Mechanics*, Vladimir A. Chobotov, AIAA Education Series, 2002.

*Modern Astrodynamics*, Victor R. Bond and Mark C. Allman, Princeton University Press, 1996.

*Spacecraft Mission Design*, Charles D. Brown, AIAA Education Series, 1992.

# APPENDIX B

# Time Systems

This appendix is a brief explanation of the time systems used in this MATLAB script.

*Coordinated Universal Time, UTC*

Coordinated Universal Time (UTC) is the time scale available from broadcast time signals. It is a compromise between the highly stable atomic time and the irregular earth rotation. UTC is the international basis of civil and scientific time.

*Terrestrial Time, TT*

Terrestrial Time 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, by the relation **TT = TAI + 32$^s$.184.** It is the time scale now used for the precise calculation of future astronomical events observable from Earth.

TT = TAI + 32.184 seconds

TT = UTC + (number of leap seconds) + 32.184 seconds

*Barycentric Dynamical Time, TDB*

Barycentric Dynamical Time 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.

TDB = TT + periodic corrections

where typical periodic corrections (USNO Circular 179) are

$$\begin{aligned} TDB = TT &+ 0.001657\sin\left(628.3076T + 6.2401\right) \\ &+ 0.000022\sin\left(575.3385T + 4.2970\right) \\ &+ 0.000014\sin\left(1256.6152T + 6.1969\right) \\ &+ 0.000005\sin\left(606.9777T + 4.0212\right) \\ &+ 0.000005\sin\left(52.9691T + 0.4444\right) \\ &+ 0.000002\sin\left(21.3299T + 5.5431\right) \\ &+ 0.000010T\sin\left(628.3076T + 4.2490\right) + \cdots \end{aligned}$$

In this equation, the coefficients are in seconds, the angular arguments are in radians, and $T$ is the number of Julian centuries of $TT$ from J2000; $T$ = (Julian Date($TT$) – 2451545.0) / 36525.

*Leap seconds calculation*

The difference between International Atomic Time (TAI) and Universal Coordinated Time (UTC) is the number of current leap seconds. International Atomic Time (TAI**,** Temps Atomique International) is a physical time scale with the unit of the SI (System International) second and derived from a statistical timescale based on a number of atomic clocks. Coordinated Universal Time (UTC) is the time scale available from broadcast time signals. It is a compromise between the highly stable atomic time and the irregular earth rotation. UTC is the international basis of civil and scientific time.

The calculation of leap seconds in this MATLAB script is performed by a function that reads a simple ASCII data file and evaluates the current value of leap seconds. The leap second function must be initialized by including the following statements in the main script.

```
% read leap seconds data file

read_leap;
```

The `read_leap` MATLAB function reads the contents of the following simple comma-separated-variable (csv) two column data file. The name of this file is `tai-utc.dat`.

```
 2441317.5,  10.0
 2441499.5,  11.0
 2441683.5,  12.0
 2442048.5,  13.0
 2442413.5,  14.0
 2442778.5,  15.0
 2443144.5,  16.0
 2443509.5,  17.0
 2443874.5,  18.0
 2444239.5,  19.0
 2444786.5,  20.0
 2445151.5,  21.0
 2445516.5,  22.0
 2446247.5,  23.0
 2447161.5,  24.0
 2447892.5,  25.0
 2448257.5,  26.0
 2448804.5,  27.0
 2449169.5,  28.0
 2449534.5,  29.0
 2450083.5,  30.0
 2450630.5,  31.0
 2451179.5,  32.0
 2453736.5,  33.0
 2454832.5,  34.0
 2456109.5,  35.0
 2457204.5,  36.0
 2457754.5,  37.0
```

The first column of this data file is the Julian date, on the UTC time scale, at which the leap second became valid. The second column is the leap second value, in seconds.

Note that this data is passed between the leap second MATLAB functions by way of a global statement.

```
global jdateleap leapsec
```

The MATLAB function that reads and evaluates the current value of leap seconds has the following syntax and single argument.

```
function leapsecond = find_leap(jdate)

% find number of leap seconds for utc julian date

% input

%  jdate = utc julian date

% input via global

%  jdateleap = array of utc julian dates
%  leapsec   = array of leap seconds

% output

%  leapsecond = number of leap seconds
```

The leap seconds data file should be updated whenever the International Earth Rotation and Reference Systems Service (IERS) announces a new leap second.

The fundamental time argument for the JPL ephemeris function used in this MATLAB script is "ephemeris" time. As implemented here, we assume this time argument to be Barycentric Dynamic Time (TDB). Here's example MATLAB code for evaluating the state vector of the moon,

```
% ephemeris time at lunar encounter (seconds)

etime = 86400.0 * (jdtdb0 - 2451545.0) + 3600.0 * x(1) + tof;

% true-of-date geocentric state vector of the moon
% (kilometers and kilometers/second)

[rmoon, vmoon] = moon(etime);
```

and the source code that creates the lunar state vector.

```
function [rmoon, vmoon] = moon(etime)

% true-of-date geocentric state vector of the moon

% input

%  etime = ephemeris time (seconds)

% output

%  rmoon = position vector of the moon (kilometers)
%  vmoon = velocity vector of the moon (kilometers/second)

% Orbital Mechanics with MATLAB

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

starg = mice_spkezr('Moon', etime, 'Earth_tod', 'NONE', 'Earth');

posvel = reshape([starg(:).state], 6, []);

rmoon = posvel(1:3);

vmoon = posvel(4)
```

To report the time of important trajectory events in Universal Coordinated Time (UTC) or civil time, we need an algorithm to make this time conversion. The following is the MATLAB source code for a function which iteratively performs this calculation using Brent's root-finding method.

```
function jdutc = tdb2utc (jdtdb)

% convert TDB julian date to UTC julian date

% input

%  jdtdb = TDB julian date

% output

%  jdutc = UTC julian date

% Orbital Mechanics with MATLAB

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

global jdsaved

jdsaved = jdtdb;

% convergence tolerance

rtol = 1.0e-8;

% set lower and upper bounds

x1 = jdsaved - 0.1;

x2 = jdsaved + 0.1;

% solve for UTC julian date using Brent's method

[xroot, froot] = brent ('jdfunc', x1, x2, rtol);


jdutc = xroot;

end
```

This function calls the following MATLAB objective function during the calculations.

```
function fx = jdfunc(jdin)

% objective function for tdb2utc

% input

%  jdin = current value for UTC julian date

% output

%  fx = delta julian date

% Orbital Mechanics with MATLAB

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

global jdsaved

tai_utc = findleap(jdin);
```

```
fx = utc2tdb (jdin, tai_utc) - jdsaved;

end
```

Notice that this function requires the `find_leap` function which calculates the number of leap seconds for the current UTC Julian date value. The `jdfunc` function is computing the difference between the TDB Julian date input by the user and the value computed by the `utc2tdb` MATLAB function. The algorithm has converged whenever this value is less than or equal to the user-defined tolerance `rtol`.

```
function leapsecond = find_leap(jdate)

% find number of leap seconds for utc julian date

% input

%  jdate = utc julian date

% input via global

%  jdateleap = array of utc julian dates
%  leapsec   = array of leap seconds

% output

%  leapsecond = number of leap seconds

% Orbital Mechanics with MATLAB

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

global jdateleap leapsec

ndata = length(jdateleap);

if (jdate <= jdateleap(1))

    % date is <= 1972

    leapsecond = leapsec(1);

elseif (jdate >= jdateleap(ndata))

    % date is >= end of current data

    leapsecond = leapsec(ndata);
else

    % find data within table

   for i = 1:1:ndata - 1

       if (jdate >= jdateleap(i) && jdate < jdateleap(i + 1))
           leapsecond = leapsec(i);

           break;

       end

   end

end
```