

Computer Methods for Orbital Transfer Design, Analysis and Optimization

Written by C. David Eagle Jr.

cdeaglejr@yahoo.com

<http://celestialandorbitalmechanicswebsite.yolasite.com/>

Program leo2geo_ocs

Continuous Low-Thrust LEO-to-GEO Trajectory Optimization

This document is the user's manual for a Fortran computer program called `leo2geo_ocs` that uses the *Sparse Optimization Suite* distributed by [Applied Mathematical Analysis](#) to solve the continuous, single-maneuver, finite-burn low Earth orbit (LEO) to geosynchronous Earth orbit (GEO) orbit transfer optimization problem. The software attempts to maximize the final spacecraft mass. Since this simulation involves a single continuous propulsive maneuver, this is equivalent to minimizing the propellant mass required for the orbital maneuver.

The important features of this scientific simulation are as follows:

- single, continuous thrust transfer trajectory
- constant propulsive thrust magnitude
- modified equinoctial equations of motion
- near-circular initial and final orbits
- two types of initial guess algorithms
- numerical verification of the optimal control solution

The *Sparse Optimization 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 *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010 (www.siam.org).

The `leo2geo_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 *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `leo2geo_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:

```
leo2geo_ocs leo2geo_jk.in
```

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

```
*****
*      program leo2geo_ocs      *
*                               *
*      low-thrust LEO-to-GEO   *
*      trajectory optimization  *
*                               *
*      March 20, 2012          *
*****

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.

The screen output created by the `leo2geo_ocs` computer program can be re-directed to a text file with a command line similar to

```
leo2geo_ocs leo2geo_ij.in >leo2geo_jk.txt
```

To create a DOS command window in Windows 7, select **start**, then **All Programs**, then **Accessories** and finally **Command Prompt**. The size, font and other characteristics of the screen can be controlled by the user with the **c:**\ icon in the upper left corner of the window. To log into the subdirectory created during the installation of the Fortran executable and support files, type **root:**\ and then **cd subdirectory** from the DOS command line where **root** is the name of the root directory, usually **c:**, and **subdirectory** is the name of the subdirectory created by the user.

The DOS command line prompt looks similar to **C:\leo2geo_ocs>_.**

Input file format and contents

The `leo2geo_ocs` software is “data-driven” by a user-created text file. 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. In the following discussion the actual input file contents are in *courier* font and all explanations are in *times italic* font.

The following is a typical input file used by the `leo2geo_ocs` computer program. This is a classic LEO-to-GEO example taken from the paper, “Minimum-Time Low-Thrust Rendezvous and Transfer Using Epoch Mean Longitude Formulation”, Jean A. Kechichian, *Journal of Guidance, Control, and*

Dynamics, Vol. 22, No. 3, May-June 1999. For this example the initial true longitude is free. However, the optimal control solution includes the effect of propellant mass depletion due to thrusting while the example in Dr. Kechichian's technical paper assumes constant mass and therefore constant thrust acceleration. This example also includes the effect of the Earth's J_2 gravity term.

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.

```
*****
** leo-to-geo low-thrust trajectory optimization
** single phase continuous-thrust maneuver
** program leo2geo_ocs - leo-to-geo orbit transfer
** J. Kechichian example - leo2geo_jk.in
*****
```

The first three program inputs are the initial spacecraft mass, thrust magnitude and specific impulse. Please note the proper units for each data item.

```
initial spacecraft mass (kilograms)
1000.0

thrust magnitude (newtons)
98.0

specific impulse (seconds)
3300.0
```

The next six numerical inputs are the classical orbital elements of the initial orbit.

```
*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
7000.0

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
0.0

true anomaly (degrees)
0.0
```

The following input determines if the software will constrain or free the initial true longitude.

```
constrain initial true longitude (1 = yes, 2 = no)
2
```

The following five inputs are the user-defined classical orbital elements of the final mission orbit.

```
*****
* FINAL ORBIT *
*****
```



```

semimajor axis (kilometers)
42000.0

orbital eccentricity (non-dimensional)
0.001

orbital inclination (degrees)
1.0

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
0.0

```

The next integer input defines the type of final orbit point constraints. Please see the “Problem setup” section for additional information about this program item.

```

*****
* type of final orbit point constraints *
-----
1 = modified equinoctial orbital elements (semimajor axis, ecc = 0, inc = 0)
2 = eci components of final state vector (hx, hy, hz, |r|, sin(gamma))
3 = all components of final classical orbital elements
-----
2

```

The next program input tells the software what type of gravity model to use during the simulation. Option 2 will include the oblateness gravity coefficient (J_2) in the equations of motion.

```

*****
* type of gravity model *
-----
1 = spherical Earth
2 = j2 gravity model
-----
2

```

The next integer input defines the type of initial guess used to estimate the transfer or thrust duration time. Please see the “Creating an initial guess” section later in this document for additional information about this program item.

```

*****
* type of initial guess for transfer time *
-----
1 = numerical integration (coplanar orbits)
2 = Edelbaum algorithm (non-coplanar orbits)
3 = user-defined
-----
2

```

The following data item is the user’s initial guess for the transfer time, in hours. This input corresponds to option 3 of the previous item.

```

*****
* user-defined initial guess for transfer time (hours) *
*****
4

```

The next integer input defines the type of initial guess to use for the simulation. Please see the “Initial guess” technical discussion for additional information about this program item.

```

*****
* initial guess options*
*****
1 = linear guess with tangential thrusting
2 = numerical integration with Edelbaum steering
3 = binary data file
-----
2

```

If the user elects to use a binary data file (option 3 above) for the initial guess, the following text input specifies the name of the file to use.

```

name of binary initial guess data file
leo2geo_jk.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 3, an input of yes to this item will update the binary file used to initialize the simulation.

```

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

create/update binary 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 the Sparse Optimization Suite software. Options 2 and 3 allow the user to specify either the number of nodes or time step size of the data file.

```

*****
* type of comma-delimited solution data file *
*****
1 = OCS-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
50

```

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
leo2geo_jk.csv

```

The next series of program inputs are algorithm control options and parameters for the Sparse Optimization Suite. The first input is an integer that specifies the type of collocation method to use during the solution process.

```

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

```

```

discretization/collocation method
-----
1 = trapezoidal
2 = separated Hermite-Simpson
3 = compressed Hermite-Simpson
-----
1

```

The next input is an integer that defines the number of grid points to use for the initial guess.

```

number of grid points
100

```

The next input defines the relative error in the objective function.

```

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.

```

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
500000

```

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

```

maximum number of algorithm iterations
1000

```

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

```

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

```

The level of output from the Sparse Optimization Suite 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 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 Sparse Optimization Suite 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 configuration input files. The user should always create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the leo2geo_ocs software. Please consult Appendix B for additional information about this program option.

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

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

name of optimal control configuration file
leo2geo_config.txt

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

name of optimal control configuration file
leo2geo_config1.txt
```

Optimal control solution and graphics

The following is the program output for this example. It includes the orbital elements of the initial LEO and the final GEO mission orbit. It also summarizes the total thrust duration, final spacecraft mass, propellant mass required for the orbit transfer and the accumulated delta-v.

The leo2geo_ocs software will also create a comma-separated-variable (csv) output file. This file contains the state vector, orbital elements and steering angles during the transfer trajectory. It has the file name specified by the user in the simulation definition (input) data file. Please consult Appendix A for additional information about the contents of this file.

```

=====
program leo2geo_ocs
=====

transfer time algorithm
-----
Edelbaum algorithm (non-coplanar orbits)

initial guess type
-----
linear guess with tangential thrusting

gravity model type
-----
j2 earth gravity model

-----
beginning of finite burn
-----

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.700000000000D+04  0.615067684519D-16  0.285000000000D+02  0.000000000000D+00

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.206748308333D-14  0.285866456178D+03  0.285866456178D+03  0.971419368695D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
0.191377284137D+04  -.591734870255D+04  -.321285820480D+04  0.700000000000D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.725856076669D+01  0.181305405204D+01  0.984408031306D+00  0.754605384101D+01

-----
end of finite burn
-----

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.420000000000D+05  0.999999999999D-03  0.100000000000D+01  0.360000000000D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.000000000000D+00  0.423922295262D+02  0.423922295262D+02  0.142768906774D+04

      rx (km)      ry (km)      rz (km)      rmag (km)
0.309960418384D+05  0.282912582669D+05  0.493825749949D+03  0.419689619582D+05

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-.207699129982D+01  0.227794898386D+01  0.397617474165D-01  0.308294103563D+01

transfer time      15.0823889600487      hours
spacecraft mass    835.576420850925      kilograms
propellant mass    164.423579149075      kilograms
delta-v            5813.28840908474      meters/second

```

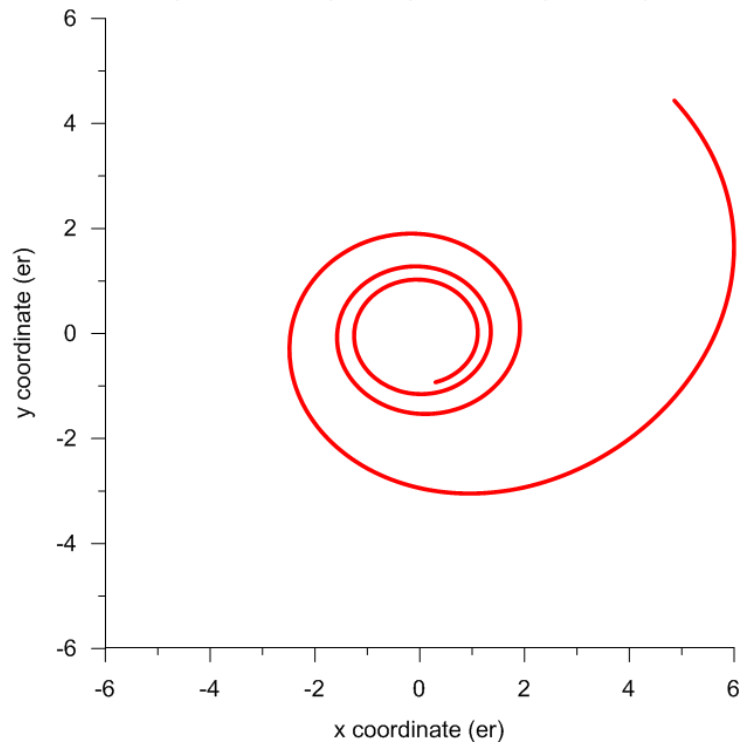
The following are typical transfer trajectory, optimal control and orbital element plots for this example. They were created with the Grapher plotting program (www.goldensoftware.com).

The first plot is a view of the transfer trajectory from a north pole viewpoint looking down on the equatorial plane. The unit of each trajectory coordinate is Earth radii.

Continuous, Low-Thrust LEO-to-GEO Orbit Transfer

Mass = 1000 kg Thrust = 98 nt Isp = 3300 s

(transfer trajectory in the x-y plane)

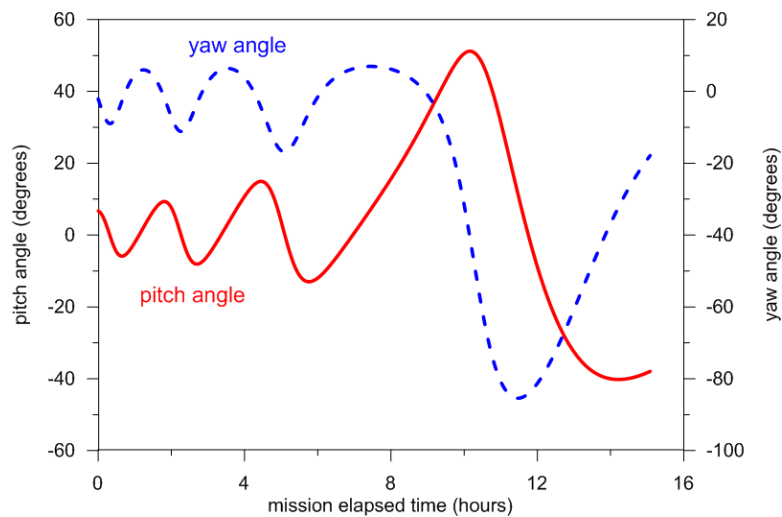


This next plot illustrates the behavior of the pitch and yaw steering angles during the transfer.

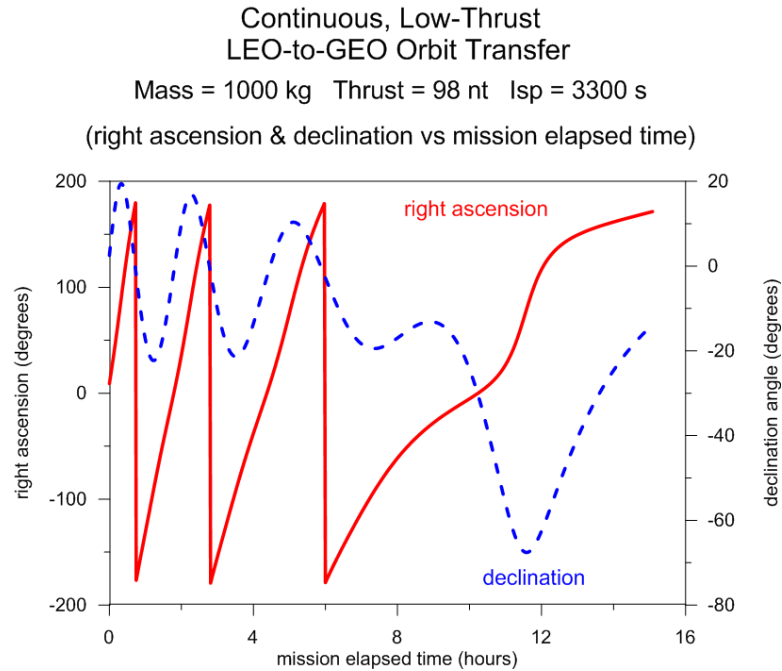
Continuous, Low-Thrust LEO-to-GEO Orbit Transfer

Mass = 1000 kg Thrust = 98 nt Isp = 3300 s

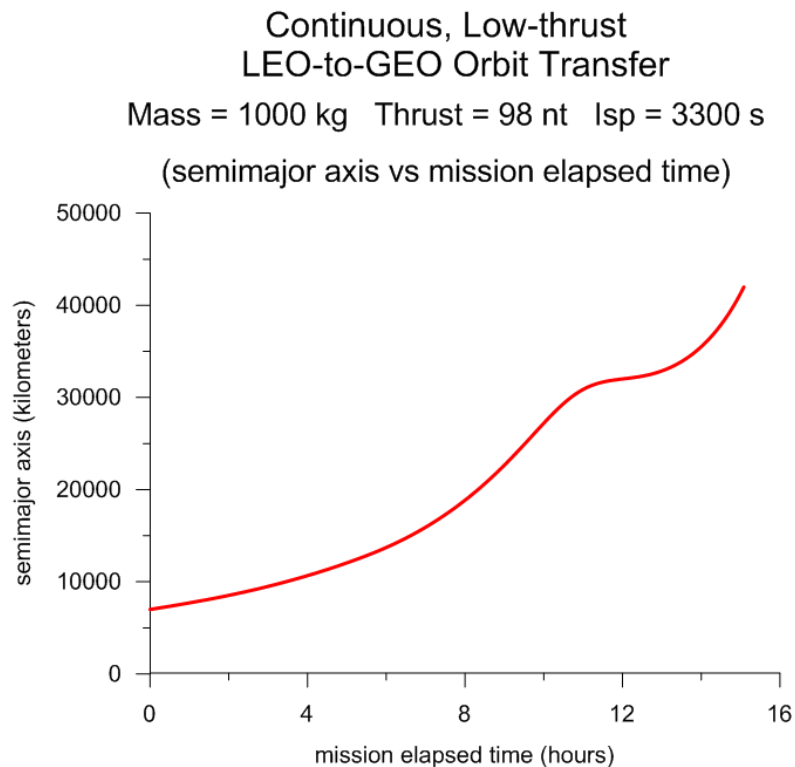
(pitch & yaw angles vs mission elapsed time)



This plot summarizes the inertial right ascension and declination angles for this maneuver.



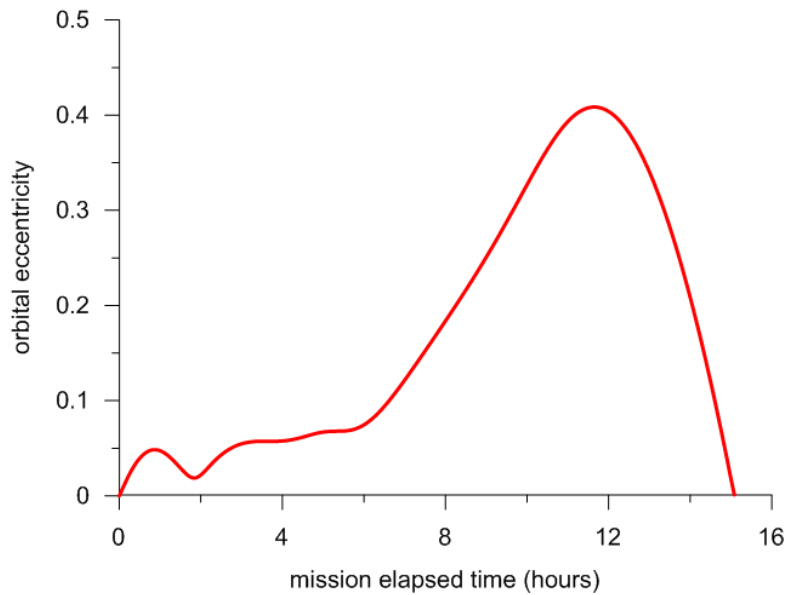
The next three plots illustrate the behavior of the semimajor axis, orbital eccentricity and orbital inclination of the transfer orbit during the continuous low-thrust maneuver.



Continuous, Low-thrust LEO-to-GEO Orbit Transfer

Mass = 1000 kg Thrust = 98 nt Isp = 3300 s

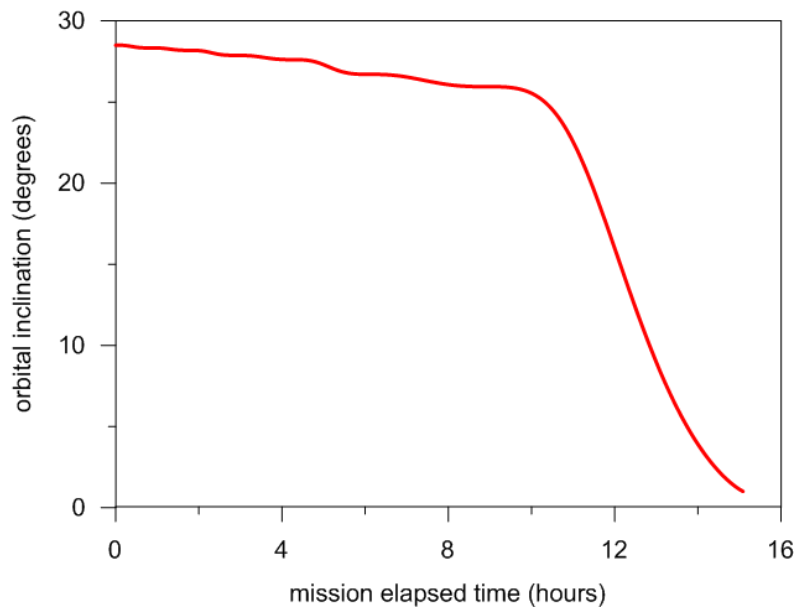
(eccentricity vs mission elapsed time)



Continuous, Low-thrust LEO-to-GEO Orbit Transfer

Mass = 1000 kg Thrust = 98 nt Isp = 3300 s

(inclination vs mission elapsed time)



Verification of the optimal control solution

The optimal control solution determined by the software can be verified by numerically integrating the orbital equations of motion with the optimal control-computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution. This part of the `leo2geo_ocs` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is a summary of the final optimal solution computed using this *explicit* numerical integration method.

```
=====
verification of optimal control solution
=====

mission orbit state vector and orbital elements
-----

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.419999999440D+05  0.100000252738D-02  0.999999595528D+00  0.359999478450D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.359999985471D+03  0.423927660632D+02  0.423922445128D+02  0.142768906488D+04

      rx (km)      ry (km)      rz (km)      rmag (km)
0.309960417074D+05  0.282912586047D+05  0.493825693260D+03  0.419689620884D+05

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-.207699129066D+01  0.227794897693D+01  0.397617220168D-01  0.308294102401D+01

transfer time      15.0823889600487      hours
spacecraft mass    835.576420851553      kilograms
propellant mass    164.423579148447      kilograms
delta-v            5813.28840906505      meters/second
```

Creating an initial guess

The solution to this classic orbit transfer problem requires a good initial guess for the transfer time along with a reasonable guess for the dynamic variables during the orbit transfer.

Transfer time initial guess

The `leo2geo_ocs` software implements three options for specifying the transfer time initial guess. The first technique simply integrates the *coplanar* transfer orbit trajectory using tangential thrusting until the final orbit radius is reached. The second method uses Edelbaum's algorithm. The first method is best for coplanar orbit transfers and the second method should be used for non-coplanar orbit transfers. The third initial guess option is a user-specified transfer time.

For the first technique, the unit thrust vector for tangential steering during the numerical integration is simply $\mathbf{u}_T = (0 \ 1 \ 0)^T$.

The Edelbaum algorithm is described in Chapter 14 of the book *Orbital Mechanics* by V. Chobotov and the technical paper, “The Reformulation of Edelbaum's Low-thrust Transfer Problem Using Optimal Control Theory” by J. A. Kechichian, AIAA-92-4576-CP. The original Edelbaum algorithm is described in “Propulsion Requirements for Controllable Satellites”, ARS Journal, Aug. 1961, pp. 1079-1089. This algorithm is valid for total inclination changes Δi given by $0 < \Delta i < 114.6^\circ$ and assumes that the thrust acceleration magnitude and spacecraft mass are both constant during the orbit transfer.

The initial thrust vector yaw angle β_0 is given by the following expression

$$\tan \beta_0 = \frac{\sin\left(\frac{\pi}{2} \Delta i\right)}{\frac{V_0}{V_f} - \cos\left(\frac{\pi}{2} \Delta i\right)}$$

where the speed on the initial circular orbit is $V_0 = \sqrt{\mu/r_0}$ and the speed on the final circular orbit is $V_f = \sqrt{\mu/r_f}$. In these equations $r_0 = r_e + h_0$ is the geocentric radius of the initial orbit, $r_f = r_e + h_f$ is the geocentric radius of the final orbit, r_e is the radius of the Earth and μ is the gravitational constant of the Earth. The initial altitude is h_0 and the final altitude is h_f .

The total velocity change required for a low-thrust orbit transfer is given by

$$\Delta V = V_0 \cos \beta_0 - \frac{V_0 \sin \beta_0}{\tan\left(\frac{\pi}{2} \Delta i + \beta_0\right)}$$

The total transfer time is given by $t = \Delta V / f$ where f is the thrust acceleration. This is the transfer time used for the Edelbaum guess option in the `leo2geo_ocs` software.

Dynamic variables initial guess

The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the program to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

Binary restart file initial guess

Binary restart data files can also be used to initialize a `leo2geo_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 multiple simulations to eventually reach the final desired solution.

Problem setup

This part of the user's manual provides details about the software implementation within the `leo2geo_ocs` computer program. It defines such things as point and path constraints (boundary conditions), bounds on the dynamic variables, and the performance index or objective function.

(1) Performance index – maximize final spacecraft mass

The objective function or performance index J for this simulation is the mass of the spacecraft when it reaches the final mission orbit. This is simply

$$J = m_f$$

The value of the `maxmin` indicator in the software tells the program whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the initial value provided by the user.

(2) Path constraint – unit thrust vector scalar magnitude

At any point during the transfer trajectory, the scalar magnitude of the components of the unit thrust vector is constrained as follows:

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

(3) Initial true longitude

The software allows the user to either fix or free the initial true longitude. For an unconstrained initial true longitude, the true longitude bounds are $0 \leq L \leq 2\pi$. Otherwise, the initial true longitude is fixed to the value $L = (\Omega + \omega + \theta)_i$.

(4) Mission constraint “matching” at the final orbit

The `leo2geo_ocs` software implements three techniques for “targeting” the final mission orbit. The correct option to use depends on the characteristics of the final mission orbit.

For final orbits that are circular and equatorial, the set of final constraints are specified in terms of the final modified equinoctial elements or dynamic variables according to

$$p = p_f$$

$$f_f = g_f = h_f = k_f = 0$$

The f subscript indicates values on the user-specified final orbit. This set of constraints or boundary conditions follows from the orbital element definitions

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

These boundary conditions are enforced using lower and upper bounds on the dynamic variables at the final time.

For final orbits that may be near-circular and/or inclined, the final mission constraints are enforced using *point functions*. This set of point functions is given by

$$\begin{aligned} h_x &= h_{x_f} & h_y &= h_{y_f} & h_z &= h_{z_f} \\ r &= r_f \\ \sin \gamma &= \sin \gamma_f \end{aligned}$$

where h_x, h_y, h_z are the inertial components of the angular momentum vector, r is the geocentric radius and γ is the flight path angle. As noted previously, the f subscript indicates mission values on the user-specified final orbit.

A third program option allows the user to constrain a classical orbital element set consisting of the semimajor axis, orbital eccentricity and inclination using point functions. This set of mission constraints is

$$a = a_f \quad e = e_f \quad i = i_f$$

As before the f subscript indicates values on the user-specified final orbit.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables during the orbital transfer.

$$\begin{aligned} 0.05m_{sc_i} &\leq m_{sc} \leq 1.05m_{sc_i} \\ 1.1p_f &\leq p \leq 0.9p_i \\ -1 &\leq f \leq +1 & -1 &\leq g \leq +1 \\ -1 &\leq h \leq +1 & -1 &\leq k \leq +1 \\ 0 &\leq L \leq 200 \cdot 2\pi \end{aligned}$$

where m_{sc_i} is the initial spacecraft mass, and p_i is the semiparameter of the initial orbit and p_f is the semiparameter of the final orbit. The upper bound on true longitude L allows for a maximum of 200 complete orbits during the transfer.

Finally, the components of the unit thrust vector are constrained as follows:

$$-1.1 \leq u_r \leq +1.1$$

$$-1.1 \leq u_t \leq +1.1$$

$$-1.1 \leq u_n \leq +1.1$$

Technical discussion

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

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

The relationship between classical and modified equinoctial orbital elements is:

$$\text{semimajor axis} \quad a = \frac{p}{1 - f^2 - g^2}$$

$$\text{orbital eccentricity} \quad e = \sqrt{f^2 + g^2}$$

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

argument of periapsis

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

right ascension of the ascending node

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

true anomaly

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

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

position vector

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

velocity vector

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

where

$$\alpha^2 = h^2 - k^2 \quad s^2 = 1 + h^2 + k^2$$

$$r = \frac{p}{w} \quad w = 1 + f \cos L + g \sin L$$

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

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

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

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

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

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

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

where $\Delta_r, \Delta_t, \Delta_n$ are *non-two-body* perturbations in the radial, tangential and normal directions, respectively. 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 in the direction of the angular momentum vector of the spacecraft's orbit.

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

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

where

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

and

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

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

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

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

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

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.

Propulsive thrust

The acceleration due to propulsive thrust can be expressed as

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

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

The propellant mass flow rate is determined from

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

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

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:

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

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

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

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

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

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

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

where

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

This relationship can also be expressed as

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

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

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

For the case of tangential steering

$$\hat{\mathbf{u}}_{T_{ECI}} = \begin{bmatrix} (\hat{\mathbf{h}} \times \hat{\mathbf{r}})_x & (\hat{\mathbf{h}} \times \hat{\mathbf{r}})_y & (\hat{\mathbf{h}} \times \hat{\mathbf{r}})_z \end{bmatrix}^T$$

In the `leo2geo_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

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

where the calculation for right ascension is a four quadrant inverse tangent operation.

Gravitational acceleration

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 a *zonal* gravity model of order n are as follows:

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

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

where

μ = gravitational constant

r = geocentric distance of the spacecraft

R_e = equatorial radius of the Earth

ϕ = geocentric latitude

J_k = zonal gravity coefficient

P_k = k^{th} order Legendre polynomial

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

Finally, the zonal gravity perturbation contribution is

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

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

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

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

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

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

These are the equations implemented in this computer program.

Algorithm resources

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.

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

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

“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.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“Optimal Interplanetary Orbit Transfers by Direct Transcription”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

“Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, 20-22 August 1990.

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 the CSV data files produced by the `leo2geo_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. 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 spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

transfer time = total thrust duration in hours

final mass = final spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the accumulated delta-v in meters/seconds

The delta-v is determined using a cubic spline integration of the thrust acceleration data at each collocation node.

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

time (hrs) = simulation time since ignition in hours

time (days) = simulation time since ignition in days
semimajor axis (km) = semimajor axis in kilometers
eccentricity = orbital eccentricity (non-dimensional)
inclination (deg) = orbital inclination in degrees
argument of perigee (deg) = argument of perigee in degrees
raan (deg) = right ascension of the ascending node in degrees
true anomaly = true anomaly in degrees
period (min) = orbital period in minutes
mass (kg) = spacecraft mass in kilograms
T/W = thrust-to-weight ratio
yaw = thrust vector yaw angle in degrees
pitch = thrust vector pitch angle in degrees
perigee altitude = perigee altitude in kilometers
apogee altitude = apogee altitude in kilometers
ut-radial = radial component of unit thrust vector
ut-tangential = tangential component of unit thrust vector
ut-normal = normal component of unit thrust vector
semi-parameter = orbital semiparameter in kilometers
f equinoctial element = modified equinoctial orbital element
g equinoctial element = modified equinoctial orbital element
h equinoctial element = modified equinoctial orbital element
k equinoctial element = modified equinoctial orbital element
true longitude = true longitude in degrees
rx (er) = x-component of the spacecraft's position vector in Earth radii
ry (er) = y-component of the spacecraft's position vector in Earth radii
rz (er) = z-component of the spacecraft's position vector in Earth radii
fpa (deg) = flight path angle in degrees
dvacc (mps) = accumulative delta-v in meters per second

APPENDIX B

Typical Sparse Optimization Suite Configuration File

The `leo2geo_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 **INOCS** routine in section 6.2, *Subprograms for Optimal Control*, and the **INSNLP** routine in Section 2.2, *Subprograms for Optimization* of the *Sparse Optimization Suite* user's manual. Please note that the `leo2geo_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 `leo2geo_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
TOLMSA=0.1490116119384766D-07
TOLMR=0.1490116119384766D-07
IDSCPH=0
IDSCND=0
IDSCVR=0
IDSCFN=0
IDTSFD=-1
IPFAUX=0
IPFSFD=0
IPRSFD=1
IPGRD=0
IPNLP=10
IODE=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=hyper1.rsbin
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:RHO1WR=0.100000000000000D+03
INSNLP:IMAXMU=10
INSNLP:MUCALC=3
INSNLP:MXQPIT=1

Program aeroassist_ocs

Aero-assist Trajectory Optimization

This document is the user's manual for a Fortran computer program called `aeroassist_ocs` that uses the *Sparse Optimization Suite* distributed by [Applied Mathematical Analysis](#) to solve the aero-assist trajectory optimization problem. The trajectory is modeled as a single phase with several types of user-defined initial and final boundary conditions. The software attempts to maximize the speed of the vehicle at atmospheric exit or maximize the orbital plane change during the atmospheric phase of the mission. The type of optimization is selected by the user.

The important features of this scientific simulation are as follows:

- Normalized lift coefficient and bank angle control variables
- 3-DOF flight path equations of motion relative to a spherical, rotating Earth
- U.S. Standard 1976 atmosphere model and fourth-order zonal gravity model
- User-defined aerodynamic characteristics and point-mass vehicle properties
- User-defined path constraints such as heat rate and dynamic pressure

The *Sparse Optimization Suite* 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 *Sparse Optimization Suite* software can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `aeroassist_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 *Sparse Optimization Suite* software will use this information to *automatically* transcribe the user's problem and perform the optimization using a sparse nonlinear programming method. The software allows the user to select the type of 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:

```
Aeroassist_ocs geo2leo_max_speed.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 aeroassist_ocs      *
*                                  *
*      aeroassist trajectory        *
*      optimization                 *
*                                  *
*      June 9, 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 `aeroassist_ocs` software is “data-driven” by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in *times italic* font.

This data file defines a simulation that maximizes the speed at atmospheric exit while enforcing a heat rate path constraint. The simulation starts in a geosynchronous equatorial orbit (GEO) and finishes in an elliptical low Earth orbit (LEO) with a final orbital inclination of 28.5 degrees. The heat rate path constraint enforces a value ≤ 600 BTU/foot²-second.

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.

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.

```
*****
aeroassist trajectory optimization
simulation definition data file ==> geo2leo_max_speed.in
geo-to-leo w/ plane change and aeroheating constraints
maximize speed at atmospheric exit - June 9, 2011
*****
```

The first program input is an integer that defines the type of entry interface conditions to use. Please consult the “Problem setup” section later in this document for an explanation of these three program options.

```
*****
initial conditions type
*****
1 = user input of flight path coordinates at entry interface
2 = derived from deorbit maneuver; fixed entry conditions
3 = derived from deorbit maneuver; bounded entry conditions
-----
3
```

The next program input is an integer that defines the type of mission to simulate. Please note that program option 0 (no optimization) will solve the two-point boundary value problem (TPBVP).

```
*****
simulation type
*****
0 = no optimization
1 = maximize final speed
2 = maximize inclination change
-----
1
```

This section allows the user to define the vehicle weight, aerodynamic reference area, the nose radius used in the aero-heating calculations, and other vehicle aerodynamic properties.

```
*****
vehicle weight and aerodynamics characteristics
*****

vehicle weight (pounds)
5000.0

aerodynamic reference area (square feet)
125.84

nose radius (feet)
1.0

drag coefficient at zero angle-of-attack; cd0 (nondimensional)
0.032

drag polar constant k (non-dimensional)
1.4
```

The next set of inputs defines the conditions at the deorbit point in the initial circular orbit. The calendar date and universal time are required in order to transform coordinates.

```
*****
deorbit conditions
*****

calendar date at deorbit maneuver (month, day, year)
1,1,2001

universal time at deorbit maneuver (hours, minutes, seconds)
0,0,0

altitude at deorbit maneuver (nautical miles)
19323.0
```

orbital inclination at deorbit maneuver (degrees)
0.0

right ascension of the ascending node at deorbit maneuver (degrees)
0.0

true anomaly at deorbit maneuver (degrees)
30.0

The following series of data items are reserved for the initial conditions at the entry interface (EI) or point of atmospheric entry. To constrain one or more initial conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more initial states, set the lower and/or upper bounds to 1.0d99. Please note the units and valid data range for each item.

flight conditions and bounds at atmospheric entry

NOTE 1: set upper and lower bounds to
the initial value to constrain or
"fix" a flight condition.

NOTE 2: set bound to 1.0d99 to ignore

calendar date at entry interface (month, day, year)
1,1,2001

universal time at entry interface (hours, minutes, seconds)
0,0,0

initial time (seconds)
0.0

upper bound for initial time (seconds)
0.0

lower bound for initial time (seconds)
0.0

initial altitude (feet)
400000.0

upper bound for initial altitude (feet)
400000.0

lower bound for initial altitude (feet)
400000.0

initial velocity (feet/second)
32268.637

upper bound for initial velocity (feet/second)
35000.0

lower bound for initial velocity (feet/second)
20000.0

initial flight path angle (-90 <= fpa <= +90; degrees)
-4.0

upper bound for initial flight path angle (-90 <= fpa <= +90; degrees)
-0.5

```

lower bound for initial flight path angle (-90 <= fpa <= +90; degrees)
-10.0

initial flight azimuth (0 <= azimuth <= 360; degrees)
94.0

upper bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

initial declination (-90 <= declination <= +90; degrees)
0.0

upper bound for initial declination (-90 <= declination <= +90; degrees)
1.0d99

lower bound for initial declination (-90 <= declination <= +90; degrees)
1.0d99

initial east longitude (0 <= longitude <= 360; degrees)
18.0

upper bound for initial east longitude (0 <= longitude <= 360; degrees)
0.0

lower bound for initial east longitude (0 <= longitude <= 360; degrees)
0.0

```

The following series of data items allow the user to define the flight conditions at atmospheric exit. To constrain one or more conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more final states, set the lower and/or upper bounds to 1.0d99. Please note the units and valid data range for each item.

```

*****
flight conditions and bounds at atmospheric exit
*****

```

```

NOTE 1: set upper and lower bounds
to the final value to constrain or
"fix" a flight condition.
NOTE 2: set bound to 1.0d99 to ignore
-----

```

```

final time (seconds)
750.0

upper bound for final time (seconds)
10000.0

lower bound for final time (seconds)
100.0

final altitude (feet)
400000.0

upper bound for final altitude (feet)
400000.0

```

lower bound for final altitude (feet)
400000.0

final velocity (feet/second)
24314.43

upper bound for final velocity (feet/second)
35000.0

lower bound for final velocity (feet/second)
1000.0

final flight path angle (-90 <= fpa <= +90; degrees)
1.25

upper bound for final flight path angle (-90 <= fpa <= +90; degrees)
+20.0

lower bound for final flight path angle (-90 <= fpa <= +90; degrees)
-20.0

final flight azimuth (0 <= azimuth <= 360; degrees)
90.0

upper bound for final flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for final flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

final declination (-90 <= declination <= +90; degrees)
+10.0

upper bound for final declination (-90 <= declination <= +90; degrees)
1.0d99

lower bound for final declination (-90 <= declination <= +90; degrees)
1.0d99

final east longitude (0 <= longitude <= 360; degrees)
30.0

upper bound for final east longitude (0 <= longitude <= 360; degrees)
1.0d99

lower bound for final east longitude (0 <= longitude <= 360; degrees)
1.0d99

The next series of data inputs define lower and upper bounds on the state variables during the aero-assist phase. To free or un-constrain one or more states, set the lower and/or upper bounds to 1.0d99.

upper and lower bounds on the flight conditions during phase

NOTE: set bound to 1.0d99 to ignore

upper bound for altitude (feet)
450000.0d0

```

lower bound for altitude (feet)
10000.0

upper bound for velocity (feet/second)
35000.0d0

lower bound for velocity (feet/second)
1000.0

upper bound for flight path angle (-90 <= fpa <= +90; degrees)
+20.0

lower bound for flight path angle (-90 <= fpa <= +90; degrees)
-20.0

upper bound for flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

upper bound for declination (-90 <= declination <= +90; degrees)
1.0d99

lower bound for declination (-90 <= declination <= +90; degrees)
1.0d99

upper bound for east longitude (0 <= longitude <= 360; degrees)
1.0d99

lower bound for east longitude (0 <= longitude <= 360; degrees)
1.0d99

```

This section of the input file defines the initial guesses and bounds for the control variables. To free or un-constrain one or more control variables, set the lower and/or upper bounds to 1.0d99.

```

*****
initial flight controls and bounds
*****
NOTE 1: set upper and lower bounds
to the initial value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
-----

initial normalized lift coefficient
0.5

upper bound for initial normalized lift coefficient
+2.0

lower bound for initial normalized lift coefficient
+0.0d0

initial bank angle (degrees)
-90.0d0

upper bound for initial bank angle (degrees)
+0.0d0

```

```
lower bound for initial bank angle (degrees)
-180.0d0
```

This section of the input file defines the final guesses and bounds for the control variables. To free one or more control variables, set the lower and/or upper bounds to 1.0d99.

```
*****
final flight controls and bounds
*****
NOTE 1: set upper and lower bounds
to the final value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
-----

final normalized lift coefficient
0.5

upper bound for final normalized lift coefficient
+2.0

lower bound for final normalized lift coefficient
+0.0

final bank angle (degrees)
-90.0d0

upper bound for final bank angle (degrees)
+0.0d0

lower bound for final bank angle (degrees)
-180.0d0
```

This section of the input file defines the bounds for the control variables during the aero-assist phase. To free or un-constrain one or more control variables, set the lower and/or upper bounds to 1.0d99.

```
*****
upper and lower bounds on the flight controls during phase
*****
NOTE: set bound to 1.0d99 to ignore
-----

upper bound for normalized lift coefficient
+2.0d0

lower bound for normalized lift coefficient
+0.0d0

upper bound for bank angle (degrees)
+0.0d0

lower bound for bank angle (degrees)
-180.0d0
```

This next section allows the user to define and enforce one or more point and path constraints during the atmospheric phase. Path constraints are enforced at all points along the trajectory and point constraints are enforced at atmospheric exit only. The user should be careful not to enforce constraints that are inconsistent with either the initial and/or final boundary conditions. For example, while maximizing the final orbital inclination do not enforce an orbital inclination point constraint.


```
*****
flight constraints
*****
```

```
enforce an orbital inclination constraint (yes or no)
yes
```

```
orbital inclination constraint value (degrees)
28.5d0
```

```
enforce a heating rate constraint (yes or no)
yes
```

```
heating rate upper bound constraint value (BTU/foot**2-second)
600.0d0
```

```
enforce a dynamic pressure constraint (yes or no)
no
```

```
dynamic pressure upper bound constraint value (pounds per square foot)
700.0d0
```

The type of trajectory initial guess or restart is specified by the next integer input. Program option 1 will use a simple linear initial guess created from the initial and final values provided by the user. Option 2 will read and use a binary file to initialize the simulation. Be sure to create a binary file first.

```
*****
initial guess/restart option
*****
  1 = linear guess
  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 binary restart file
geo2leo_max_speed.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 options 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 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 the Sparse Optimization Suite. Options 2 and 3 allow the user to specify either the number of nodes (option 2) or time step size of the data file (option 3).

```

*****
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 (option 2) or the time step size of the data output in the solution file (option 3).

```

number of user-defined nodes or print step size in solution data file
10.0

```

The software also creates a comma-separated-variable (csv) ASCII data file that contains the optimal control solution and many other flight parameters. The name of this output file is specified in the next line of information. Please consult Appendix A for additional information about the contents of this file.

```

name of solution output file
geo2leo_max_speed.csv

```

The next series of program inputs are algorithm control options and parameters for the Sparse Optimization Suite. 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.

```

*****
discretization/collocation method
*****
1 = trapezoidal
2 = separated Hermite-Simpson
3 = compressed Hermite-Simpson
-----
1

```

This input defines the relative error in the objective function.

```

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.

```

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
100000

```

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

```

maximum number of algorithm iterations
10000

```

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

```

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

```

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

```

*****
user-defined output
-----
input no to ignore
*****
a0b0c0d0e0f0g0h0i0j1k0l0m0n0o0p0q0r0

```

The last series of inputs allow the reading and writing of 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 aeroassist_ocs software. Please consult Appendix F.

```

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

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

```

```

name of optimal control configuration file
aeroassist_config1.txt

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

name of optimal control configuration file
aeroassist_config1.txt

```

Atmosphere model and constants data file

The `aeroassist_ocs` software also requires a user-created ASCII data file with the name `tutility.dat` that defines the fundamental constants that will be used during the simulation. The following is the companion data file for this example. If the user provides a value of zero for the Earth's rotation rate, the simulation results will be with respect to a non-rotating, spherical Earth.

```

*****
simulation environment and
planet/utility constants
*****

radius of the earth (feet)
20925656.8d0

gravitational constant of the earth (feet**3/second**2)
1.407644381252d16

surface gravity (feet/second**2)
32.174d0

earth rotation rate (radians/second)
7.2921151467d-5

atmospheric surface density (slugs/feet**3)
0.0023765d0

j2 gravity coefficient (non-dimensional)
1.08262668355d-3

j3 gravity coefficient (non-dimensional)
0.0d0

j4 gravity coefficient (non-dimensional)
0.0d0

```

The user's input for `j2`, `j3` and `j4` control the type of gravity model used during the simulation.

Optimal control solution and graphics

After the `aeroassist_ocs` scientific simulation has converged, it will display a complete summary of the initial conditions and the optimized trajectory. It also provides a summary of the relative flight path coordinates and the aerodynamic characteristics of the vehicle. The classical orbital elements at atmospheric exit are determined from the inertial state vector which is computed using the relative flight path coordinates at exit.

The following is a summary of the solution for this example.

program aeroassist_ocs
=====

input file ==> geo2leo_max_speed.in

bounded entry conditions derived from deorbit maneuver

maximize speed at atmospheric exit

orbital elements and state vector prior to deorbit impulse

calendar date January 1, 2001

universal time 00:00:00.000

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.227669201902D+05	0.111022302463D-15	0.000000000000D+00	0.000000000000D+00
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.000000000000D+00	0.300000000000D+02	0.300000000000D+02	0.143607658003D+04
r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.227669201902D+05	0.193230000000D+05	0.227669201902D+05	0.193230000000D+05
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
0.119801136085D+09	0.691672181680D+08	0.000000000000D+00	0.138334436336D+09
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.504372416457D+04	0.873598651240D+04	0.000000000000D+00	0.100874483291D+05

orbital elements and state vector after deorbit impulse

calendar date January 1, 2001

universal time 00:00:00.000

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.131213107012D+05	0.735110211827D+00	0.000000000000D+00	0.210000000000D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.000000000000D+00	0.180000000000D+03	0.300000000000D+02	0.628328849946D+03
r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.347570121219D+04	0.317810220239D+02	0.227669201902D+05	0.193230000000D+05
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
0.119801136085D+09	0.691672181680D+08	0.000000000000D+00	0.138334436336D+09
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.259587595393D+04	0.449618904236D+04	0.000000000000D+00	0.519175190787D+04

flight path coordinates at atmospheric entry

altitude	400000.000000000	feet
velocity	32268.5194754447	feet/second
declination	2.637383834618052E-013	degrees
longitude	18.8080427045337	degrees
azimuth	90.0000000000005	degrees
flight path angle	-5.45162080704044	degrees

inertial fpa -5.20130119814137 degrees

orbital elements and state vector at atmospheric entry

calendar date January 1, 2001

universal time 05:11:55.842

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.131213104261D+05	0.735110206972D+00	0.581675704578D-12	0.210000003999D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.000000000000D+00	0.347714838486D+03	0.197714842485D+03	0.628328830184D+03
r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.347570120302D+04	0.317810128543D+02	0.227669196491D+05	0.193229994589D+05
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
-.203144515089D+08	-.648896739412D+07	0.981641981048D-07	0.213256568000D+08
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
0.131677427163D+05	-.311479253279D+05	-.318848277882D-09	0.338168996284D+05

flight path coordinates at atmospheric exit

altitude	400000.000000000	feet
velocity	25600.1592225824	feet/second
declination	11.8749410000429	degrees
longitude	59.7589681113454	degrees
azimuth	62.4006932817757	degrees
flight path angle	2.85290494398532	degrees

orbital elements and state vector at atmospheric exit

calendar date January 1, 2001

universal time 05:21:20.255

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.390347950906D+04	0.111289864741D+00	0.285000000002D+02	0.357704478388D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.218238520146D+03	0.278426340657D+02	0.255471124534D+02	0.101952769539D+03
r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.346906180248D+04	0.251416123121D+02	0.433789721564D+04	0.893977025480D+03
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
-.101099984601D+08	-.182568968034D+08	0.438831268249D+07	0.213256568000D+08
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
0.217306749776D+05	-.106726700692D+05	0.118541683058D+05	0.269564357364D+05

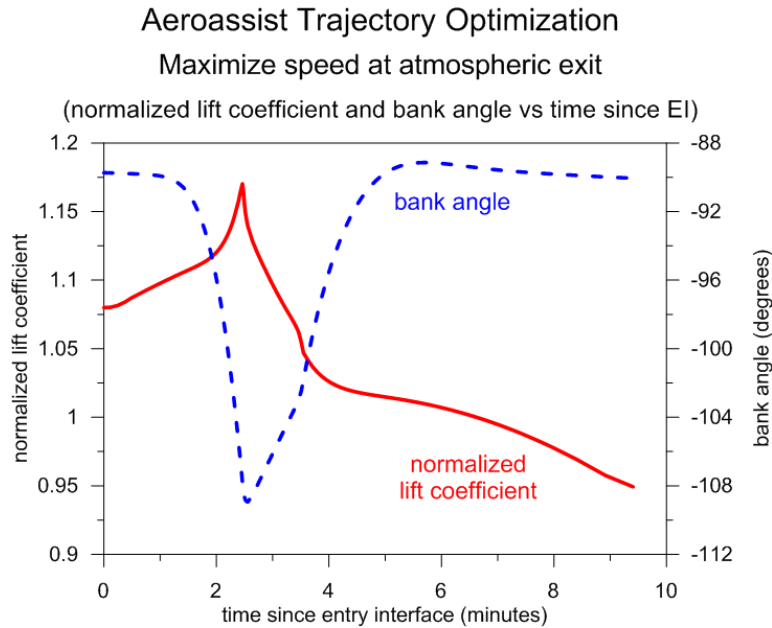
aerodynamic characteristics

drag coefficient at aoa = 0 degrees 3.200000000000000E-002

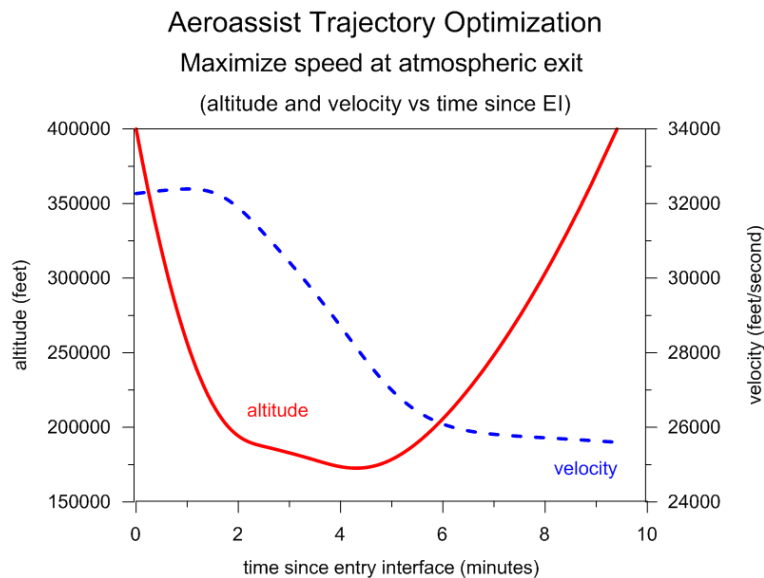
drag coefficient at max L/D 6.400000000000000E-002

lift coefficient at max L/D	0.151185789203691
maximum lift-to-drag ratio	2.36227795630767

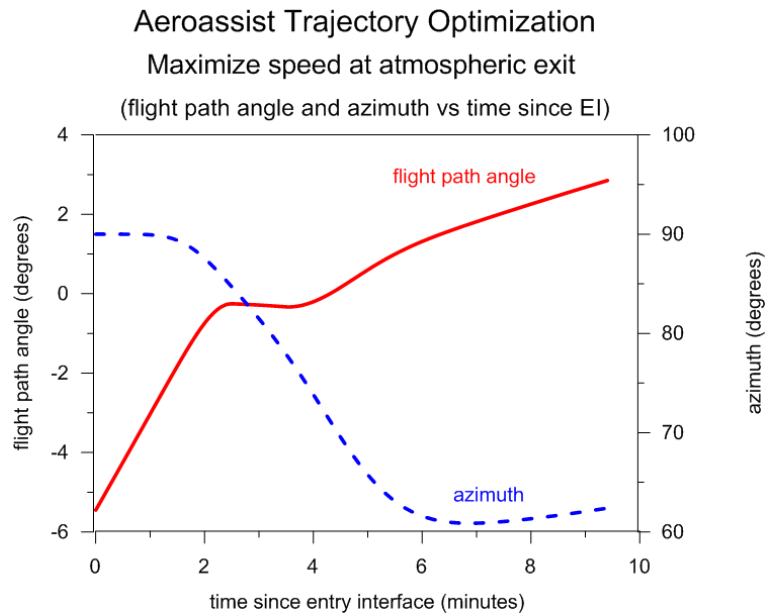
The following are plots created from the trajectory summary file. The first plot illustrates the behavior of the normalized lift coefficient and bank angle during the atmospheric pass.



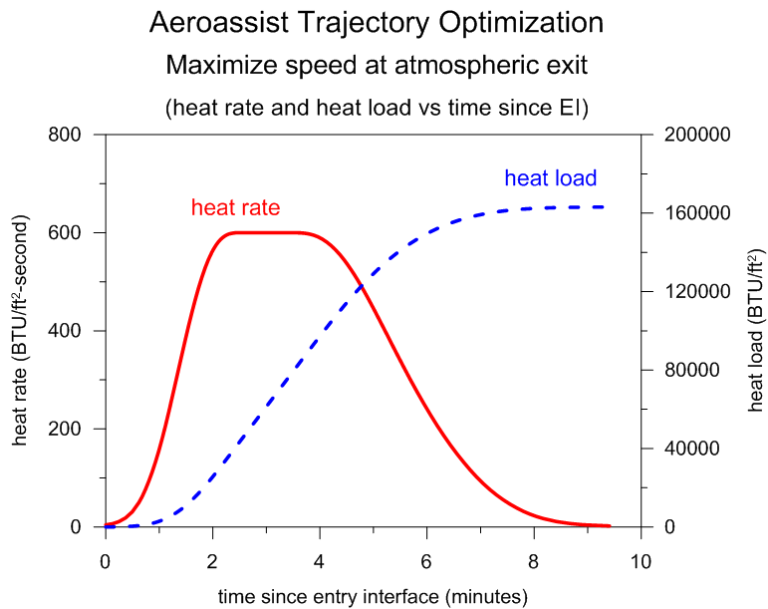
This next plot summarizes the altitude and relative velocity of the vehicle as a function of time since the entry interface (EI).



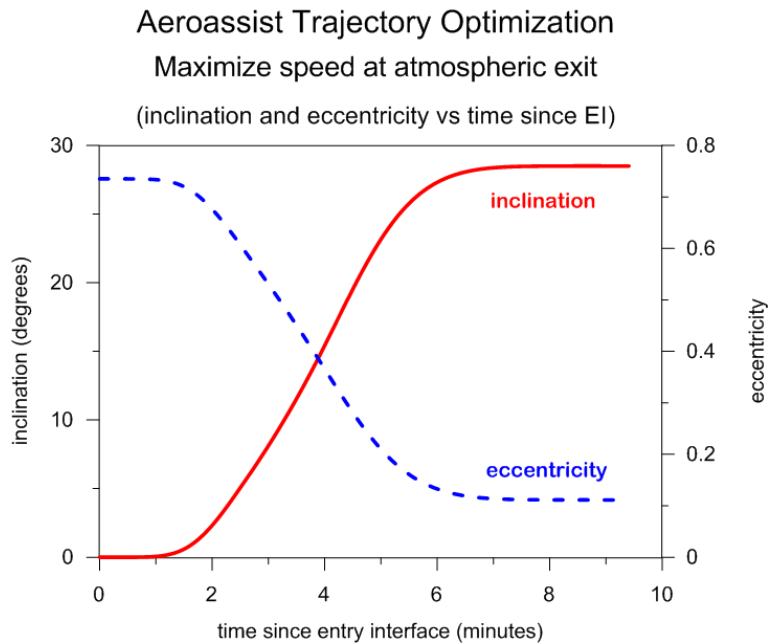
The next plot illustrates the behavior of the relative flight path and azimuth angles during the aero-assist pass through the atmosphere.



This plot illustrates the behavior of the heat rate and heat load during the atmospheric portion of the trajectory. It confirms that the solution has satisfied the maximum heat rate path constraint.



This final plot summarizes the behavior of the orbital eccentricity and inclination during the atmospheric phase of the mission.



Problem setup

This section provides additional details about the *Sparse Optimization Suite* implementation. It briefly explains such things as initial conditions, path constraints and the performance index options.

(1) Entry interface initial conditions

The `aeroassist_ocs` computer program includes three options for specifying initial conditions at the entry interface (EI). This section summarizes these options and describes how the user invokes each.

(a) user input of flight path coordinates at entry interface

For this option, all Earth relative coordinates at the entry interface are defined by the user. The user can provide initial guesses and lower and upper bounds for these coordinates in the `flight conditions` and `bounds at atmospheric entry` part of the input data file.

(b) derived from deorbit maneuver; fixed entry conditions

For this program option, the software uses the flight path angle and entry altitude provided in the input data file to constrain the entry altitude and *inertial* flight path angle. The software then uses the deorbit algorithm described later in this section to compute the Earth relative flight path angle, speed and other flight path coordinates at the entry interface. This option is valid for initial circular orbits.

(c) derived from deorbit maneuver; bounded entry conditions

For this program option, the software will use the entry altitude provided in the input data file to constrain the entry altitude. The flight path angle provided in the data file is used for an *inertial* flight path angle initial guess. During the trajectory optimization, the software will change the inertial flight path angle between the lower and upper bounds provided by the user (the inertial flight path angle is

treated as a problem parameter). The software then uses the deorbit algorithm described later in this section to compute the Earth relative flight path angle, speed and other coordinates at the entry interface. This option is valid for initial circular orbits.

For the second and third initial conditions options, the `aeroassist_ocs` computer program calculates the single impulsive maneuver required to establish an entry interface altitude and flight path angle relative to the user-defined initial circular orbit. The algorithm uses a tangential ΔV applied opposite to the velocity vector to establish the deorbit trajectory. The entry altitude and flight path angle initial guesses are provided by the user.

The algorithm used to compute the scalar magnitude of the deorbit maneuver along with other important flight characteristics is described in Appendix E.

(2) Performance index

This section describes the two types of trajectory optimization performed by the `aeroassist_ocs` software.

(a) *maximize final speed*

The performance index for this type of optimization is simply

$$J = v_f$$

where v_f is the relative speed of the vehicle as it exits from the atmosphere. For this program option, the optimization indicator is set to `maxmin = +1`. This option minimizes the energy loss during the aero-assist maneuver.

(b) *maximize inclination change*

The performance index for this program option is given by

$$J = \cos^{-1} \left(\frac{h_z}{h} \right)$$

where h_z is the z-component of the ECI angular momentum vector and h is the angular momentum magnitude of the vehicle at exit from the atmosphere at the user-defined altitude. For this program option, the optimization indicator is also set to `maxmin = +1`.

(3) Path and point constraints

This section summarizes how the software computes the heat rate and dynamic pressure path constraints, and the orbital inclination point constraint.

(a) *Dynamic pressure*

To enforce this path constraint, the software ensures that

$$q \leq q_{\max}$$

where q_{\max} is the user-defined value of the maximum dynamic pressure. The dynamic pressure at any simulation time is given by

$$q = \frac{1}{2} \rho v^2$$

where ρ is the atmospheric density and v is the relative speed at the current flight condition.

(b) Heat rate

To enforce this path constraint, the software ensures that

$$\dot{Q} \leq \dot{Q}_{\max}$$

where \dot{Q}_{\max} is the user-defined value of the maximum heat rate. The heat rate at any simulation time is computed from Chapman's stagnation point equation given by

$$\dot{Q} = \frac{dQ}{dt} = \frac{17,600}{\sqrt{R_N}} \left(\frac{V}{V_0} \right)^{3.15} \sqrt{\frac{\rho}{\rho_0}} \quad \left(\frac{\text{BTU}}{\text{ft}^2 - \text{sec}} \right)$$

where

R_N = nose radius (feet)

V = relative velocity at the spacecraft location (feet/second)

V_0 = "local circular velocity" at the Earth's surface = $\sqrt{\mu/r_e}$ (feet/second)

ρ = atmospheric density at the spacecraft location (slugs/feet³)

ρ_0 = atmospheric density at the Earth's surface (slugs/feet³)

μ = gravitational constant of the Earth (feet³/second²)

r_e = radius of the Earth (feet)

(c) Orbital inclination

A final orbital inclination point constraint is enforced as follows

$$\cos i = \frac{h_z}{h}$$

where h_z is the z-component of the angular momentum vector and h is the angular momentum magnitude at the atmospheric exit. In this equation, i is the user-defined final orbit inclination.

Technical Discussion

This section describes the numerical algorithms implemented in the `aeroassist_ocs` computer program. It summarizes the equations of motion, Earth gravity model, vehicle aerodynamics, coordinate conversions and other important software features.

Flight path equations of motion

The first-order flight path equations of motion of an aerospace vehicle relative to a rotating spherical Earth and a zonal gravity model are summarized as follows:

geocentric radius

$$\dot{r} = \frac{dr}{dt} = V \sin \gamma$$

geographic longitude

$$\dot{\lambda} = \frac{d\lambda}{dt} = V \frac{\cos \gamma \sin \psi}{r \cos \delta}$$

geocentric declination

$$\dot{\delta} = \frac{d\delta}{dt} = V \frac{\cos \gamma \cos \psi}{r}$$

speed

$$\begin{aligned} \dot{V} = \frac{dV}{dt} = & \frac{(T \cos \alpha - D)}{m} - g_r \sin \gamma + g_\phi \cos \gamma \cos \psi \\ & + \omega_e^2 r \cos \delta (\sin \gamma \cos \delta - \sin \delta \cos \gamma \cos \psi) \end{aligned}$$

flight path angle

$$\begin{aligned} \dot{\gamma} = \frac{d\gamma}{dt} = & \frac{V}{r} \cos \gamma + \left(\frac{T \sin \alpha + L}{mV} \right) \cos \beta - \frac{g_r \cos \gamma}{V} - \frac{g_\phi \sin \gamma \cos \psi}{V} \\ & + 2\omega_e \sin \psi \cos \delta + \omega_e^2 \frac{r}{V} \cos \delta (\cos \psi \sin \gamma \sin \delta + \cos \gamma \cos \delta) \end{aligned}$$

flight azimuth

$$\begin{aligned} \dot{\psi} = \frac{d\psi}{dt} = & \frac{V}{r} \tan \delta \sin \psi \cos \gamma + \left(\frac{T \sin \alpha + L}{mV \cos \gamma} \right) \sin \beta - \frac{\sin \psi}{V \cos \gamma} g_\phi \\ & + 2\omega_e (\sin \delta - \cos \psi \cos \delta \tan \gamma) + \frac{r}{V \cos \gamma} \omega_e^2 \sin \psi \cos \delta \sin \delta \end{aligned}$$

where

- r = geocentric radius
- V = speed
- γ = flight path angle
- δ = geocentric declination
- λ = longitude (+ east)
- ψ = flight azimuth (+ clockwise from north)
- β = bank angle (+ for a right turn)
- α = angle of attack
- L = aerodynamic lift force $= \frac{1}{2} \rho V^2 C_L S$
- D = aerodynamic drag force $= \frac{1}{2} \rho V^2 C_D S$
- T = propulsive thrust
- m = spacecraft mass
- C_L = lift coefficient (non-dimensional)
- C_D = drag coefficient (non-dimensional)
- S = aerodynamic reference area
- ρ = atmospheric density $= f(h)$
- h = altitude

and

- r_e = Earth equatorial radius
- ω_e = Earth inertial rotation rate
- g_r = radial component of gravity
- g_ϕ = latitudinal component of gravity

The bank angle is the angle between the instantaneous orbit plane and the aerodynamic lift vector. The bank angle is positive for a right turn as viewed from the rear of the vehicle.

Earth gravity model

The components of the gravity vector can be determined from the gradient of the potential function according to

$$\mathbf{F}_G = \nabla U = \begin{Bmatrix} \frac{1}{r} \frac{\partial U}{\partial \phi} \\ 0 \\ -\frac{\partial U}{\partial r} \end{Bmatrix} = \begin{Bmatrix} g_\phi \\ 0 \\ g_r \end{Bmatrix}$$

where

$$U = \frac{\mu}{r} \left[1 - \sum_{j=1}^{\infty} \left(\frac{r_e}{r} \right)^j J_j P_{j0}(\sin \phi) \right]$$

$$\frac{\partial U}{\partial r} = \frac{\mu}{r^2} \left[-1 + \sum_{l=2}^{\infty} (l+1) \left(\frac{r_e}{r} \right)^l J_l P_{l0}(\sin \phi) \right]$$

$$\frac{1}{r} \frac{\partial U}{\partial \phi} = -\frac{\mu}{r^2} \sum_{l=2}^{\infty} \left(\frac{r_e}{r} \right)^l J_l \frac{\partial P_{l0}}{\partial \phi}$$

$$P_{j0}(\sin \phi) = \sum_{t=0}^k T_{jt} \sin^{j-2t} \phi$$

$$T_{jt} = (-1)^t (2j-2t)! / 2t! (j-t)! (j-2t)!$$

For a *zonal-only* gravity model of order four, the Legendre functions and their partial derivatives are given by

$$P_{20} = \frac{1}{2} (3 \sin^2 \phi - 1) \quad P_{30} = \frac{1}{2} (5 \sin^3 \phi - 3 \sin \phi) \quad P_{40} = \frac{1}{8} (35 \sin^4 \phi - 30 \sin^2 \phi + 3)$$

$$\frac{\partial P_{20}}{\partial \phi} = 3 \sin \phi \cos \phi \quad \frac{\partial P_{30}}{\partial \phi} = \frac{3}{2} (5 \sin^2 \phi - 1) \cos \phi \quad \frac{\partial P_{40}}{\partial \phi} = \frac{5}{2} (7 \sin^2 \phi - 3) \sin \phi \cos \phi$$

This is the Earth gravity model implemented in the `aeroassist_ocs` computer program.

Coordinate systems and transformations

This section describes the relationship between flight path coordinates and inertial position and velocity. Flight path coordinates are used during the aero-assist pass and inertial coordinates are used to model the deorbit-to-entry portion of the flight.

(1) converting an ECI state vector to flight path coordinates

This coordinate conversion is necessary for `aeroassist_ocs` simulations that determine entry interface conditions from an impulsive deorbit maneuver from an initial circular orbit.

The transformation of an ECI position vector \mathbf{r}_{eci} to an ECF position vector \mathbf{r}_{ecf} is given by the following vector-matrix operation

$$\mathbf{r}_{ECF} = [\mathbf{T}] \mathbf{r}_{ECI}$$

where the elements of the transformation matrix $[\mathbf{T}]$ are given by

$$[\mathbf{T}] = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and θ is the Greenwich apparent sidereal time at the moment of interest. Greenwich apparent sidereal time is given by the following expression:

$$\theta = \theta_{g0} + \omega_e t$$

where θ_{g0} is the Greenwich apparent sidereal time at 0 hours UT, ω_e is the inertial rotation rate of the Earth, and t is the elapsed time since 0 hours UT.

Finally, the flight path coordinates are determined from the following set of equations

$$\begin{aligned} r &= \sqrt{r_{ECF_x}^2 + r_{ECF_y}^2 + r_{ECF_z}^2} & v &= \sqrt{v_{ECF_x}^2 + v_{ECF_y}^2 + v_{ECF_z}^2} \\ \lambda &= \tan^{-1}(r_{ECF_y}, r_{ECF_x}) & \delta &= \sin^{-1}\left(\frac{r_{ECF_z}}{|\mathbf{r}_{ECF}|}\right) \\ \gamma &= \sin^{-1}\left(-\frac{v_{r_z}}{|\mathbf{v}_r|}\right) & \psi &= \tan^{-1}[v_{r_y}, v_{r_x}] \end{aligned}$$

where

$$\mathbf{v}_r = \begin{bmatrix} -\sin \delta \cos \lambda & -\sin \delta \sin \lambda & \cos \delta \\ -\sin \lambda & \cos \lambda & 0 \\ -\cos \delta \cos \lambda & -\cos \delta \sin \lambda & -\sin \delta \end{bmatrix} \mathbf{v}_{ECF}$$

(2) converting flight path coordinates to an ECI state vector

This coordinate conversion is necessary in order to determine the orbital inclination and other orbital characteristics at exit from the atmosphere.

The Earth-centered-fixed (ECF) position and velocity vectors of the aero-assist vehicle can be determined from the flight path coordinates with the following set of equations:

$$\begin{aligned} r_{ECF_x} &= r \cos \delta \cos \lambda \\ r_{ECF_y} &= r \cos \delta \sin \lambda \\ r_{ECF_z} &= r \sin \delta \end{aligned}$$

$$\begin{aligned}
v_{ECF_x} &= v \left[\cos \lambda \left(-\cos \psi \sin \beta \sin \delta + \cos \beta \cos \delta \right) - \sin \psi \sin \beta \sin \lambda \right] \\
v_{ECF_y} &= v \left[\sin \lambda \left(-\cos \psi \sin \beta \sin \delta + \cos \beta \cos \delta \right) + \sin \psi \sin \beta \cos \lambda \right] \\
v_{ECF_z} &= v \left(\cos \psi \cos \delta \sin \beta + \cos \beta \cos \delta \right)
\end{aligned}$$

where $\beta = 90^\circ - \gamma$.

The transformation from ECF to ECI coordinates involves the transpose of the ECI-to-ECF transformation matrices described above as follows:

$$\begin{aligned}
\mathbf{r}_{ECI} &= [\mathbf{T}]^T \mathbf{r}_{ECF} \\
\mathbf{v}_{ECI} &= [\mathbf{T}]^T \dot{\mathbf{r}}_{ECF} + [\dot{\mathbf{T}}]^T \mathbf{r}_{ECF} = [\mathbf{T}]^T \mathbf{v}_{ECF} + [\dot{\mathbf{T}}]^T \mathbf{r}_{ECF}
\end{aligned}$$

The ECF velocity vector is determined by differentiating this expression

$$\mathbf{v}_{ECF} = [\mathbf{T}] \dot{\mathbf{r}}_{ECI} + [\dot{\mathbf{T}}] \mathbf{r}_{ECI} = [\mathbf{T}] \mathbf{v}_{ECI} + [\dot{\mathbf{T}}] \mathbf{r}_{ECI}$$

The elements of the $[\dot{\mathbf{T}}]$ matrix are as follows:

$$[\dot{\mathbf{T}}] = \begin{bmatrix} -\omega_e \sin \theta & \omega_e \cos \theta & 0 \\ -\omega_e \cos \theta & -\omega_e \sin \theta & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Aerodynamic characteristics of aero-assist vehicles

This section provides general information about vehicle aerodynamics. It also describes the type of aerodynamic modeling used in the `aeroassist_ocs` computer program.

General form of a drag polar

$$C_D = C_{D_0} + k |C_L|^n$$

Lift-to-drag ratio

$$E = \frac{L}{D} = \frac{C_L}{C_D} = \frac{C_L}{C_{D_0} + k C_L^n}$$

Maximum lift-to-drag ratio (value of E at which $dE/dC_L = 0$)

$$E^* = \frac{(C_{D_0} + k C_L^n) - C_L (n k C_L^{n-1})}{(C_{D_0} + k C_L^n)^2}$$

Lift coefficient at maximum lift-to-drag ratio

$$C_L^* = \sqrt[n]{\frac{C_{D_0}}{k(n-1)}}$$

Drag coefficient at maximum lift-to-drag ratio

$$C_D^* = \frac{nC_{D_0}}{(n-1)}$$

In general,

$$E^* = \frac{C_L^*}{C_D^*} = \frac{\sqrt[n]{(n-1)^{n-1}}}{n\sqrt[n]{kC_{D_0}^{n-1}}}$$

For a *parabolic* drag polar ($n = 2$),

$$C_D = C_{D_0} + kC_L^2$$

where

C_D = drag coefficient

C_{D_0} = drag coefficient at angle-of-attack = 0°

C_L = lift coefficient

k = constant

and

$C_D^* = 2C_{D_0}$ = drag coefficient at maximum L/D

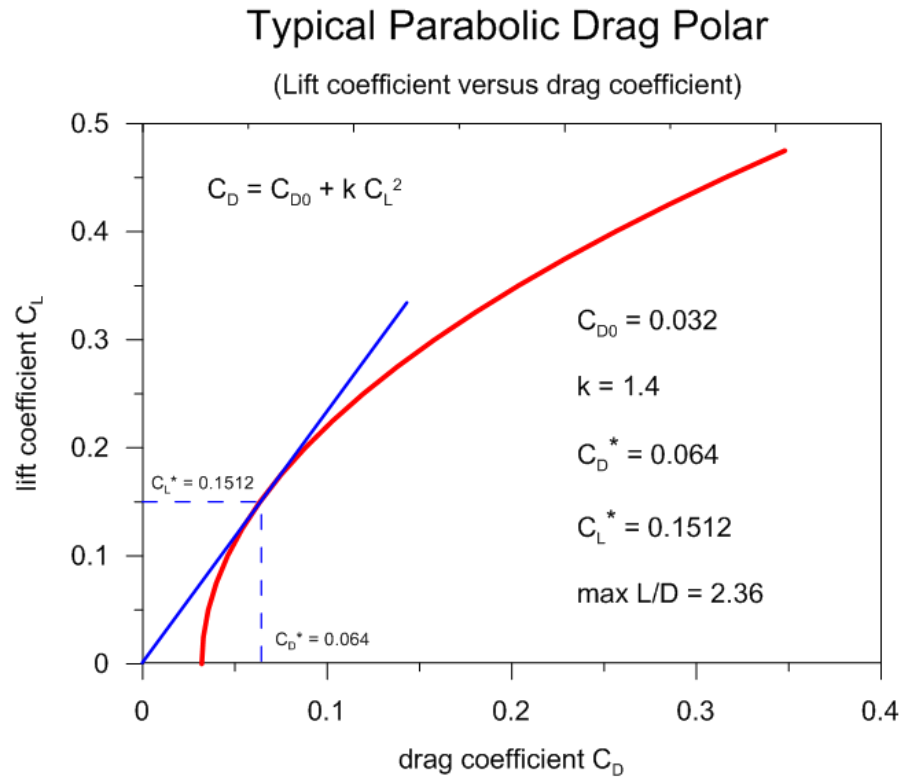
$C_L^* = \sqrt{C_{D_0}/k}$ = lift coefficient at maximum L/D

$E^* = \left(\frac{C_L}{C_D} \right)_{\max} = \frac{1}{2\sqrt{kC_{D_0}}} = \text{maximum L/D}$

In the `aeroassist_ocs` computer program, the vehicle aerodynamics are modeled using a parabolic drag polar ($n = 2$). The normalized lift coefficient is given by $\lambda = C_L/C_L^*$ where C_L is the lift coefficient at any simulation time and C_L^* is the lift coefficient corresponding to maximum L/D .

In this computer program, the control variables are the normalized lift coefficient and bank angle.

The following is a graphic display of a typical parabolic drag polar.



References and Bibliography

- (1) "Direct Trajectory Optimization Using Nonlinear Programming and Collocation", C. R. Hargraves and S. W. Paris, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 10, No. 4, July-August, 1987, pp. 338-342.
- (2) "Optimal Finite-Thrust Spacecraft Trajectories Using Direct Transcription and Nonlinear Programming", Paul J. Enright, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1991.
- (3) "Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.
- (5) "Improved Collocation Methods with Application to Direct Trajectory Optimization", Albert L. Herman, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1995.
- (6) "Minimum-Fuel Aero-assisted Coplanar Orbit Transfer Using Lift Modulation", Kenneth D. Mease and Nguyen X. Vinh, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 8, No. 1, Jan.-Feb. 1985.
- (7) "Survey of Numerical Methods for Trajectory Optimization", John T. Betts, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998, pp. 193-207.
- (8) "Variational Solutions for the Heat-Rate-Limited Aero-assisted Orbital Transfer Problem", Hans Seywald, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 19, No. 3, May-June, 1996, pp. 686-692.
- (9) "Fuel-Optimal Trajectories of Aero-assisted Orbital Transfer with Plane Change", D. S. Naidu, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27, No. 2, March 1991, pp. 361-368.
- (10) *Optimal Trajectories in Atmospheric Flight*, N. X. Vinh, Elsevier, 1981.
- (11) *Hypersonic and Planetary Flight Mechanics*, N. X. Vinh, R. D. Culp, and A. Busemann, University of Michigan Press, 1980.
- (12) "A Survey of Aero-assisted Orbit Transfer", G. D. Walberg, *AIAA Journal of Spacecraft and Rockets*, Vol. 22, No. 1, Jan-Feb 1985, pp. 3-18.
- (13) "Optimal Maneuvers of Orbital Transfer Vehicles", John M. Hanson, Ph.D. Thesis, University of Michigan, Department of Aerospace Engineering, 1983.
- (14) "Optimal Aero-assisted Return From High Earth Orbit With Plane Change", Nguyen X. Vinh and John M. Hanson, *Acta Astronautica*, Vol. 12, No. 1, 1985.
- (15) "Combining Propulsive and Aerodynamic Maneuvers to Achieve Optimal Orbital Transfer", John M. Hanson, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 12, No. 5, Sept.-Oct. 1989.

APPENDIX A

Contents of the Simulation Summary CSV File

This appendix is a brief summary of the information contained in the CSV data file produced by the `aeroassist_ocs` software. The comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information:

time (min) = simulation time since entry interface in minutes

altitude (ft) = altitude relative to a spherical Earth in feet

velocity (fps) = Earth-relative velocity in feet per second

flight path angle (d) = Earth-relative flight path angle in degrees

azimuth (deg) = Earth-relative azimuth angle in degrees

declination (deg) = geocentric declination in degrees

longitude (deg) = geographic longitude in degrees

mach number = Mach number (non-dimensional)

dynamic pressure (psf) = dynamic pressure in pounds per square foot

bank angle (deg) = bank angle in degrees

heat rate (btu/ft²-s) = heat rate in BTU/ft²-second

heat load (btu/ft²) = accumulated heat load in BTU/ft²

lift-to-drag = lift-to-drag ratio (non-dimensional)

lift coefficient = lift coefficient (non-dimensional)

drag coefficient = drag coefficient (non-dimensional)

lift force (lbf) = lift force in pounds

drag force (lbf) = drag force in pounds

density (slugs/ft³) = atmospheric density in slugs per cubic feet

pressure (psf) = atmospheric pressure in pounds per square foot

temperature (deg K) = atmospheric temperature in degrees K

crossrange (nm) = crossrange distance in nautical miles

downrange (nm) = downrange distance in nautical miles

altitude rate (fps) = rate of change of altitude in feet per second

longitude rate (dps) = rate of change of longitude in degrees per second

declination rate (dps) = rate of change of declination in degrees per second

velocity rate (fps/s) = rate of change of velocity in feet per second per second

fpa rate (dps) = rate of change of flight path angle in degrees per second

azimuth rate (dps) = rate of change of azimuth angle in degrees per second

perigee altitude (nm) = perigee altitude in nautical miles

perigee radius (nm) = perigee radius in nautical miles

apogee altitude (nm) = apogee altitude in nautical miles

apogee radius (nm) = apogee radius in nautical miles

Notes:

- (1) The accumulated heat load is determined from a cubic spline integration of the heat rate of the optimized solution at all collocation nodes.
- (2) The rate of change of the flight variables is determined from the equations of motion.

APPENDIX B

Fortran Functions and Subroutines

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

Aeroassist_ocs.f - main executive program

atan3.for - four quadrant inverse tangent function

atmos76.for - U.S. Standard 1976 atmosphere model

cdeorbit.for - impulsive deorbit from a circular orbit subroutine

crdr.for - subroutine that calculates crossrange and downrange

csint.for - cubic spline integration of tabular data subroutine

gast.for - Greenwich apparent sidereal time subroutine

gravity.for - fourth-order zonal gravity model 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

readfpn.for - read and echo floating point number from an input file subroutine

readint.for - read and echo an integer from an input file subroutine

readtext.for - read and echo text from an input file subroutine

twobody2.for - two-body orbit propagation subroutine

utility.for - number and text manipulation functions and subroutines

us76.for - U.S. standard 1976 atmosphere subroutine

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 the source code for a single Fortran 77 subroutine and illustrates typical programming conventions used in the `aeroassist_ocs` software. This subroutine is the point function routine required by the *Sparse Optimization Suite*.

```
      subroutine odepf(iphase, iphend, time, ydyn, nydyn, parm,
&                    nparm, ptf, nptf, iferr)

c     aeroassist point functions
c     *****

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

      include 'socscom1.inc'

      include 'pconstr.inc'

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

      dimension ydyn(nydyn), parm(nparm), ptf(nptf), ywrk(6)

      dimension reci(3), veci(3), fpc(6), hv(3), oev(6)

      iferr = 0

c     -----
c     extract current flight path coordinates
c     -----

c     altitude (feet)

      xalt = ydyn(1)

c     longitude (radians)

      elon = ydyn(2)

c     geocentric declination

      dec = ydyn(3)

c     relative speed (feet/second)

      vrel = ydyn(4)

c     flight path angle (radians)

      fpa = ydyn(5)

c     flight azimuth (radians)

      azim = ydyn(6)

c     geocentric radius (feet)

      rmag = xalt + req

      if (iphase .eq. 1 .and. iphend .eq. -1
&        .and. ic_type .eq. 3) then
```

```

c      -----
c      beginning of phase 1 - atmospheric entry
c      (parameter #1 ==> inertial flight path angle)
c      -----

c      current inertial flight path angle

      fpae = parm(1)

c      compute flight path coordinates at entry interface
c      using user-defined entry altitude and orbital elements

      call cdeorbit(fpae, ywrk)

c      -----
c      match states at atmospheric entry
c      -----

c      east longitude (radians)

      ptf(1) = ydyn(2) - ywrk(2)

c      declination (radians)

      ptf(2) = ydyn(3) - ywrk(3)

c      velocity (feet/second)

      ptf(3) = ydyn(4) - ywrk(4)

c      flight path angle (radians)

      ptf(4) = ydyn(5) - ywrk(5)

c      azimuth (radians)

      ptf(5) = ydyn(6) - ywrk(6)

end if

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

c      -----
c      end of phase 1 - atmospheric exit
c      -----

c      compute inertial state vector at end of phase 1

      fpc(1) = elon
      fpc(2) = dec
      fpc(3) = fpa
      fpc(4) = azim
      fpc(5) = rmag
      fpc(6) = vrel

      call fpc2eci(time, fpc, reci, veci)

c      compute angular momentum vector and magnitude

      call vcross(reci, veci, hv)

```



```

hmag = vecmag(hv)

if (ic_inc .eq. 1) then

C      -----
C      final orbit inclination constraint at atmospheric exit
C      (constrain cosine of final orbit inclination)
C      -----

      ptf(1) = hv(3) / hmag

end if

if (iopt .eq. 2) then

C      -----
C      maximize orbital inclination at atmospheric exit
C      -----

      ptf(1) = acos(hv(3) / hmag)

end if

end if

return
end

```

APPENDIX D

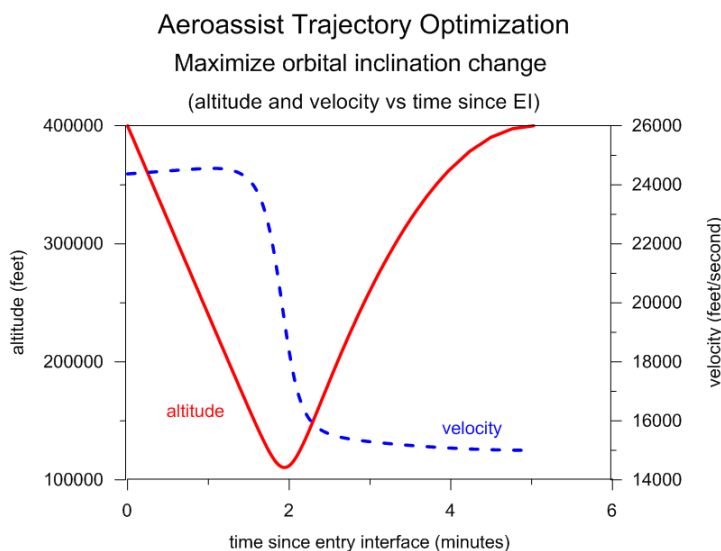
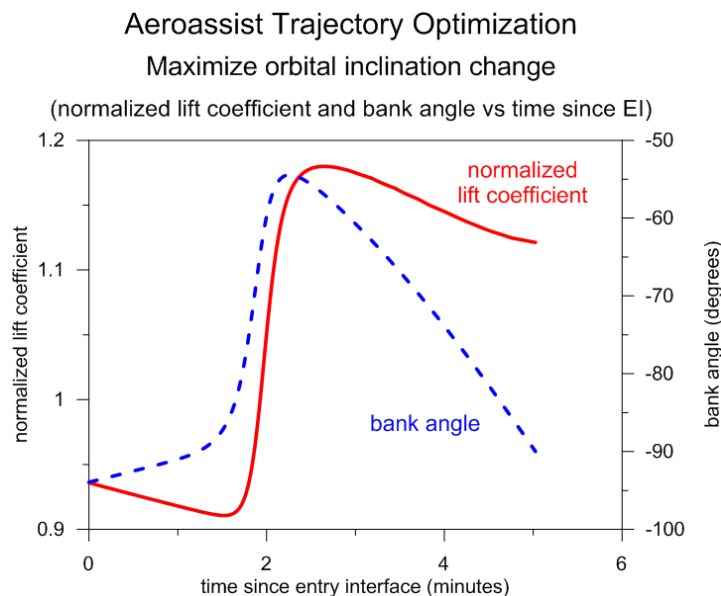
Maximize Orbital Inclination Example

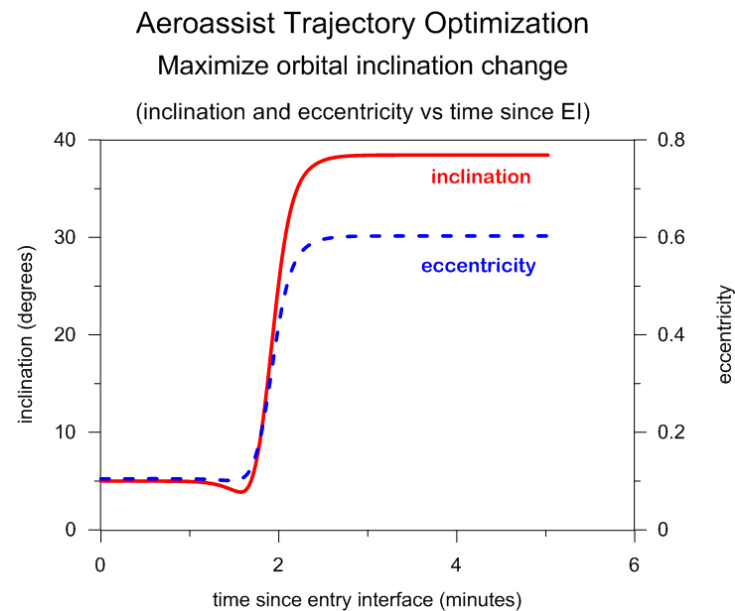
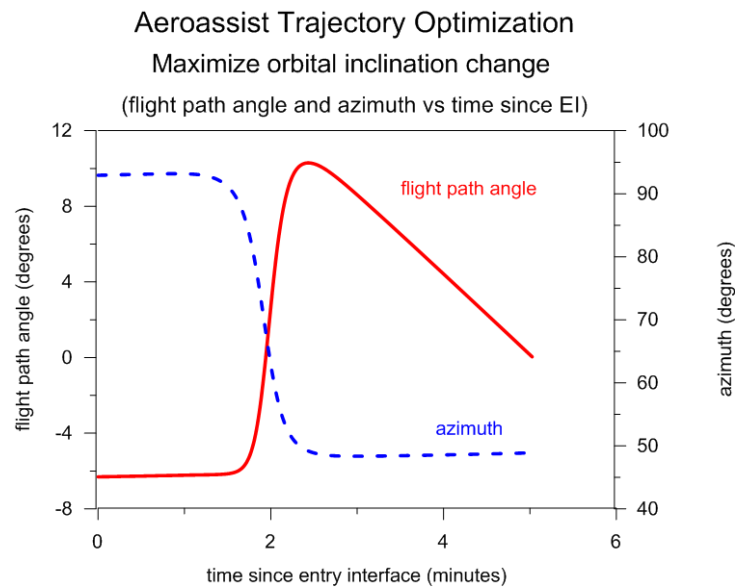
This appendix contains graphics and a simulation summary for an aero-assist trajectory that maximizes the orbital inclination change. The mission starts in a 500 nautical mile circular Earth orbit with an initial inclination equal to 5 degrees. The speed at atmospheric exit is constrained to be $\geq 15,000$ feet per second.

For this type of trajectory optimization make sure the orbital inclination at the atmospheric exit is not constrained by using the following statement in the input file

```
enforce an orbital inclination constraint (yes or no)
no
```

The following are plots of the important trajectory parameters for this example.





The following is the `aeroassist_ocs` program output for this example.

```

program aeroassist_ocs
=====

input file ==> leo2leo_max_inc.in

bounded entry conditions derived from deorbit maneuver

maximize orbital inclination change

orbital elements and state vector prior to deorbit impulse
-----

calendar date          January    1, 2001

```

universal time	00:00:00.000		
sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.394392019016D+04	0.248334991895D-15	0.500000000000D+01	0.000000000000D+00
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.431780632080D-14	0.300000000000D+02	0.300000000000D+02	0.103541236919D+03
r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.394392019016D+04	0.500000000000D+03	0.394392019016D+04	0.500000000000D+03
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
0.207531855633D+08	0.119362626872D+08	0.104428766999D+07	0.239637145430D+08
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.121182361413D+05	0.209095296884D+05	0.182934680739D+04	0.242364722827D+05

orbital elements and state vector after deorbit impulse

calendar date	January 1, 2001		
universal time	00:00:00.000		
sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.357017458724D+04	0.104685525536D+00	0.500000000000D+01	0.210000000000D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.456326147825D-14	0.180000000000D+03	0.300000000000D+02	0.891775123771D+02
r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.319642898432D+04	-.247491205844D+03	0.394392019016D+04	0.500000000000D+03
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
0.207531855633D+08	0.119362626872D+08	0.104428766999D+07	0.239637145430D+08
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.114664033296D+05	0.197848183550D+05	0.173094731598D+04	0.229328066592D+05

flight path coordinates at atmospheric entry

altitude	400000.000000000	feet
velocity	24368.0971648879	feet/second
declination	4.17633335544806	degrees
longitude	16.1785108288942	degrees
azimuth	92.9277990467808	degrees
flight path angle	-6.30689859387635	degrees
inertial fpa	-5.93056753615202	degrees

orbital elements and state vector at atmospheric entry

calendar date	January 1, 2001		
universal time	00:26:03.919		
sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.357017458725D+04	0.104685525537D+00	0.500000000013D+01	0.20999999997D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.373173703543D-06	0.273322949484D+03	0.123322949481D+03	0.891775123775D+02

r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.319642898433D+04	-.247491205835D+03	0.394392019018D+04	0.500000000016D+03
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
-.117154107210D+08	0.177516413688D+08	0.155306738546D+07	0.213256568000D+08
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.200622216447D+05	-.163311939918D+05	-.142879432473D+04	0.259083401194D+05

flight path coordinates at atmospheric exit

altitude	400000.000000000	feet
velocity	15000.000000000	feet/second
declination	8.99670339727738	degrees
longitude	29.5623003632225	degrees
azimuth	48.8880628810677	degrees
flight path angle	3.984582451546240E-002	degrees

orbital elements and state vector at atmospheric exit

calendar date January 1, 2001

universal time 00:31:05.462

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.218954752865D+04	0.602957760689D+00	0.384433270708D+02	0.194591552768D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.126562332360D+03	0.179975688582D+03	0.145672413500D+02	0.428304598571D+02
r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.869342853852D+03	-.257457733631D+04	0.350975220345D+04	0.658320132855D+02
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
-.156695378708D+08	0.140757933205D+08	0.333485580891D+07	0.213256568000D+08
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.743898214339D+04	-.105738521023D+05	0.974327143645D+04	0.161887659164D+05

aerodynamic characteristics

drag coefficient at aoa = 0 degrees 5.000000000000000E-002

drag coefficient at max L/D 0.100000000000000

lift coefficient at max L/D 0.188982236504614

maximum lift-to-drag ratio 1.88982236504614

APPENDIX E

De-orbit from a Circular Earth Orbit

The scalar magnitude of the single impulsive maneuver required to de-orbit a spacecraft from an initial circular orbit can be determined from the following expression

$$\Delta V = V_{c_e} \sqrt{\frac{1}{\tilde{r}}} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\} = V_{c_i} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\}$$

where

$$\tilde{r} = \frac{h_i + r_{eq}}{h_e + r_{eq}} = \frac{r_i}{r_e} = \text{radius ratio}$$

$$V_{c_e} = \sqrt{\frac{\mu}{(h_e + r_{eq})}} = \sqrt{\frac{\mu}{r_e}} = \text{local circular velocity at entry interface}$$

$$V_{c_i} = \sqrt{\frac{\mu}{(h_i + r_{eq})}} = \sqrt{\frac{\mu}{r_i}} = \text{local circular velocity of initial circular orbit}$$

γ_e = flight path angle at entry interface

h_i = altitude of initial circular orbit

h_e = altitude at entry interface

r_i = radius of initial circular orbit

r_e = radius at entry interface

r_{eq} = Earth equatorial radius

μ = Earth gravitational constant

This algorithm is described in the technical article, “Deboost from Circular Orbits”, A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, Jul-Aug., 1966. Additional information can be found in Chapter 5 of *Hypersonic and Planetary Entry Flight Mechanics* by Vinh, Busemann and Culp, The University of Michigan Press.

The true anomaly on the de-orbit trajectory at the entry interface θ_e can be determined from the following two equations

$$\sin \theta_e = \frac{\dot{r}}{e_d} \sqrt{\frac{a_d(1-e_d^2)}{\mu}}$$

$$\cos \theta_e = \frac{a_d(1-e_d^2)}{e_d r_e} - \frac{1}{e_d}$$

and the following four quadrant inverse tangent operation

$$\theta_e = \tan^{-1}(\sin \theta_e, \cos \theta_e)$$

where

e_d = eccentricity of the de-orbit trajectory

a_d = semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu[2a_d r_e - r_e^2 - a_d^2(1-e_d^2)]}{a_d r_e^2}}$$

The elapsed time-of-flight between perigee of the de-orbit trajectory and the entry true anomaly θ_e is given by

$$t(\theta_e) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1-e_d}{1+e_d}} \tan \frac{\theta_e}{2} \right\} - \frac{e_d \sqrt{1-e_d^2} \sin \theta_e}{1+e_d \cos \theta_e} \right]$$

In this equation τ is the Keplerian orbital period of the de-orbit trajectory and is equal to $2\pi\sqrt{a_d^3/\mu}$.

Therefore, the flight time between the de-orbit impulse and entry interface is given by

$$\Delta t = t(\theta_e) - t(180^\circ) = t(\theta_e) - \frac{\tau}{2}$$

Finally, the orbital speed at the entry interface V_e can be determined from

$$V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$$

APPENDIX F

Typical Sparse Optimization Suite Configuration File

The `aeroassist_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 *Sparse Optimization Suite* user's manual. Please note that the `aeroassist_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 `aeroassist` 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=0
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
NPATAUX=100
```


NSSWCH=-1
 SOCOUT=A0B0C0D0E0F0G0H0I0J2K0L0M0N0O0P0Q0R0S1T0U0V0W0X0Y0Z0
 SPRTHS=SPARSE
 NLPALG=SNLPMN
 NLPOMR=M
 KEYDPL=.lueiLUE
 RHSTMP=RHSTMPLT
 RSTFIL=socx.restart
 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=500000
 INSNLP:MNSAME=2
 INSNLP:NEWTON=0
 INSNLP:NITMAX=50000
 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

Single Impulse De-orbit

This document describes two MATLAB scripts that be used to compute the characteristics of single impulse de-orbit from Earth orbits. Scripts are provided for calculating the impulsive maneuver required to de-orbit from both circular and elliptical orbits.

cdeorbit.m – single impulse de-orbit from a circular orbit

This MATLAB script calculates the single impulsive maneuver required to establish a reentry altitude and flight path angle relative to a non-rotating, spherical Earth. The algorithm uses a tangential delta-v applied opposite to the velocity vector of an initial circular orbit to establish the de-orbit trajectory.

The scalar magnitude of the single impulsive maneuver required to de-orbit a spacecraft from an initial circular orbit can be determined from the following expression

$$\Delta V = V_{c_e} \sqrt{\frac{1}{\tilde{r}}} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\} = V_{c_i} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\}$$

where

$$\tilde{r} = \frac{h_i + r_{eq}}{h_e + r_{eq}} = \frac{r_i}{r_e} = \text{radius ratio}$$

$$V_{c_e} = \sqrt{\frac{\mu}{(h_e + r_{eq})}} = \sqrt{\frac{\mu}{r_e}} = \text{local circular velocity at entry interface}$$

$$V_{c_i} = \sqrt{\frac{\mu}{(h_i + r_{eq})}} = \sqrt{\frac{\mu}{r_i}} = \text{local circular velocity of initial circular orbit}$$

and

γ_e = flight path angle at entry interface

h_i = altitude of initial circular orbit

h_e = altitude at entry interface

r_i = radius of initial circular orbit

r_e = radius at entry interface

r_{eq} = Earth equatorial radius

μ = Earth gravitational constant

This algorithm is described in the technical article, “Deboost from Circular Orbits”, A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, Jul-Aug., 1966. Additional information can be found in Chapter 5 of *Hypersonic and Planetary Entry Flight Mechanics* by Vinh, Busemann and Culp, The University of Michigan Press.

The true anomaly on the de-orbit trajectory at the entry interface θ_e can be determined from the following two equations

$$\sin \theta_e = \frac{\dot{r}}{e_d} \sqrt{\frac{a_d(1-e_d^2)}{\mu}}$$

$$\cos \theta_e = \frac{a_d(1-e_d^2)}{e_d r_e} - \frac{1}{e_d}$$

and the following four quadrant inverse tangent operation

$$\theta_e = \tan^{-1}(\sin \theta_e, \cos \theta_e)$$

where

e_d = eccentricity of the de-orbit trajectory

a_d = semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu[2a_d r_e - r_e^2 - a_d^2(1-e_d^2)]}{a_d r_e^2}}$$

The elapsed time-of-flight between perigee of the de-orbit trajectory and the entry true anomaly θ_e is given by

$$t(\theta_e) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1-e_d}{1+e_d}} \tan \frac{\theta_e}{2} \right\} - \frac{e_d \sqrt{1-e_d^2} \sin \theta_e}{1+e_d \cos \theta_e} \right]$$

In this equation τ is the Keplerian orbital period of the de-orbit trajectory and is equal to $2\pi\sqrt{a_d^3/\mu}$.

Therefore, the flight time between the de-orbit impulse and entry interface is given by

$$\Delta t = t(\theta_e) - t(180^\circ) = t(\theta_e) - \frac{\tau}{2}$$

Finally, the orbital speed at the entry interface V_e can be determined from

$$V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$$

Orbital Mechanics with MATLAB

This MATLAB script will prompt you for the altitude of the initial circular orbit, and the entry altitude and flight path angle. The following is a typical user interaction with this script.

```
program cdeorbit

< single impulse deorbit from circular orbits >

please input the initial altitude (kilometers)
? 1000

please input the entry altitude (kilometers)
? 100

please input the entry flight path angle (degrees)
? -2
```

The following is the script output created for this example.

```
program cdeorbit

< single impulse deorbit from circular orbits >

initial altitude      1000.000000  kilometers
entry altitude        100.000000  kilometers
entry fpa             -2.000000   degrees

entry trajectory

semimajor axis        6896.07935765  kilometers
eccentricity          0.06990358
argument of perigee   180.00000000  degrees
perigee altitude      35.87871531  kilometers
apogee altitude       1000.00000000  kilometers
entry true anomaly    328.04948058  degrees
entry velocity        8078.31275892  meters/second
impulse-to-entry time 40.13350666  minutes
deorbit delta-v       261.55416617  meters/second
```

The software will also calculate and display the entry velocity and flight path angle relative to a rotating spherical Earth. The following is the relative flight information for this example.

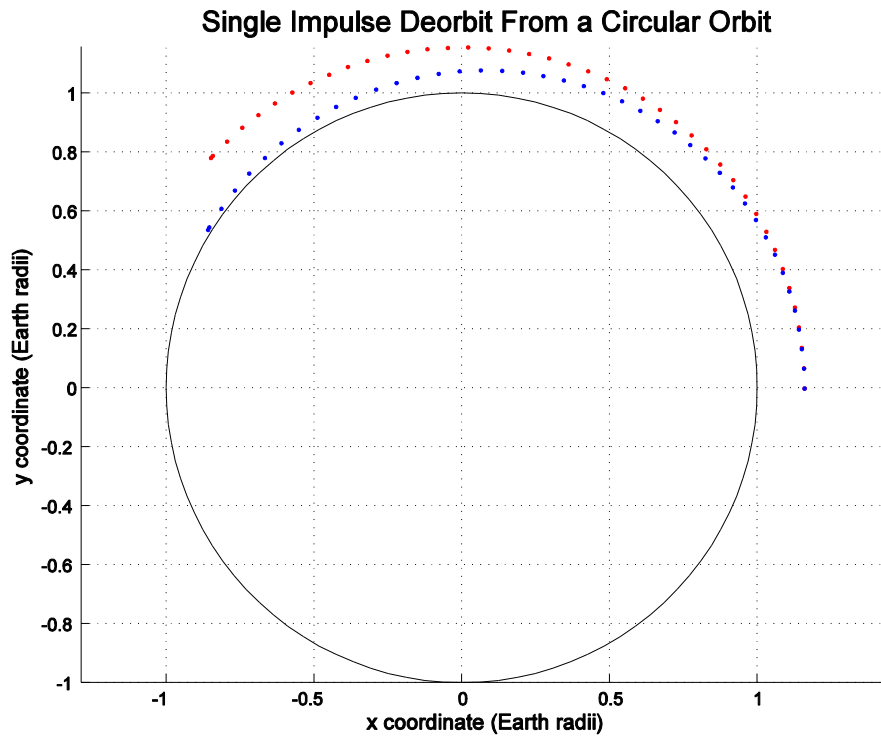
Orbital Mechanics with MATLAB

relative flight path coordinates

flight path angle -2.12418719 degrees

velocity magnitude 7.60622497 kilometers/second

Finally, the software will graphically display the initial circular orbit and the de-orbit trajectory. The graphic display for this example is as follows where the red dots represent the original circular orbit and the blue dots represent the de-orbit trajectory, both at one minute intervals. The black circle is the surface of a spherical Earth and the distances are in Earth radii.



The maneuver creates an elliptical de-orbit trajectory with an apogee located at the maneuver point. The apogee altitude of this trajectory is equal to the altitude of the initial circular orbit.

edeorbit.m – single impulse de-orbit from an elliptical orbit

This MATLAB script calculates the single impulsive maneuver required to establish a reentry altitude and flight path angle relative to a non-rotating spherical Earth. The algorithm uses a tangential ΔV applied opposite to the velocity vector at apogee of the initial elliptical orbit to establish the de-orbit trajectory that enters the Earth's atmosphere.

The scalar magnitude of this de-orbit delta-v is given by

$$\Delta V = \sqrt{\frac{\mu}{r_e}} \left(\sqrt{\frac{2\tilde{r}_p}{\tilde{r}_a(\tilde{r}_a + \tilde{r}_p)}} - \sqrt{\frac{2(\tilde{r}_a - 1)}{\tilde{r}_a(\tilde{r}_a^2 - \cos^2 \gamma_e)}} \cos \gamma_e \right)$$

where

r_e = geocentric radius at the entry altitude

$\tilde{r}_a = r_a / r_e$

$\tilde{r}_p = r_p / r_e$

γ_e = flight path angle at entry

r_a = apogee radius of the initial elliptical orbit

r_p = perigee radius of the initial elliptical orbit

μ = gravitational constant of the Earth

The true anomaly at entry can be determined from the following series of equations:

$$\sin \theta_e = \frac{\dot{r}}{e_d} \sqrt{\frac{a_d (1 - e_d^2)}{\mu}}$$

$$\cos \theta_e = \frac{a_d (1 - e_d^2)}{e_d r_e} - \frac{1}{e_d}$$

$$\theta_e = \tan^{-1}(\sin \theta_e, \cos \theta_e)$$

where

e_d = eccentricity of deorbit trajectory

a_d = semimajor axis of deorbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu [2a_d r_e - r_e^2 - a_d^2 (1 - e_d^2)]}{a_d r_e^2}}$$

and the inverse tangent is a four quadrant operation.

The time of flight between perigee and the entry true anomaly θ_e is given by:

$$t(\theta_e) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1 - e_d}{1 + e_d}} \tan \frac{\theta_e}{2} \right\} - \frac{e_d \sqrt{1 - e_d^2} \sin \theta_e}{1 + e_d \cos \theta_e} \right]$$

In this equation τ is the orbital period of the de-orbit trajectory.

Therefore, the flight time between the de-orbit impulse time and entry is given by

$$\Delta t = t(\theta_e) - t(180^\circ) = t(\theta_e) - \frac{\tau}{2}$$

Finally, the speed at reentry V_e can be determined from

$$V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$$

Please note that these equations are also valid for the case of de-orbit from an initial circular orbit as described in the previous `cdeorbit.m` script.

The following is a typical user interaction with this script.

```
program edeorbit

< single impulse deorbit from elliptical orbits >

please input the perigee altitude (kilometers)
? 285.798

please input the apogee altitude (kilometers)
? 35785.922

please input the entry altitude (kilometers)
? 111.252

please input the entry flight path angle (degrees)
? -4
```

The following is the script output created for this example.

```
program edeorbit

< single impulse deorbit from elliptical orbits >

initial orbit

perigee altitude      285.798000  kilometers
apogee altitude      35785.922000  kilometers
semimajor axis       24414.000000  kilometers
eccentricity          0.727044
entry altitude        111.252000  kilometers
entry fpa             -4.000000   degrees

entry trajectory

semimajor axis        24308.08290588 kilometers
eccentricity          0.73456961
```

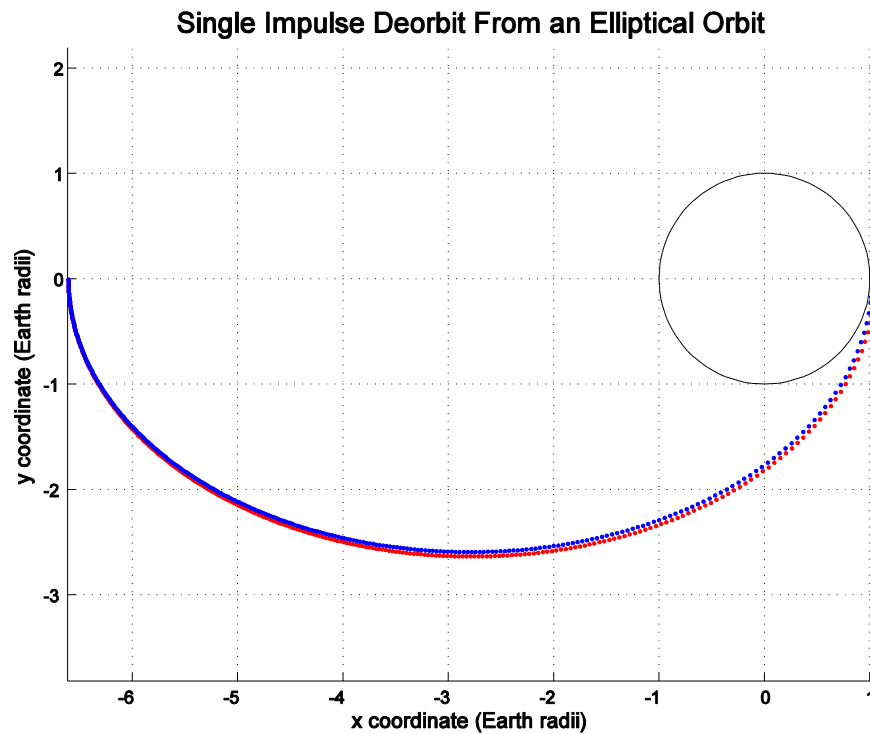
Orbital Mechanics with MATLAB

perigee altitude	73.96381175	kilometers
apogee altitude	35785.92200000	kilometers
entry true anomaly	350.55084585	degrees
entry velocity	10317.40933180	meters/second
entry fpa	-4.00000000	degrees
impulse-to-entry time	312.58844372	minutes
deorbit delta-v	22.29796787	meters/second

The software will also calculate and display the entry velocity and flight path angle relative to a rotating spherical Earth. The following is the relative flight path information for this example.

relative flight path coordinates		
entry velocity	9845.40345708	meters/second
entry fpa	-4.19210209	degrees

This MATLAB script will also graphically display the initial elliptic orbit and the de-orbit trajectory. The graphic display for this example is as follows where the red dots represent the original elliptical orbit and the blue dots represent the de-orbit trajectory, both at one minute intervals. The black circle is the surface of a spherical Earth and the distances are in Earth radii.



Program deorbit_ocs

Finite-Burn De-orbit Trajectory Optimization

This document is the user's manual for a Fortran computer program called `deorbit_ocs` that uses the *Sparse Optimization Suite* distributed by [Applied Mathematical Analysis](#) to solve the de-orbit trajectory optimization problem. The software models the trajectory as a single, finite-burn propulsive maneuver followed by a user-defined, time-bounded final coast phase. This computer attempts to maximize the final spacecraft mass. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass.

The important features of this scientific simulation are as follows:

- single, continuous thrust orbital maneuver
- variable inertial attitude steering
- constant propulsive thrust magnitude
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified final coast phase and entry interface (EI) target conditions

The *Sparse Optimization 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 *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010 (www.siam.org).

The `deorbit_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 *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `deorbit_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:

```
deorbit_ocs cleo2ei.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 deorbit_ocs      *
*                               *
* deorbit trajectory optimization *
*                               *
*      May 10, 2012             *
*****

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.

The screen output created by the `deorbit_ocs` computer program can be re-directed to a text file with a command line similar to

```
deorbit_ocs cleo2ei.in >cleo2ei.txt
```

To create a DOS command window in Windows 7, select **start**, then **All Programs**, then **Accessories** and finally **Command Prompt**. The size, font and other characteristics of the screen can be controlled by the user with the **c:** icon in the upper left corner of the window. To log into the subdirectory created during the installation of the Fortran executable and support files, type **root:** and then **cd subdirectory** from the DOS command line where **root** is the name of the root directory, usually **c:**, and subdirectory is the name of the subdirectory created by the user.

The DOS command line prompt looks similar to **C:\deorbit_ocs>_**.

Input File Format and Contents

The `deorbit_ocs` software is “data-driven” by a user-created text file. This text file should be simple ASCII format with no special characters.

The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in *times italic* font. This example attempts to optimize the maneuver required to de-orbit a spacecraft from a circular Earth orbit (LEO) to typical entry interface (EI) conditions.

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.

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.

```
*****
** de-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program deorbit_ocs
** cleo2ei.in - May 11, 2012
*****
```

The first two inputs define the calendar date and Universal Coordinated Time (UTC) of the de-orbit maneuver. Please be sure to provide all four digits of the calendar year.

```
maneuver calendar date (month, day, year)
-----
3, 18, 2010

maneuver UTC (hour, minute, second)
-----
12, 30, 45.875
```

The next three inputs define the initial mass prior to the propulsive maneuver, and the thrust magnitude and specific impulse of the upper stage or spacecraft propulsion system.

```
initial spacecraft mass (kilograms)
8000.0

thrust magnitude (newtons)
2000.0

specific impulse (seconds)
325.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

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

The next two numeric inputs define either the user's initial guess for the delta-v magnitude or the maneuver duration, and should be consistent with the previous input.

```
initial guess for delta-v (meters/second)
150.0

initial guess for thrust duration (seconds)
700.0
```

The next two inputs define the lower and upper bounds for the thrust duration. These inputs are required for either type of propulsive initial guess.

```
lower bound for thrust duration (seconds)
10.0

upper bound for thrust duration (seconds)
1000.0
```

The next section of the input data file lets the user define the characteristics of a final coast phase that follows the propulsive maneuver. These three inputs define an initial guess for the coast duration as well as lower and upper bounds on the coast duration. All inputs are in minutes.

```
*****
* coast maneuver *
*****

initial guess for coast duration (minutes)
30.0

lower bound for coast duration (minutes)
20.0

upper bound for coast duration (minutes)
40.0
```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
6878.14d0

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
100.0

right ascension of the ascending node (degrees)
220.0d0

true anomaly (degrees)
180.0
```

This next integer input allows the user to define the type of initial orbit constraints to use during the simulation.

```
*****
* initial orbit constraint options *
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----
3
```

The next series of inputs define the entry interface (EI) mission constraints. These elements are defined with respect to the relative coordinate system. Please note the proper units for each mission constraint. Important note: To disregard a mission constraint, input the value 1.0d99 for that constraint.

```
*****
* entry interface constraints (set to 1.0d99 to ignore) *
*****
```

```

geodetic altitude (kilometers)
121.92d0

relative flight path angle (degrees)
-2.0d0

geodetic latitude (degrees)
-14.0

east longitude (degrees)
181.0

relative azimuth (degrees)
1.0d99

relative velocity (meters/second)
1.0d99

```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```

*****
* type of gravity model *
-----
1 = spherical Earth
2 = oblate gravity model
-----
2

```

This next input specifies the type of solution data file to create.

```

*****
* type of comma-delimited solution data file *
*****
1 = OCS-defined nodes
2 = user-defined nodes
3 = user-defined step size
-----
2

```

For options 2 or 3, this 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
100

```

The name of the comma-separated-variable solution data file is defined in this next line.

```

name of solution output file
cleo2ei.csv

```

The next series of program inputs are algorithm control options and parameters for the Sparse Optimization Suite. 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 input defines the relative error in the objective function.

```

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.

```

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
10000

```

The level of output from the Sparse Optimization Suite NLP algorithm is controlled with the following integer input.

```

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

```

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

```

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

```

Optimal control solution

The following is the optimal control solution for this example. The output includes the time and orbital characteristics at the beginning and end of the propulsive maneuver. This example optimizes the finite-burn maneuver required to transfer from a circular low Earth orbit (LEO) to an entry interface defined by a relative flight path angle, geodetic altitude and latitude, and a geographic east longitude. Appendix B contains a brief summary of this information.

```

program deorbit_ocs
=====

input data file ==> cleo2ei.in

oblate earth gravity model

initial epoch
-----

calendar date          March 18, 2010

UTC time                12:30:45.875

-----
beginning of finite burn
-----

mission elapsed time    00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.687814000000D+04  0.256247074642D-15  0.285000000000D+02  0.000000000000D+00

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.220000000000D+03  0.199001209386D+03  0.199001209386D+03  0.946163624135D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
0.371682044841D+04  0.568790088946D+04  -1.106856870886D+04  0.687814000000D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-1.596468785281D+01  0.325246126525D+01  -1.343449740362D+01  0.761260651018D+01

-----

```

end of finite burn

mission elapsed time 00:09:39.318

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.662834854917D+04	0.373976098013D-01	0.286374887088D+02	0.314330866858D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.220229462304D+03	0.203823970508D+03	0.235257057194D+03	0.895092125778D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.210335480056D+03	0.629617719700D+04	-.269907170370D+04	0.685354480337D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.711928147802D+01	-.122317403027D+01	-.200086582577D+01	0.749558453521D+01

The following program output is the final spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v.

final mass	7636.46768849416	kilograms
propellant mass	363.532311505839	kilograms
thrust duration	579.318048180925	seconds
	9.65530080301541	minutes
delta-v	148.223366147974	meters/second

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

This section of the numeric results summarizes the time and orbital conditions at the beginning and end of the final coast.

beginning of coast maneuver

mission elapsed time 00:09:39.318

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.662834854917D+04	0.373976098013D-01	0.286374887088D+02	0.314330866858D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.220229462304D+03	0.203823970508D+03	0.235257057194D+03	0.895092125778D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.210335480056D+03	0.629617719700D+04	-.269907170370D+04	0.685354480337D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.711928147802D+01	-.122317403027D+01	-.200086582577D+01	0.749558453521D+01

end of coast maneuver

mission elapsed time 00:33:37.138

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.663096280332D+04	0.386110555799D-01	0.286533250810D+02	0.307476929735D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.220038827173D+03	0.299159749780D+03	0.329907442753D+03	0.895621721271D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.614443018434D+04	-.142801717018D+04	-.156247691255D+04	0.649881446349D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.107390419948D+01	-.708739128858D+01	0.334249561358D+01	0.790927698554D+01
coast duration	1437.81997871958	seconds	
	23.9636663119930	minutes	
	0.399394438533216	hours	

This final section of the numeric results summarizes both the relative and inertial flight conditions at the entry interface.

```
relative flight path coordinates at entry interface
-----
geodetic altitude      121.9199999999994      kilometers
geodetic latitude     -13.9999999999992      degrees
east longitude         180.999999897136      degrees
flight path angle     -2.000000000000018      degrees
azimuth                63.1924684582462      degrees
velocity               7496.23024105710      meters/second

inertial flight path coordinates at entry interface
-----
right ascension        193.083753391143      degrees
flight path angle     -1.89551468207218      degrees
azimuth                64.6963141835643      degrees
velocity               7909.27698553550      meters/second
```

Verification of the optimal control solution

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

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

```
=====
verification of optimal control solution
=====
```

beginning of coast maneuver

mission elapsed time 00:09:39.318

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.662834854907D+04	0.373976097701D-01	0.286374887086D+02	0.314330866910D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.220229462304D+03	0.203823970504D+03	0.235257057195D+03	0.895092125757D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.210335480218D+03	0.629617719672D+04	-.269907170359D+04	0.685354480309D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.711928147827D+01	-.122317403037D+01	-.200086582582D+01	0.749558453547D+01

end of coast maneuver

mission elapsed time 00:33:37.138

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.663096280315D+04	0.386110556971D-01	0.286533250796D+02	0.307476930541D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.220038827181D+03	0.299159749691D+03	0.329907442745D+03	0.895621721237D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.614443018404D+04	-.142801717004D+04	-.156247691279D+04	0.649881446323D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.107390420036D+01	-.708739128871D+01	0.334249561350D+01	0.790927698574D+01

relative flight path coordinates at entry interface

geodetic altitude	121.919999748011	kilometers
geodetic latitude	-14.0000000027976	degrees
east longitude	180.999999896537	degrees
flight path angle	-2.00000000777524	degrees
azimuth	63.1924684615364	degrees
velocity	7496.23024127568	meters/second

inertial flight path coordinates at entry interface

right ascension	193.083753390544	degrees
flight path angle	-1.89551468944646	degrees
azimuth	64.6963141865741	degrees
velocity	7909.27698574230	meters/second

mass and propulsive properties

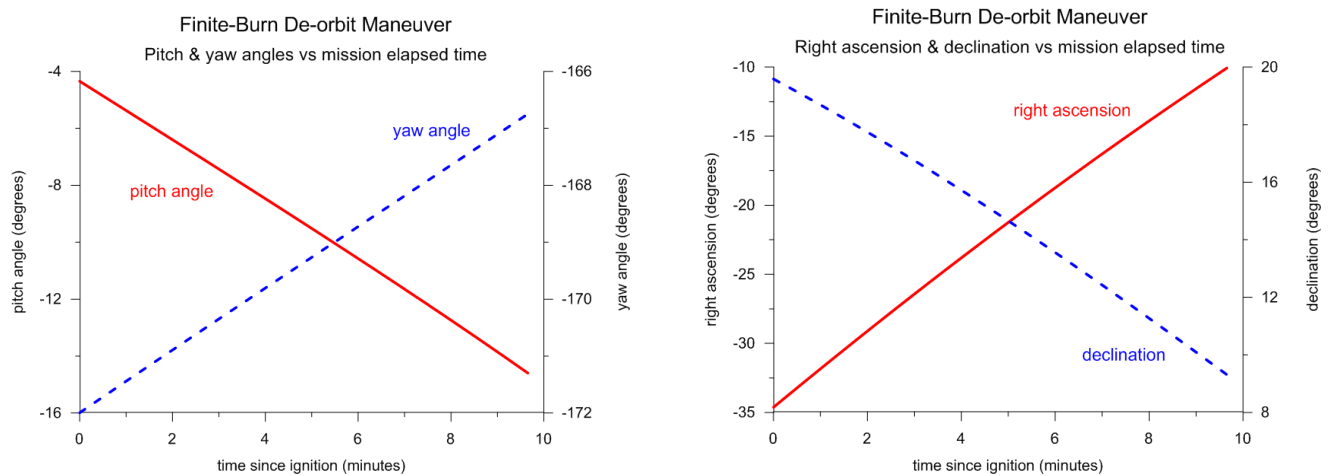
```

-----
final mass          7636.46768849181    kilograms
propellant mass     363.532311508195    kilograms
thrust duration     579.318048180925    seconds
                   9.65530080301541    minutes
delta-v            148.223363619416    meters/second

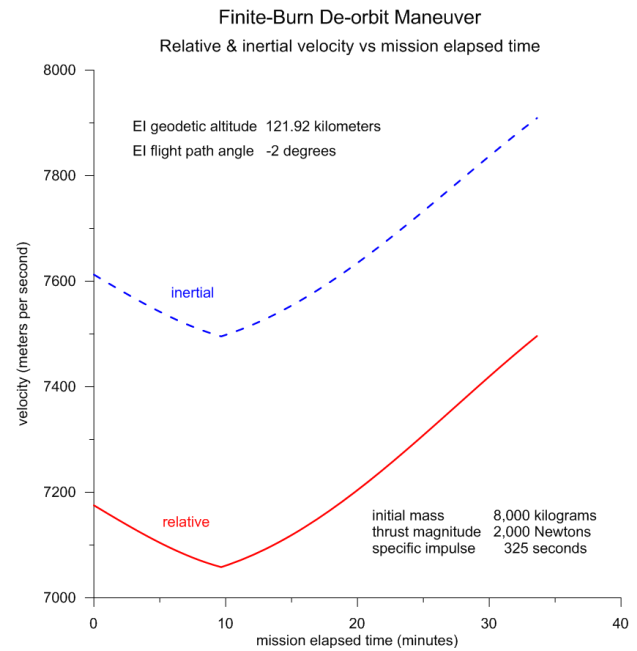
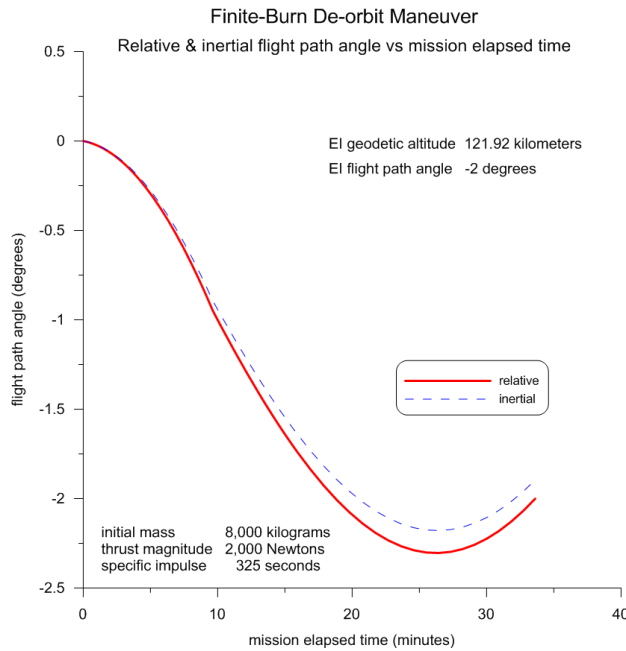
```

In addition to the user-defined solution output file, the `deorbit_ocs` computer program will create a comma-separated-variable data file named `maneuver.csv`. This data file contains the information described in Appendix B starting at ignition and ending at burnout of the propulsive maneuver.

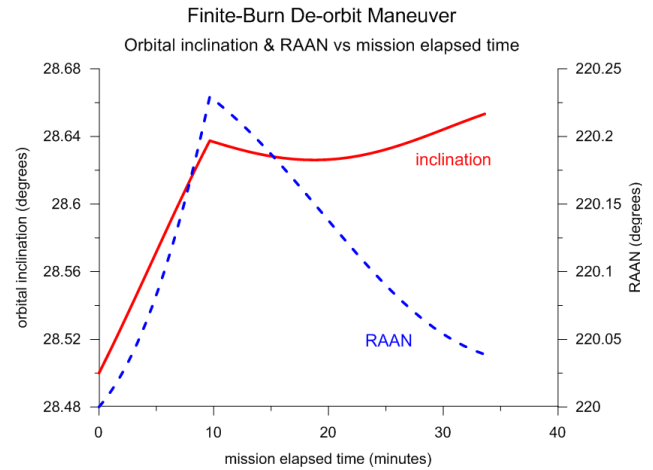
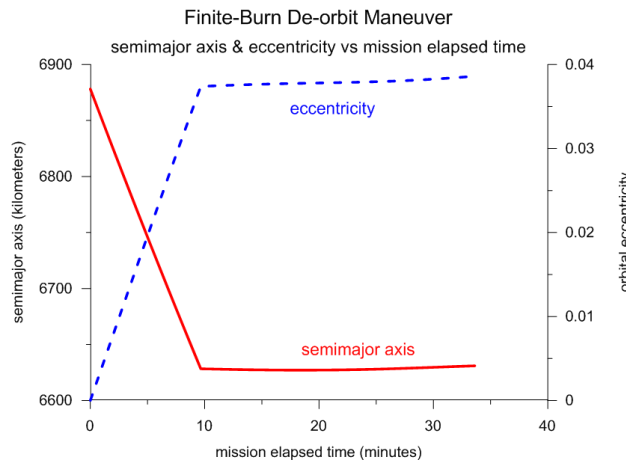
The following are graphic displays of several important flight conditions for this example. The first two images illustrate the behavior of the orbit-relative pitch and yaw angles, and the inertial right ascension and declination angles of the unit thrust vector during the de-orbit maneuver.



These two plots summarize the relative and inertial flight path angles and velocities.



The next two plots illustrate the behavior of the semimajor axis, orbital eccentricity, orbital inclination and right ascension of the ascending node (RAAN) during the simulation.



Creating an initial guess

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

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

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

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

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

In these equations

- m_i = initial mass
- m_f = final mass
- m_p = propellant mass
- V_{ex} = exhaust velocity = $g I_{sp}$
- I_{sp} = specific impulse
- ΔV = impulsive velocity increment
- F = thrust
- g = acceleration of gravity

The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting opposite to the flight path, the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = [0 \quad -1 \quad 0]^T$. Please note that this type of steering method creates a *coplanar* initial guess.

The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* software to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the program to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

An initial guess for the modified equinoctial orbital elements at the beginning of the final coast phase are determined by numerically integrating the equations of motion using the initial orbital elements, spacecraft mass, and propulsive characteristics provided by the user.

An algorithm for estimating the impulse de-orbit delta-v from initial circular orbits is explained in Appendix C. A numerical method for calculating the impulsive maneuver required for de-orbiting from an initial elliptical Earth orbit can be found in Appendix D.

Problem setup

This part of the user's manual provides details about the software implementation within `deorbit_ocs`. It defines such things as point and path constraints (boundary conditions), bounds on the dynamic variables, and the performance index or objective function.

(1) Point functions – initial orbit constraints

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

- 1) constrain semimajor axis, eccentricity and inclination

- 2) constrain all initial orbital elements
- 3) option 2 with unconstrained true longitude

For option 1, the initial orbit inclination is constrained by enforcing

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

where i is the initial orbit inclination.

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

$$f = 0 \quad \text{and} \quad g = 0$$

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

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

is enforced, where e is the initial orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the initial modified equinoctial orbital elements as follows:

$$p_L = p_U = p_i$$

$$f_L = f_U = f_i$$

$$g_L = g_U = g_i$$

$$h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the initial true longitude unbounded. In optimal control terminology, these derived 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 or termination of the propulsive maneuver. This is simply

$$J = m_f$$

The value of the `maxmin` indicator in the *Sparse Optimization Suite* algorithm tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

(3) Path constraint – unit thrust vector scalar magnitude

For a *variable steering* trajectory, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows:

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

(4) Point functions – entry interface mission constraints

The entry interface mission constraints are relative flight path coordinates defined with respect to a rotating Earth. They are calculated from the inertial spacecraft coordinates at the entry interface using the algorithm described in Appendix E.

This set of possible constraints consists of the following elements;

$$h_p - h_u = 0 \rightarrow \text{geodetic altitude}$$

$$\gamma_p - \gamma_u = 0 \rightarrow \text{relative flight path angle}$$

$$\phi_p - \phi_u = 0 \rightarrow \text{geodetic latitude}$$

$$\lambda_p - \lambda_u = 0 \rightarrow \text{east longitude}$$

$$\psi_p - \psi_u = 0 \rightarrow \text{relative azimuth}$$

$$v_p - v_u = 0 \rightarrow \text{relative velocity}$$

where the p subscript refers to values predicted by the software and the u subscript are the values defined by the user.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05m_{sc_i} \leq m_{sc} \leq 1.05m_{sc_i}$$

$$100p_f \leq p \leq 0.8p_i$$

$$-1 \leq f \leq +1$$

$$-1 \leq g \leq +1$$

$$-1 \leq h \leq +1$$

$$-1 \leq k \leq +1$$

where m_{sc_i} is the initial spacecraft mass.

Finally, the three components of the unit thrust vector are constrained as follows

$$-1.1 \leq u_r \leq +1.1$$

$$-1.1 \leq u_t \leq +1.1$$

$$-1.1 \leq u_n \leq +1.1$$

Technical Discussion

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 summarized as follows:

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

true anomaly

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

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

position vector

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

velocity vector

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

where

$$\alpha^2 = h^2 - k^2 \quad s^2 = 1 + h^2 + k^2$$

$$r = \frac{p}{w} \quad w = 1 + f \cos L + g \sin L$$

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

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

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

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

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

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

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

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

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

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

where

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

and

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

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

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

where $\hat{\mathbf{i}}_r, \hat{\mathbf{i}}_t$ and $\hat{\mathbf{i}}_n$ are unit vectors in the radial, tangential and normal directions.

These unit vectors can be computed from the inertial position vector \mathbf{r} and velocity vector \mathbf{v} according to

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

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:

$$\begin{aligned}\Delta_{J_{2r}} &= -\frac{3\mu J_2 R_e^2}{2r^4} \left[1 - \frac{12(h \sin L - k \cos L)^2}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2t}} &= -\frac{12\mu J_2 R_e^2}{r^4} \left[\frac{(h \sin L - k \cos L)(h \cos L + k \sin L)}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2n}} &= -\frac{6\mu J_2 R_e^2}{r^4} \left[\frac{(1 - h^2 - k^2)(h \sin L - k \cos L)}{(1 + h^2 + k^2)^2} \right]\end{aligned}$$

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

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

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = [u_{T_r} \quad u_{T_t} \quad u_{T_n}]^T$ 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.*

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 and \dot{m} is the propellant flow rate.

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

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

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

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the “local horizontal” and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four quadrant operation.

The `deorbit_ocs` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. The following section summarizes the inertial-to/from-modified equinoctial coordinate transformations and the calculation of the inertial unit thrust vector in terms of right ascension and declination angles.

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

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

where

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

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 `deorbit_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

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

where the calculation for right ascension is a four quadrant inverse tangent operation.

Flight path coordinates

The mathematical relationship between flight path and inertial coordinates is explained in Appendix E.

Geodetic coordinates

An algorithm for converting from geocentric declination and radius to geodetic altitude and latitude is described in Appendix F. It uses a series solution involving the flattening factor of the Earth.

Algorithm Resources

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“Survey of Numerical Methods for Trajectory Optimization”, John T. Betts, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998.

“Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, 20-22 August 1990.

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.

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

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

“Optimum Deboost Altitude for Specified Atmospheric Entry Angle”, Jerome M. Baker, Bruce E. Baxter, and Paul D. Arthur, *AIAA Journal*, Vol. 1, No. 7, July 1963.

“Deboost from Circular Orbits”, A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, Jul-Aug., 1966.

Hypersonic and Planetary Entry Flight Mechanics, Vinh, Busemann and Culp, The University of Michigan Press, 1980.

“On Autonomous Optimal Deorbit Guidance”, Morgan C. Baldwin, Binfeng Pan and Ping Lu, AIAA 2009-5667, AIAA Guidance, Navigation, and Control Conference, August 10-13, 2009.

“Autonomous Optimal Deorbit Targeting”, Donald J. Jezewski, AAS 91-136, AAS/AIAA Spaceflight Mechanics Meeting, February 11-13, 1991.

“Analysis of the Accuracy of Ballistic Descent from a Circular Circumterrestrial Orbit”, Yu. G. Sikharulidze and A. N. Korchagin, *Cosmic Research*, Vol. 40, No. 1, 2002, pp.75-87.

“Geometric Theory of Optimum Disorbit Problems”, A. Busemann and N. X. Vinh, NASA CR-750, April 1967.

“Nearly Circular Transfer Trajectories for Descending Satellites”, George M. Low, NASA Technical Report R-3, 1959.

APPENDIX A

Example De-orbit from an Elliptical Earth Orbit

This appendix illustrates the characteristics for a typical de-orbit from a highly elliptical Earth orbit (HEO). For this example, the entry interface constraints consist of the geodetic altitude and relative flight path angle. An estimate for the delta-v required for this example and the coast time were determined using the algorithm described in Appendix D.

The main portion of the simulation definition file for this example is as follows:

```
*****
** de-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program deorbit_ocs
** heo2ei.in - May 10, 2012
*****

maneuver calendar date (month, day, year)
-----
3, 18, 2010

maneuver UTC (hour, minute, second)
-----
12, 30, 45.875

initial spacecraft mass (kilograms)
8000.0

thrust magnitude (newtons)
1000.0

specific impulse (seconds)
325.0

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

initial guess for delta-v (meters/second)
35.0d0

initial guess for thrust duration (seconds)
700.0

lower bound for thrust duration (seconds)
1.0

upper bound for thrust duration (seconds)
1000.0

*****
* coast maneuver *
*****

initial guess for coast duration (minutes)
300.0
```



```

lower bound for coast duration (minutes; > 0)
200.0

upper bound for coast duration (minutes)
500.0

*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
24414.0d0

orbital eccentricity (non-dimensional)
0.727044

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
270.0

right ascension of the ascending node (degrees)
220.0d0

true anomaly (degrees)
180.0d0

*****
* initial orbit constraint options *
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----
3

*****
* entry interface constraints (set to 1.0d99 to ignore) *
*****

geodetic altitude (kilometers)
121.92d0

relative flight path angle (degrees)
-2.0d0

geodetic latitude (degrees)
1.0d99

east longitude (degrees)
1.0d99

relative azimuth (degrees)
1.0d99

relative velocity (meters/second)
1.0d99

*****
* type of gravity model *
-----
1 = spherical Earth
2 = oblate gravity model
-----
2

```

The program output for this example is

```

program deorbit_ocs
=====

input data file ==> heo2ei.in

oblate earth gravity model

initial epoch
-----

calendar date          March 18, 2010
UTC time               12:30:45.875

-----
beginning of finite burn
-----

mission elapsed time    00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.244140000000D+05  0.727044000000D+00  0.285000000000D+02  0.270000000000D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.220000000000D+03  0.179838597519D+03  0.898385975188D+02  0.632729583647D+03

      rx (km)      ry (km)      rz (km)      rmag (km)
0.237268343662D+05  -.284613210725D+05  0.201186544321D+05  0.421636066104D+05

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.123989347177D+01  0.102137536270D+01  0.791045192336D-02  0.160642647766D+01

-----
end of finite burn
-----

mission elapsed time    00:02:25.905

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.243269419485D+05  0.733224470367D+00  0.284999999981D+02  0.269999908966D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.220000048521D+03  0.180155324772D+03  0.901552337382D+02  0.629348220341D+03

      rx (km)      ry (km)      rz (km)      rmag (km)
0.239053722555D+05  -.283115456389D+05  0.201186693081D+05  0.421636252457D+05

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.120740294448D+01  0.103164365370D+01  -.769905249676D-02  0.158813405416D+01

final mass            7954.22115070684      kilograms
propellant mass       45.7788492931622      kilograms
thrust duration       145.904574520815      seconds
                     2.43174290868026      minutes
delta-v              18.2904541073263      meters/second

```

beginning of coast maneuver

mission elapsed time 00:02:25.905

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.243269419485D+05	0.733224470367D+00	0.284999999981D+02	0.269999908966D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.220000048521D+03	0.180155324772D+03	0.901552337382D+02	0.629348220341D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.239053722555D+05	-.283115456389D+05	0.201186693081D+05	0.421636252457D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.120740294448D+01	0.103164365370D+01	-.769905249677D-02	0.158813405416D+01

end of coast maneuver

mission elapsed time 05:15:04.421

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.243574533818D+05	0.733691641752D+00	0.284878735015D+02	0.270125093234D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.219924531651D+03	0.355464345323D+03	0.265589438557D+03	0.630532606936D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.326983940138D+04	0.468558859467D+04	-.308885964145D+04	0.649520161919D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.815487392014D+01	-.631067229242D+01	-.213830021530D+00	0.103136936504D+02

coast duration	18758.5159416022	seconds
	312.641932360037	minutes
	5.21069887266728	hours

relative flight path coordinates at entry interface

geodetic altitude	121.919999999998	kilometers
geodetic latitude	-28.5540897239011	degrees
east longitude	42.2691975996772	degrees
flight path angle	-2.00000000000468	degrees
azimuth	92.4904131009538	degrees
velocity	9897.66361056145	meters/second

inertial flight path coordinates at entry interface

right ascension	124.909273063683	degrees
-----------------	------------------	---------

flight path angle	-1.91929389968213	degrees
azimuth	92.3897809434936	degrees
velocity	10313.6936503760	meters/second

Here are the verification results for this example.

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

beginning of coast maneuver

mission elapsed time 00:02:25.905

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.243269419484D+05	0.733224470367D+00	0.284999999981D+02	0.269999908963D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.220000048522D+03	0.180155324775D+03	0.901552337371D+02	0.629348220339D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.239053722554D+05	-.283115456388D+05	0.201186693081D+05	0.421636252456D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.120740294437D+01	0.103164365384D+01	-.769905257556D-02	0.158813405416D+01

end of coast maneuver

mission elapsed time 05:15:04.421

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.243574533952D+05	0.733691641944D+00	0.284878734954D+02	0.270125093237D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.219924531681D+03	0.355464335541D+03	0.265589428778D+03	0.630532607457D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.326983852645D+04	0.468558927113D+04	-.308885961732D+04	0.649520165524D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.815487442942D+01	-.631067156466D+01	-.213830504622D+00	0.103136936178D+02

relative flight path coordinates at entry interface

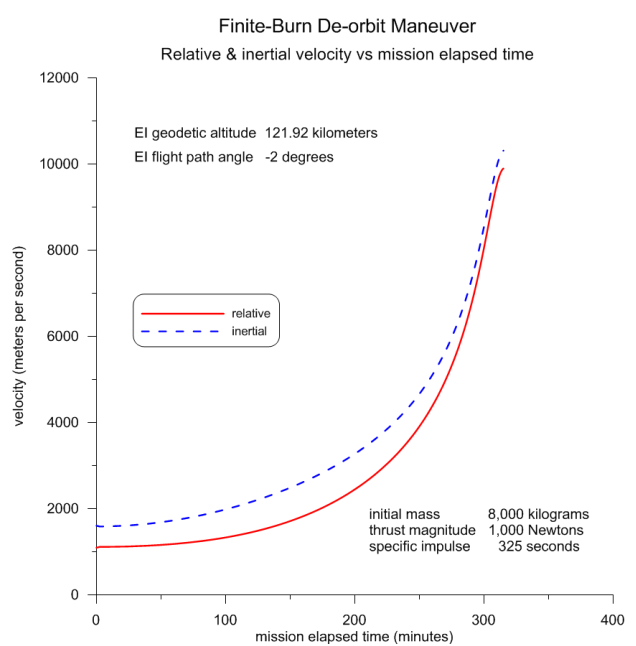
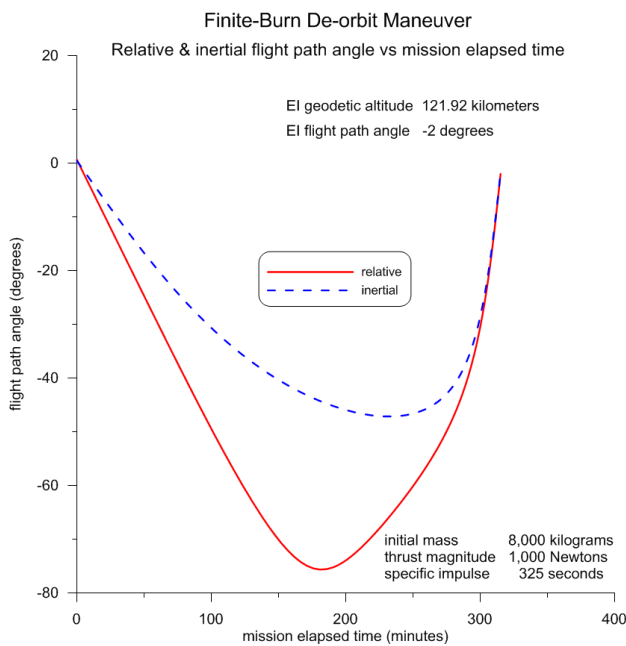
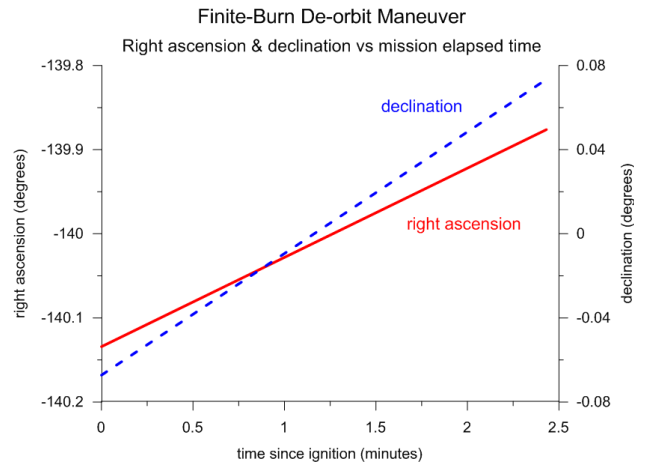
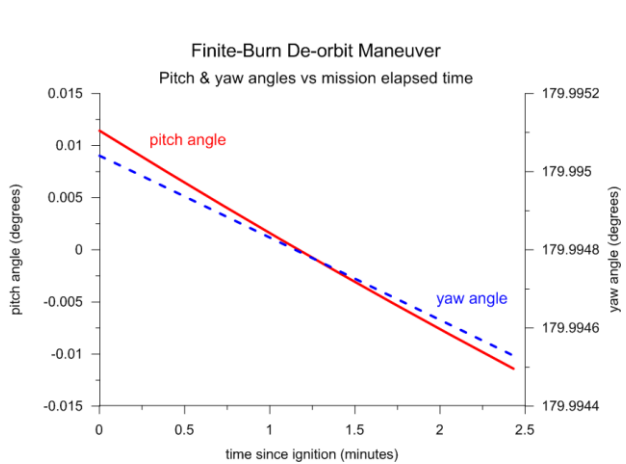
geodetic altitude	121.920035928964	kilometers
geodetic latitude	-28.5540893076028	degrees
east longitude	42.2691865228324	degrees
flight path angle	-2.00000431339059	degrees
azimuth	92.4904186061209	degrees
velocity	9897.66357674878	meters/second

inertial flight path coordinates at entry interface

right ascension	124.909261986838	degrees
flight path angle	-1.91929803838563	degrees
azimuth	92.3897862248354	degrees
velocity	10313.6936177802	meters/second

mass and propulsive properties

final mass	7954.22115071184	kilograms
propellant mass	45.7788492881609	kilograms
thrust duration	145.904574520815	seconds
	2.43174290868026	minutes
delta-v	18.2904541154479	meters/second



APPENDIX B

Contents of the Simulation Summary and CSV Files

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

The simulation summary screen display contains the following information:

mission elapsed time = simulation time (hh:mm:ss.sss)

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 spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

geodetic altitude = geodetic altitude in kilometers

geodetic latitude = geodetic latitude in degrees)

east longitude = east longitude in degrees

flight path angle = relative or inertial flight path angle in degrees

azimuth = relative or inertial azimuth angle in degrees

velocity = relative or inertial velocity in meters per second

final mass = final spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

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

```
time (sec) = mission elapsed time in seconds
time (min) = mission elapsed time in minutes
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
period (min) = orbital period in minutes
mass (kg) = spacecraft mass in kilograms
thracc (mps/s) = thrust acceleration in meters/second**2
perigee altitude = perigee altitude in kilometers
apogee altitude = apogee altitude in kilometers
geodetic altitude (km) = geodetic altitude in kilometers
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 eci unit thrust vector
ut-eci-y = y-component of eci unit thrust vector
ut-eci-z = z-component of eci unit thrust vector
semi-parameter = orbital semiparameter in kilometers
f equinoctial element = modified equinoctial orbital element
g equinoctial element = modified equinoctial orbital element
h equinoctial element = modified equinoctial orbital element
k equinoctial element = modified equinoctial orbital element
true longitude = true longitude in degrees
rx (km) = x-component of the spacecraft's position vector in kilometers
ry (km) = y-component of the spacecraft's position vector in kilometers
rz (km) = z-component of the spacecraft's position vector in kilometers
```

rmag (km) = magnitude of spacecraft's position vector in kilometers

vx (km) = x-component of the spacecraft's velocity vector in kilometers/second

vy (km) = y-component of the spacecraft's velocity vector in kilometers/second

vz (km) = z-component of the spacecraft's velocity vector in kilometers/second

vmag (km) = magnitude of spacecraft's velocity vector in kilometers/second

rasc (deg) = inertial right ascension of the unit thrust vector in degrees

decl (deg) = inertial declination of the unit thrust vector in degrees

yaw (deg) = out-of-plane yaw angle of the unit thrust vector in degrees

pitch (deg) = in-plane pitch angle of the unit thrust vector in degrees

declination (deg) = geocentric declination in degrees

geodetic lat (deg) = geodetic latitude in degrees

longitude (deg) = east longitude in degrees

azimuth (deg) = relative azimuth in degrees

vmagr (mps) = relative velocity in meters per second

fpar (deg) = relative flight path angle in degrees

fpai (deg) = inertial flight path angle in degrees

deltav (mps) = accumulative delta-v in meters per second

APPENDIX C

De-orbit from a Circular Earth Orbit

This appendix summarizes an algorithm can be used to compute the impulsive delta-v required to de-orbit a spacecraft initially in a circular Earth orbit. The “targets” at the entry interface consist of the altitude and flight path angle.

The scalar magnitude of the single impulsive maneuver required to de-orbit a spacecraft from an initial circular orbit can be determined from the following expression

$$\Delta V = V_{c_e} \sqrt{\frac{1}{\tilde{r}}} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\} = V_{c_i} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\}$$

where

$$\tilde{r} = \frac{h_i + r_{eq}}{h_e + r_{eq}} = \frac{r_i}{r_e} = \text{radius ratio}$$

$$V_{c_e} = \sqrt{\frac{\mu}{(h_e + r_{eq})}} = \sqrt{\frac{\mu}{r_e}} = \text{local circular velocity at entry interface}$$

$$V_{c_i} = \sqrt{\frac{\mu}{(h_i + r_{eq})}} = \sqrt{\frac{\mu}{r_i}} = \text{local circular velocity of initial circular orbit}$$

γ_e = flight path angle at entry interface

h_i = altitude of initial circular orbit

h_e = altitude at entry interface

r_i = radius of initial circular orbit

r_e = radius at entry interface

r_{eq} = Earth equatorial radius

μ = Earth gravitational constant

This algorithm is described in the technical article, “Deboost from Circular Orbits”, A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, Jul-Aug., 1966. Additional information can be found in Chapter 5 of *Hypersonic and Planetary Entry Flight Mechanics* by Vinh, Busemann and Culp, The University of Michigan Press.

The true anomaly on the de-orbit trajectory at the entry interface θ_e can be determined from the following two equations

$$\sin \theta_e = \frac{\dot{r}}{e_d} \sqrt{\frac{a_d(1-e_d^2)}{\mu}}$$

$$\cos \theta_e = \frac{a_d(1-e_d^2)}{e_d r_e} - \frac{1}{e_d}$$

and the following four quadrant inverse tangent operation

$$\theta_e = \tan^{-1}(\sin \theta_e, \cos \theta_e)$$

where

e_d = eccentricity of the de-orbit trajectory

a_d = semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu[2a_d r_e - r_e^2 - a_d^2(1-e_d^2)]}{a_d r_e^2}}$$

The elapsed time-of-flight between perigee of the de-orbit trajectory and the entry true anomaly θ_e is given by

$$t(\theta_e) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1-e_d}{1+e_d}} \tan \frac{\theta_e}{2} \right\} - \frac{e_d \sqrt{1-e_d^2} \sin \theta_e}{1+e_d \cos \theta_e} \right]$$

In this equation τ is the Keplerian orbital period of the de-orbit trajectory and is equal to $2\pi\sqrt{a_d^3/\mu}$.

Therefore, the flight time between the de-orbit impulse and entry interface is given by

$$\Delta t = t(\theta_e) - t(180^\circ) = t(\theta_e) - \frac{\tau}{2}$$

Finally, the orbital speed at the entry interface V_e can be determined from

$$V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$$

APPENDIX D

De-orbit from an Elliptical Earth Orbit

This appendix summarizes an algorithm can be used to compute the impulsive delta-v required to de-orbit a spacecraft initially in an elliptical Earth orbit.

The scalar magnitude of the impulsive delta-v for de-orbit from an initial elliptical orbit is given by

$$\Delta V = \sqrt{\frac{\mu}{r_e}} \left(\sqrt{\frac{2\tilde{r}_p}{\tilde{r}_a(\tilde{r}_a + \tilde{r}_p)}} - \sqrt{\frac{2(\tilde{r}_a - 1)}{\tilde{r}_a(\tilde{r}_a^2 - \cos^2 \gamma_e)}} \cos \gamma_e \right)$$

where

r_e = geocentric radius at the entry altitude

$\tilde{r}_a = r_a / r_e$

$\tilde{r}_p = r_p / r_e$

γ_e = flight path angle at entry

r_a = apogee radius of the initial elliptical orbit

r_p = perigee radius of the initial elliptical orbit

μ = gravitational constant of the Earth

The true anomaly at entry can be determined from the following series of equations.

$$\sin \theta_e = \frac{\dot{r}}{e_d} \sqrt{\frac{a_d(1 - e_d^2)}{\mu}}$$

$$\cos \theta_e = \frac{a_d(1 - e_d^2)}{e_d r_e} - \frac{1}{e_d}$$

$$\theta_e = \tan^{-1}(\sin \theta_e, \cos \theta_e)$$

where

e_d = eccentricity of the de-orbit trajectory

a_d = semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu[2a_d r_e - r_e^2 - a_d^2(1 - e_d^2)]}{a_d r_e^2}}$$

The time-of-flight between perigee and the entry interface true anomaly θ_e is given by

$$t(\theta_e) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1-e_d}{1+e_d}} \tan \frac{\theta_e}{2} \right\} - \frac{e_d \sqrt{1-e_d^2} \sin \theta_e}{1+e_d \cos \theta_e} \right]$$

In this equation, τ is the orbital period of the de-orbit trajectory.

Therefore, the flight time between the de-orbit impulse time and the entry interface is given by

$$\Delta t = t(\theta_e) - t(180^\circ) = t(\theta_e) - \frac{\tau}{2}$$

Finally, the speed at the entry interface V_e can be determined from

$$V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$$

APPENDIX E

Flight Path Coordinates

Relative flight path coordinates are defined with respect to a rotating spherical Earth. This set of coordinates consists of the following elements;

- r = geocentric radius
- V = speed
- γ = flight path angle
- δ = geocentric declination
- λ = geographic longitude (+ east)
- ψ = flight azimuth (+ clockwise from north)

Please note the sign and direction convention.

The following are several useful equations that summarize the relationships between inertial and relative flight path coordinates.

$$\begin{aligned}v_r \sin \gamma_r &= v_i \sin \gamma_i \\v_r \cos \gamma_r \cos \psi_r &= v_i \cos \gamma_i \cos \psi_i \\v_r \cos \gamma_r \sin \psi_r + \omega_e r \cos \delta &= v_i \cos \gamma_i \sin \psi_i\end{aligned}$$

where the r subscript denotes relative coordinates and the i subscript inertial coordinates.

The inertial speed can also be computed from the following expression

$$v_i = \sqrt{v^2 + 2vr\omega \cos \gamma \sin \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}$$

The inertial flight path angle can be computed from

$$\cos \gamma_i = \frac{\sqrt{v^2 \cos^2 \gamma + 2vr\omega \cos \gamma \cos \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}}{v^2 + 2vr\omega \cos \gamma \cos \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}$$

The inertial azimuth can be computed from

$$\cos \psi_i = \frac{v \cos \gamma \cos \psi + r\omega \cos \delta}{\sqrt{v^2 \cos^2 \gamma + 2vr\omega \cos \gamma \cos \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}}$$

where all coordinates on the right-hand-side of these equations are relative to a rotating Earth.

The transformation of an Earth-centered inertial (ECI) position vector \mathbf{r}_{ECI} to an Earth-centered fixed (ECF) position vector \mathbf{r}_{ECF} is given by the following vector-matrix operation

$$\mathbf{r}_{ECF} = [\mathbf{T}] \mathbf{r}_{ECI}$$

where the elements of the transformation matrix $[\mathbf{T}]$ are given by

$$[\mathbf{T}] = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and θ is the Greenwich apparent sidereal time at the moment of interest. Greenwich sidereal time is given by the following expression

$$\theta = \theta_{g0} + \omega_e t$$

where θ_{g0} is the Greenwich sidereal time at 0 hours UTC, ω_e is the inertial rotation rate of the Earth, and t is the elapsed time since 0 hours UTC.

Finally, the flight path coordinates are determined from the following set of equations

$$r = \sqrt{r_{ECF}^2 + r_{ECF_y}^2 + r_{ECF_z}^2}$$

$$v = \sqrt{v_{ECF}^2 + v_{ECF_y}^2 + v_{ECF_z}^2}$$

$$\lambda = \tan^{-1}(r_{ECF_y}, r_{ECF_x})$$

$$\delta = \sin^{-1}\left(\frac{r_{ECF_z}}{|\mathbf{r}_{ECF}|}\right)$$

$$\gamma = \sin^{-1}\left(-\frac{v_{R_z}}{|\mathbf{v}_R|}\right)$$

$$\psi = \tan^{-1}[v_{R_y}, v_{R_x}]$$

where

$$\mathbf{v}_R = \begin{bmatrix} -\sin \delta \cos \lambda & -\sin \delta \sin \lambda & \cos \delta \\ -\sin \lambda & \cos \lambda & 0 \\ -\cos \delta \cos \lambda & -\cos \delta \sin \lambda & -\sin \delta \end{bmatrix} \mathbf{v}_{ECF}$$

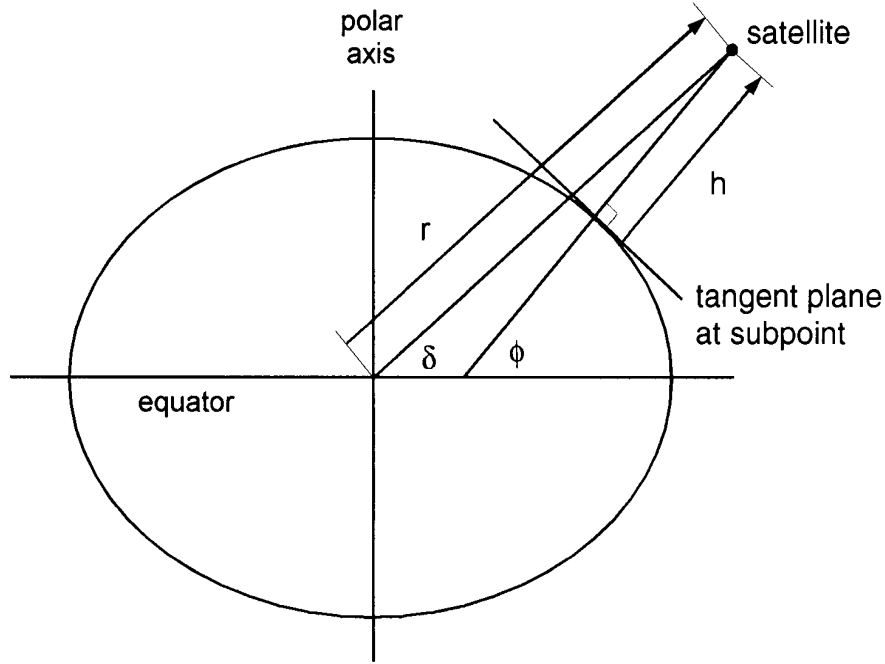
Please note that the two argument inverse tangent calculation is a four quadrant operation.

A set of *inertial* flight path coordinates can be determined from these equations by setting the value of Earth rotation to zero.

APPENDIX F

Geodetic Coordinates

The following diagram illustrates the geometric relationship between geocentric (radius and declination) and geodetic (altitude and latitude) coordinates.



In this diagram, δ is the geocentric declination, ϕ is the geodetic latitude, r is the geocentric distance, and h is the geodetic altitude.

The exact mathematical relationship between geocentric and geodetic coordinates is given by the following system of two nonlinear equations

$$(c + h) \cos \phi - r \cos \delta = 0$$

$$(s + h) \sin \phi - r \sin \delta = 0$$

where the geodetic constants c and s are given by

$$c = \frac{r_{eq}}{\sqrt{1 - (2f - f^2) \sin^2 \phi}}$$

$$s = c(1 - f)^2$$

In these equations, r_{eq} is the Earth equatorial radius (6378.14 kilometers) and f is the flattening factor for the Earth (1/298.257).

In this computer program, the geodetic latitude is determined using the following expression:

$$\phi = \delta + \left(\frac{\sin 2\delta}{\rho} \right) f + \left[\left(\frac{1}{\rho^2} - \frac{1}{4\rho} \right) \sin 4\delta \right] f^2$$

The geodetic altitude is calculated from

$$\hat{h} = (\hat{r} - 1) + \left\{ \left(\frac{1 - \cos 2\delta}{2} \right) f + \left[\left(\frac{1}{4\rho} - \frac{1}{16} \right) (1 - \cos 4\delta) \right] f^2 \right\}$$

In these equations, ρ is the geocentric distance of the satellite, $\hat{h} = h / r_{eq}$ and $\hat{r} = \rho / r_{eq}$.

This algorithm is based on “Derivation of Transformation Formulas Between Geocentric and Geodetic Coordinates for Nonzero Altitudes” by Sheila Ann T. Long, NASA TN D-7522, 1974.

A MATLAB Script for Optimal Single Impulse De-orbit from Earth Orbits

This document describes a MATLAB script named `deorbit_snopt` that can be used to compute the optimal impulsive maneuver required to de-orbit a spacecraft in a circular or elliptical Earth orbit. The user provides the classical orbital elements of the initial orbit along with geodetic altitude and relative flight path angle *targets* at the entry interface (EI).

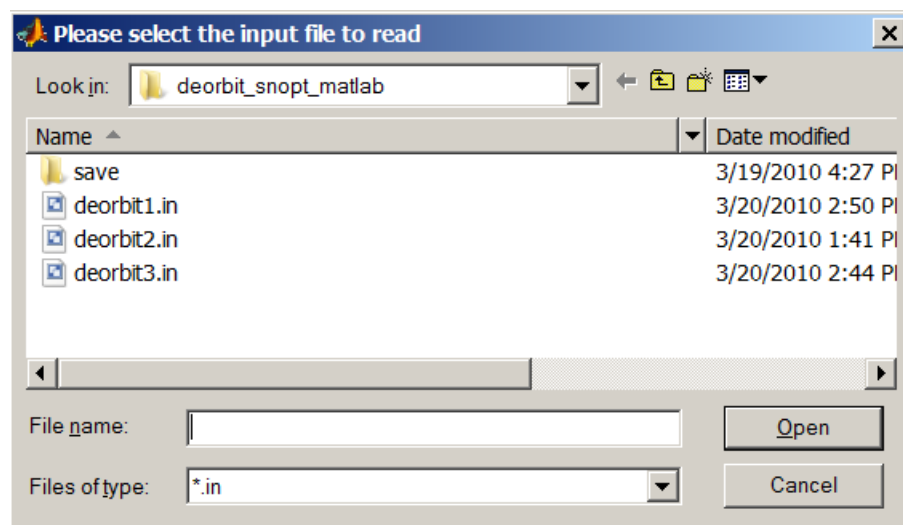
This script solves this maneuver optimization problem using a *simple shooting* method. During the solution process, the script numerically integrates the spacecraft equations of motion subject to the Earth's J_2 gravity coefficient. The numerical integration is performed using MATLAB's `ode45` function. The entry interface targets are computed with respect to an oblate, rotating Earth.

In this classic maneuver optimization problem, the maneuver true anomaly, the ECI components of the maneuver delta-v vector and the flight time from the maneuver to the entry interface are the *control variables*. The scalar magnitude of the de-orbit ΔV is the *objective function* or *performance index*, and the geodetic altitude and relative flight path angle at the entry interface are treated as *nonlinear equality constraints*. The algorithm uses an initial guess determined from the analytic de-orbit solution relative to a spherical, non-rotating Earth.

The `deorbit_snopt` script uses the SNOPT nonlinear programming algorithm to solve this orbital mechanics problem. MATLAB versions of SNOPT for several computer platforms can be requested or purchased at Professor Philip Gill's web site which is located at <http://scicomp.ucsd.edu/~peg/>. Professor Gill's web site also includes a PDF version of the software user's guide.

Interacting with the script

This MATLAB script is "data driven" by a text file created by the user. When the `deorbit_snopt` 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 `deorbit_snopt` compatible ASCII data file. The next section describes the format and typical contents of compatible input files.

Input data file

This section describes a typical input data file for the software. In the following discussion the actual input file contents are in *courier* font and all explanations are in *times italic* font. Typical user-provided values are in bold 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 five lines of any input file are reserved for user comments. These lines are ignored by the software. However the input file must begin with five and only five initial text lines.

```
*****
** impulsive de-orbit delta-v trajectory optimization
** de-orbit from initial circular orbit
** file ==> deorbit3.in   April 18, 2020
*****
```

The first input is the calendar date of the impulsive maneuver. Be sure to include all four digits of the calendar year.

```
calendar date at time of impulsive maneuver (month, day, year)
3, 18, 2010
```

The next input is the UTC time of the de-orbit maneuver.

```
UTC at time of impulsive maneuver (hours, minutes, seconds)
12, 30, 45.875
```

The next series of inputs define the classical orbital elements of the initial Earth orbit. Notice that the true anomaly is an initial guess for the location of the maneuver. The true anomaly initial guess for elliptical Earth orbits should be 180 degrees.

```
*****
orbital elements at time of impulsive maneuver
*****

semimajor axis (kilometers)7378.14
6878.14

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5

argument of perigee (degrees)
100.0

right ascension of the ascending node (degrees)
220.0

initial guess for true anomaly (degrees)
180.0
```

The software allows the user to specify lower and upper bounds for the optimal true anomaly of the maneuver. The algorithm enforces an inequality constraint on the true anomaly according to

$$\theta_L \leq \theta \leq \theta_U$$

where θ_L and θ_U are the user-defined lower and upper bounds, respectively.

The numerical values of these bounds are defined in the next two data items.

```
lower bound for true anomaly (degrees)
170.0
```

```
upper bound for true anomaly (degrees)
190.0
```

The final two items in the simulation file define the geodetic altitude and relative flight path angle targets at the entry interface.

```
*****
entry interface mission constraints
*****

geodetic altitude (kilometers)
121.92

relative flight path angle (degrees)
-2.0
```

Script examples

The following is the deorbit_snopt numerical solution for this example.

```
*****
single impulse deorbit from Earth orbits
*****

time and conditions prior to deorbit maneuver
-----

calendar date      18-Mar-2010
UTC time           12:30:45.875

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+6.878140000000000e+03 +0.000000000000000e+00 +2.850000000000000e+01 +1.000000000000000e+02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+2.200000000000000e+02 +1.900000000000000e+02 +2.900000000000000e+02 +9.46163624134673e+01

      rx (km)      ry (km)      rz (km)      rmag (km)
-5.45318321844679e+03 +2.83906883966968e+03 -3.08403806222734e+03 +6.878140000000000e+03

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-4.00911535387506e+00 -6.35100857948859e+00 +1.24236145363939e+00 +7.61260651018449e+00

deorbit delta-v vector and magnitude
-----

x-component of delta-v      80.301516 meters/second
y-component of delta-v      117.688815 meters/second
z-component of delta-v      -21.001409 meters/second
total delta-v      144.014061 meters/second
```

Orbital Mechanics with MATLAB

deorbit delta-v pointing angles

pitch angle -2.256618 degrees

yaw angle -179.973071 degrees

time and conditions after deorbit maneuver

calendar date 18-Mar-2010

UTC time 12:30:45.875

sma (km)	eccentricity	inclination (deg)	argper (deg)
+6.62986196765162e+03	+3.74561317348360e-02	+2.84998225494152e+01	+1.08881122818562e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.20001021787282e+02	+1.81117979216534e+02	+2.89999102035096e+02	+8.95398701301721e+01
rx (km)	ry (km)	rz (km)	rmag (km)
-5.45318321844679e+03	+2.83906883966968e+03	-3.08403806222734e+03	+6.87814000000000e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-3.92881383778096e+00	-6.23331976439964e+00	+1.22136004444855e+00	+7.46870630131950e+00

time and conditions at entry interface

calendar date 18-Mar-2010

UTC time 13:00:49.638

sma (km)	eccentricity	inclination (deg)	argper (deg)
+6.63194303419306e+03	+3.87915541331080e-02	+2.85109989908368e+01	+1.07810489094078e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.19909866252810e+02	+2.99617139359573e+02	+4.74276284536510e+01	+8.95820323314157e+01
rx (km)	ry (km)	rz (km)	rmag (km)
-5.45318321844679e+03	+2.83906883966968e+03	-3.08403806222734e+03	+6.87814000000000e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
+7.50958358577267e+00	+3.73745768234497e-01	+2.46143688225353e+00	+7.91152343460458e+00

relative flight path coordinates at entry interface

east longitude 252.44489639 degrees

geocentric declination 20.58001385 degrees

flight path angle -2.00000000 degrees

relative azimuth 68.65207490 degrees

position magnitude 6497.40258326 kilometers

velocity magnitude 7.49698673 kilometers/second

geodetic coordinates at entry interface

geodetic latitude 20.70458617 degrees

geodetic altitude 121.92000000 kilometers

flight time from maneuver to EI 30.06271094 minutes

The following is a brief description of the information provided in the script output.

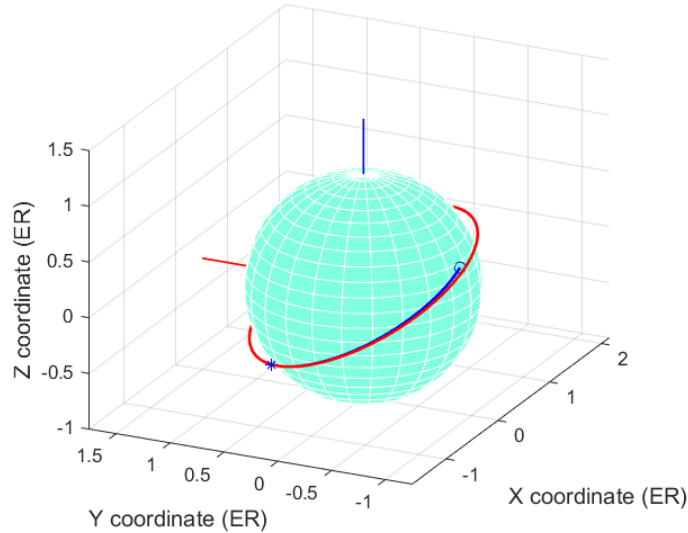
sma (km) = semimajor axis in kilometers
eccentricity = orbital eccentricity (non-dimensional)
inclination (deg) = orbital inclination in degrees
argper (deg) = argument of perigee in degrees
raan (deg) = right ascension of the ascending node in degrees
true anomaly (deg) = true anomaly in degrees
arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.
period (mins) = orbital period in minutes
rx (km) = x-component of the position vector in kilometers
ry (km) = y-component of the position vector in kilometers
rz (km) = z-component of the position vector in kilometers
rmag (km) = scalar magnitude of the position vector in kilometers
vx (kps) = x-component of the velocity vector in kilometers per second
vy (kps) = y-component of the velocity vector in kilometers per second
vz (ksp) = z-component of the velocity vector in kilometers per second
vmag (kps) = scalar magnitude of the velocity vector in kilometers per second

The components of the de-orbit delta-v vector are displayed in the ECI coordinate system. The relative flight path coordinates are with respect to a rotating Earth. The UTC time is given in hours, minutes and seconds.

The `deorbit_snopt` script will also create a three-dimensional graphics display of the initial orbit and re-entry trajectory. The following is the graphic image for this example. The initial orbit trace is red and the re-entry trajectory is blue. The dimensions are Earth radii (ER) and the plot is labeled with an Earth-centered-inertial (ECI) coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis. The impulse location is marked with a blue asterisk and entry interface is marked with a small blue circle.

Trajectory image files are saved to disk in both `tif` format and MATLAB `fig` format with a file name indicating the solution number. The disk file names are `deorbit_snopt.tif` and `deorbit_snopt.fig`. The interactive features of MATLAB graphics allow the user to manipulate the `fig` version of the trajectory display. These capabilities allow the user to interactively find the best viewpoint as well as verify basic three-dimensional geometry of the orbital maneuver and entry.

Optimal Single Impulse De-orbit



The following is the output created by this MATLAB script for the optimal de-orbit from a typical highly elliptical orbit (HEO).

```
*****
single impulse deorbit from Earth orbits
*****

time and conditions prior to deorbit maneuver
-----

calendar date      18-Mar-2010
UTC time           12:30:45.875

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+2.44140000000000e+04 +7.27044000000000e-01 +2.85000000000000e+01 +2.70000000000000e+02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+2.20000000000000e+02 +1.80010875055299e+02 +9.00108750552986e+01 +6.32729583646570e+02

      rx (km)      ry (km)      rz (km)      rmag (km)
+2.38242965164960e+04 -2.83802405542600e+04 +2.01189455552070e+04 +4.21640501929899e+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
+1.22991535564376e+00 +1.03330298046443e+00 -5.32994999257885e-04 +1.60636456496297e+00

deorbit delta-v vector and magnitude
-----

x-component of delta-v      -16.099990 meters/second
y-component of delta-v      -13.510509 meters/second
z-component of delta-v       0.004894 meters/second

total delta-v               21.017696 meters/second

deorbit delta-v pointing angles
-----
pitch angle                  -0.002649 degrees
yaw angle                    179.989285 degrees

time and conditions after deorbit maneuver
-----

calendar date      18-Mar-2010
```

Orbital Mechanics with MATLAB

```
UTC time          12:30:45.875

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+2.43140994191278e+04 +7.34139993195612e-01 +2.84999999733610e+01 +2.69999971672997e+02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+2.20000297715537e+02 +1.80010641744793e+02 +9.00106134177898e+01 +6.28849923664669e+02

      rx (km)      ry (km)      rz (km)      rmag (km)
+2.38242965164960e+04 -2.83802405542600e+04 +2.01189455552070e+04 +4.21640501929899e+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
+1.21381536549559e+00 +1.01979247133070e+00 -5.28100846447572e-04 +1.58534687213445e+00

time and conditions at entry interface
-----

calendar date      18-Mar-2010
UTC time          17:43:27.044

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+2.43457651117266e+04 +7.34618508714193e-01 +2.84881736895298e+01 +2.70116511603440e+02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+2.19931376435734e+02 +3.50932531329709e+02 +2.61049042933149e+02 +6.30078806344008e+02

      rx (km)      ry (km)      rz (km)      rmag (km)
+2.38242965164960e+04 -2.83802405542600e+04 +2.01189455552070e+04 +4.21640501929899e+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-8.39531350465642e+00 -5.97405295708906e+00 -4.38335404240751e-01 +1.03132310893426e+01

relative flight path coordinates at entry interface
-----

east longitude      37.72990551 degrees
geocentric declination -28.11018407 degrees
flight path angle    -4.00000004 degrees
relative azimuth     95.03036771 degrees
position magnitude    6495.29077143 kilometers
velocity magnitude     9.89797352 kilometers/second

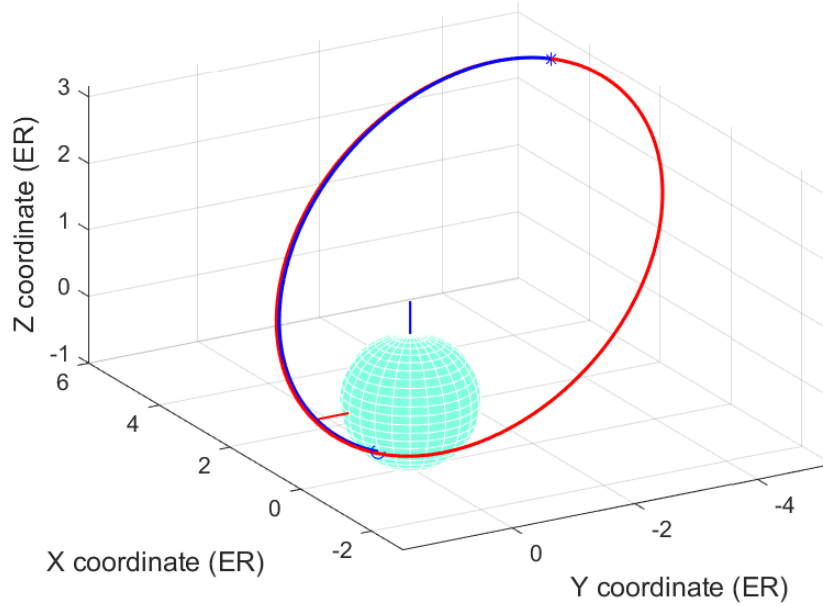
geodetic coordinates at entry interface
-----

geodetic latitude    -28.26740378 degrees
geodetic altitude     121.92000146 kilometers

flight time from maneuver to EI    312.68615417 minutes
```

Here's the trajectory graphics display for this example.

Optimal Single Impulse De-orbit



Technical discussion

This section is a brief explanation of the algorithms implemented in this MATLAB script.

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.

In this classic maneuver optimization problem, the maneuver true anomaly, the ECI components of the maneuver delta-v vector and the flight time from the maneuver to the entry interface are the *control variables*. The scalar magnitude of the de-orbit ΔV is the *objective function*, and the geodetic altitude and relative flight path angle at the entry interface are the *nonlinear equality constraints*.

Initial guess

An initial guess for the scalar magnitude of the de-orbit delta-v and time-of-flight from the maneuver location to the entry interface is determined using analytic solutions for these values relative to a non-rotating, spherical Earth model and two-body or Keplerian motion. The analytic solution for circular orbits is discussed in Appendix B and Appendix C contains the equations for elliptical orbits. Please note the elliptical orbit analytic solution assumes the de-orbit maneuver occurs at apogee (true anomaly $= 180^\circ$). This is the typical true anomaly initial guess for de-orbit from elliptical orbits.

The initial guess for the de-orbit delta-v vector is aligned opposite (retrograde) to the unit velocity vector on the initial Earth orbit at the maneuver location. This creates an impulsive velocity increment in the Earth-centered-inertial (ECI) Cartesian coordinate system.

Spacecraft equations of motion

During the solution process, the `deorbit_snopt` script numerically propagates the spacecraft trajectory from the maneuver time to the current estimate of the time at the entry interface. The system of six first-order differential equations subject to Earth gravity, defined in the ECI coordinate system (x, y, z) , is given by the following expressions

$$\dot{y}_1 = v_x = y_4 \quad \dot{y}_2 = v_y = y_5 \quad \dot{y}_3 = v_z = y_6$$

$$\dot{y}_4 = -\mu \frac{r_x}{r^3} \left\{ 1 + \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(1 - \frac{5r_z^2}{r^2} \right) \right\} \quad \dot{y}_5 = -\mu \frac{r_y}{r^3} \left\{ 1 + \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(1 - \frac{5r_z^2}{r^2} \right) \right\} \quad \dot{y}_6 = -\mu \frac{r_z}{r^3} \left\{ 1 + \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(3 - \frac{5r_z^2}{r^2} \right) \right\}$$

where $r = \sqrt{r_x^2 + r_y^2 + r_z^2} = \sqrt{y_1^2 + y_2^2 + y_3^2}$. In these equations μ and r_{eq} are the gravitational constant and equatorial radius of the Earth, and J_2 is the first order oblateness gravity coefficient.

At the entry interface, the algorithm computes the errors in the target constraints according to

$$\varepsilon_h = h_p - h_t$$

$$\varepsilon_\gamma = \gamma_p - \gamma_t$$

where h is the geodetic altitude and γ is the flight path angle relative to a rotating Earth. In these equations, the p subscript indicates the value *predicted* by the software, and the t subscript is the *target* value provided by the user. During the solution process, the SNOPT algorithm attempts to drive these two errors to zero.

The equations for calculating the relative flight path coordinates from an ECI position and velocity vectors is summarized in Appendix D. The algorithm used to calculate geodetic coordinates can be found in Appendix E and Appendix F discusses the coordinate system used to define the pitch and yaw orientation angles of the maneuver.

SNOPT algorithm implementation

This section provides details about the MATLAB source code that solve this nonlinear programming (NLP) problem using the SNOPT algorithm. MATLAB versions of SNOPT for several computer platforms can be found at Professor Philip Gill's University of California, San Diego web site which is located at <http://scicomp.ucsd.edu/~peg/>. Professor Gill's web site also includes a PDF version of the SNOPT software user's guide.

The SNOPT algorithm requires an initial guess for the control variables. For this problem they are given by

```
xg(1) = oevpo(6);
xg(2) = dvg(1);
xg(3) = dvg(2);
xg(4) = dvg(3);
xg(5) = dtof;
```

where $xg(1)$ is the user's initial guess for the true anomaly on the initial orbit at the time of the impulsive maneuver. $xg(2)$, $xg(2)$ and $xg(3)$ are the initial guesses for the ECI components of the de-orbit impulse, and $dtof$ is the initial guess for the transfer time from the de-orbit maneuver to the entry interface.

The algorithm also requires lower and upper bounds for the control variables. These are determined from the initial guesses and user-defined true anomaly boundaries as follows:

```
% lower and upper bounds for deorbit true anomaly (radians)
xlwr(1) = ta_lower;
xupr(1) = ta_upper;

% lower and upper bounds for components of
% deorbit delta-v vector (kilometers/second)
dvm = norm(dvg);
xlwr(2:4) = -(dvm + 0.1 * dvm);
xupr(2:4) = +(dvm + 0.1 * dvm);

% lower and upper bounds for flight time
% from maneuver to entry interface (seconds)
```

```
xlwr(5) = dtof - 30.0;  
xupr(5) = dtof + 30.0;
```

The algorithm also requires lower and upper bounds on the objective function. For this problem these bounds are given by

```
% bounds on objective function  
flow(1) = 0.0;  
fupp(1) = +Inf;
```

The following MATLAB code sets the lower and upper bounds for the two equality constraints (geodetic altitude and relative flight path angle) at the entry interface.

```
% geodetic altitude at entry interface equality constraint  
flow(2) = 0.0;  
fupp(2) = 0.0;  
  
% relative flight path angle at entry interface equality constraint  
flow(3) = 0.0;  
fupp(3) = 0.0;
```

The actual call to the SNOPT MATLAB interface function is as follows

```
[x, f, inform, xmul, fmul] = snopt(xg, xlwr, xupr, xmul, xstate, ...  
    flow, fupp, fmul, fstate, 'deorbit_shoot');
```

where `deorbit_shoot` is the name of the MATLAB function that integrates the spacecraft equations of motion and computes the current value of the objective function and the equality constraints at the entry interface. The solution for the control variables is returned in the `x` vector and `f` is the converged value of the objective function. Please consult the SNOPT documentation for additional information about the syntax of this function.

Algorithm Resources

“User’s Guide for SNOPT Version 7, A Fortran Package for Large-Scale Nonlinear Programming”, Philip E. Gill, Walter Murray and Michael A. Saunders, March 20, 2006.

“Optimum Deboost Altitude for Specified Atmospheric Entry Angle”, Jerome M. Baker, Bruce E. Baxter, and Paul D. Arthur, *AIAA Journal*, Vol. 1, No. 7, July 1963.

“Deboost from Circular Orbits”, A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, Jul-Aug., 1966.

Hypersonic and Planetary Entry Flight Mechanics, Vinh, Busemann and Culp, The University of Michigan Press, 1980.

“On Autonomous Optimal Deorbit Guidance”, Morgan C. Baldwin, Binfeng Pan and Ping Lu, AIAA 2009-5667, AIAA Guidance, Navigation, and Control Conference, August 10-13, 2009.

“Autonomous Optimal Deorbit Targeting”, Donald J. Jezewski, AAS 91-136, AAS/AIAA Spaceflight Mechanics Meeting, February 11-13, 1991.

“Analysis of the Accuracy of Ballistic Descent from a Circular Circumterrestrial Orbit”, Yu. G. Sikharulidze and A. N. Korchagin, *Cosmic Research*, Vol. 40, No. 1, 2002, pp.75-87.

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

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

APPENDIX A

Optimization Toolbox Implementation

There is a version of this MATLAB script named `deorbit_otb` that uses the Mathworks Optimization Toolbox to solve this orbital mechanics problem. This appendix describes the source code implementation using the `fmincon`/interior-point algorithm. Unlike SNOPT, this version requires the mission constraints and objective algorithms reside in two different MATLAB functions.

The following MATLAB source code solves the deorbit trajectory optimization problem.

```
% load initial guesses for control variables

xg(1) = oevpo(6);

xg(2) = dvg(1);

xg(3) = dvg(2);

xg(4) = dvg(3);

xg(5) = dtof;

% lower and upper bounds for deorbit true anomaly (radians)

xlwr(1) = ta_lower;

xupr(1) = ta_upper;

% lower and upper bounds for components of
% deorbit delta-v vector (kilometers/second)

dvm = norm(dvg);

xlwr(2:4) = -(dvm + 0.1 * dvm);

xupr(2:4) = +(dvm + 0.1 * dvm);

% lower and upper bounds for flight time
% from maneuver to entry interface (seconds)

xlwr(5) = dtof - 30.0;

xupr(5) = dtof + 30.0;

% solve trajectory optimization problem

options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', 'interior-point', ...
    'MaxFunctionEvaluations', 5000, 'FiniteDifferenceType', 'forward');

% optimize with user-defined mission constraints

[x, fval] = fmincon('deorbit_objective', xg, [], [], [], [], xlwr, xupr,
    'deorbit_constraints', options);
```

The MATLAB function that evaluates the objective function is named `deorbit_objective` and `deorbit_constraints` calculates the current mission constraints.

Feel free to experiment with other `fmincon` non-linear programming algorithms such as *sqp*, etc.

APPENDIX B

De-orbit from a Circular Earth Orbit

The scalar magnitude of the single impulsive maneuver required to de-orbit a spacecraft from an initial circular orbit can be determined from the following expression

$$\Delta V = V_{c_e} \sqrt{\frac{1}{\tilde{r}}} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\} = V_{c_i} \left\{ 1 - \frac{\sqrt{\frac{2(\tilde{r}-1)}{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}}}{\sqrt{\left(\frac{\tilde{r}}{\cos \gamma_e}\right)^2 - 1}} \right\}$$

where

$$\tilde{r} = \frac{h_i + r_{eq}}{h_e + r_{eq}} = \frac{r_i}{r_e} = \text{radius ratio}$$

$$V_{c_e} = \sqrt{\frac{\mu}{(h_e + r_{eq})}} = \sqrt{\frac{\mu}{r_e}} = \text{local circular velocity at entry interface}$$

$$V_{c_i} = \sqrt{\frac{\mu}{(h_i + r_{eq})}} = \sqrt{\frac{\mu}{r_i}} = \text{local circular velocity of initial circular orbit}$$

γ_e = flight path angle at entry interface

h_i = altitude of initial circular orbit

h_e = altitude at entry interface

r_i = radius of initial circular orbit

r_e = radius at entry interface

r_{eq} = Earth equatorial radius

μ = Earth gravitational constant

This algorithm is described in the technical article, “Deboost from Circular Orbits”, A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, Jul-Aug., 1966. Additional information can be found in Chapter 5 of *Hypersonic and Planetary Entry Flight Mechanics* by Vinh, Busemann and Culp, The University of Michigan Press.

The true anomaly on the de-orbit trajectory at the entry interface θ_e can be determined from the following two equations

$$\sin \theta_e = \frac{\dot{r}}{e_d} \sqrt{\frac{a_d(1-e_d^2)}{\mu}} \quad \cos \theta_e = \frac{a_d(1-e_d^2)}{e_d r_e} - \frac{1}{e_d}$$

and the following four quadrant inverse tangent operation

$$\theta_e = \tan^{-1}(\sin \theta_e, \cos \theta_e)$$

where

e_d = eccentricity of the de-orbit trajectory

a_d = semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu[2a_d r_e - r_e^2 - a_d^2(1 - e_d^2)]}{a_d r_e^2}}$$

The elapsed time-of-flight between perigee of the de-orbit trajectory and the entry true anomaly θ_e is given by

$$t(\theta_e) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1-e_d}{1+e_d}} \tan \frac{\theta_e}{2} \right\} - \frac{e_d \sqrt{1-e_d^2} \sin \theta_e}{1+e_d \cos \theta_e} \right]$$

In this equation τ is the Keplerian orbital period of the de-orbit trajectory and is equal to $2\pi\sqrt{a_d^3/\mu}$.

Therefore, the flight time between the de-orbit impulse and entry interface is given by

$$\Delta t = t(\theta_e) - t(180^\circ) = t(\theta_e) - \frac{\tau}{2}$$

Finally, the orbital speed at the entry interface V_e can be determined from

$$V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$$

APPENDIX C

De-orbit from an Elliptical Earth Orbit

The scalar magnitude of the impulsive delta-v for de-orbit from an initial elliptical orbit is given by

$$\Delta V = \sqrt{\frac{\mu}{r_e}} \left(\sqrt{\frac{2\tilde{r}_p}{\tilde{r}_a(\tilde{r}_a + \tilde{r}_p)}} - \sqrt{\frac{2(\tilde{r}_a - 1)}{\tilde{r}_a(\tilde{r}_a^2 - \cos^2 \gamma_e)}} \cos \gamma_e \right)$$

where

r_e = geocentric radius at the entry altitude

$\tilde{r}_a = r_a / r_e$

$\tilde{r}_p = r_p / r_e$

γ_e = flight path angle at entry

r_a = apogee radius of the initial elliptical orbit

r_p = perigee radius of the initial elliptical orbit

μ = gravitational constant of the Earth

The true anomaly at entry can be determined from the following series of equations.

$$\sin \theta_e = \frac{\dot{r}}{e_d} \sqrt{\frac{a_d(1-e_d^2)}{\mu}} \quad \cos \theta_e = \frac{a_d(1-e_d^2)}{e_d r_e} - \frac{1}{e_d} \quad \theta_e = \tan^{-1}(\sin \theta_e, \cos \theta_e)$$

where

e_d = eccentricity of the de-orbit trajectory

a_d = semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu[2a_d r_e - r_e^2 - a_d^2(1-e_d^2)]}{a_d r_e^2}}$$

The time-of-flight between perigee and the entry true anomaly θ_e is given by

$$\text{tof}(\theta_e) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1-e_d}{1+e_d}} \tan \frac{\theta_e}{2} \right\} - \frac{e_d \sqrt{1-e_d^2} \sin \theta_e}{1+e_d \cos \theta_e} \right]$$

In this equation, τ is the orbital period of the de-orbit trajectory.

Therefore, the flight time between the de-orbit impulse time and entry is given by

$$\Delta t = \text{tof}(\theta_e) - \text{tof}(180^\circ) = \text{tof}(\theta_e) - \frac{\tau}{2}$$

Finally, the speed at reentry V_e can be determined from

$$V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$$

APPENDIX D

Flight Path Coordinates

Relative flight path coordinates are defined with respect to a rotating Earth. This set of coordinates consists of the following trajectory elements

r = geocentric radius

V = speed

γ = flight path angle

δ = geocentric declination

λ = geographic longitude (+ east)

ψ = flight azimuth (+ clockwise from north)

Please note the sign and direction convention.

The following are several useful equations that summarize the relationships between inertial and relative flight path coordinates.

$$v_r \sin \gamma_r = v_i \sin \gamma_i$$

$$v_r \cos \gamma_r \cos \psi_r = v_i \cos \gamma_i \cos \psi_i$$

$$v_r \cos \gamma_r \sin \psi_r + \omega_e r \cos \delta = v_i \cos \gamma_i \sin \psi_i$$

where the r subscript denotes relative coordinates and the i subscript inertial coordinates.

The inertial speed can also be computed from the following expression

$$v_i = \sqrt{v^2 + 2vr\omega \cos \gamma \sin \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}$$

The inertial flight path angle can be computed from

$$\cos \gamma_i = \sqrt{\frac{v^2 \cos^2 \gamma + 2vr\omega \cos \gamma \cos \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}{v^2 + 2vr\omega \cos \gamma \cos \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}}$$

The inertial azimuth can be computed from

$$\cos \psi_i = \frac{v \cos \gamma \cos \psi + r\omega \cos \delta}{\sqrt{v^2 \cos^2 \gamma + 2vr\omega \cos \gamma \cos \psi \cos \delta + r^2 \omega^2 \cos^2 \delta}}$$

where all coordinates on the right-hand-side of these equations are relative to a rotating Earth.

The transformation of an Earth-centered inertial (ECI) position vector \mathbf{r}_{ECI} to an Earth-centered fixed (ECF) position vector \mathbf{r}_{ECF} is given by the following vector-matrix operation

$$\mathbf{r}_{ECF} = [\mathbf{T}] \mathbf{r}_{ECI}$$

where the elements of the transformation matrix $[\mathbf{T}]$ are given by

$$[\mathbf{T}] = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and θ is the Greenwich apparent sidereal time at the moment of interest. Greenwich sidereal time is given by the following expression:

$$\theta = \theta_{g0} + \omega_e t$$

where θ_{g0} is the Greenwich sidereal time at 0 hours UTC, ω_e is the inertial rotation rate of the Earth, and t is the elapsed time since 0 hours UTC.

Finally, the flight path coordinates are determined from the following set of equations

$$\begin{aligned} r &= \sqrt{r_{ECF}^2 + r_{ECF_y}^2 + r_{ECF_z}^2} & v &= \sqrt{v_{ECF}^2 + v_{ECF_y}^2 + v_{ECF_z}^2} \\ \lambda &= \tan^{-1}(r_{ECF_y}, r_{ECF_x}) & \delta &= \sin^{-1}\left(\frac{r_{ECF_z}}{|\mathbf{r}_{ECF}|}\right) \\ \gamma &= \sin^{-1}\left(-\frac{v_{R_z}}{|\mathbf{v}_R|}\right) & \psi &= \tan^{-1}[v_{R_y}, v_{R_x}] \end{aligned}$$

where

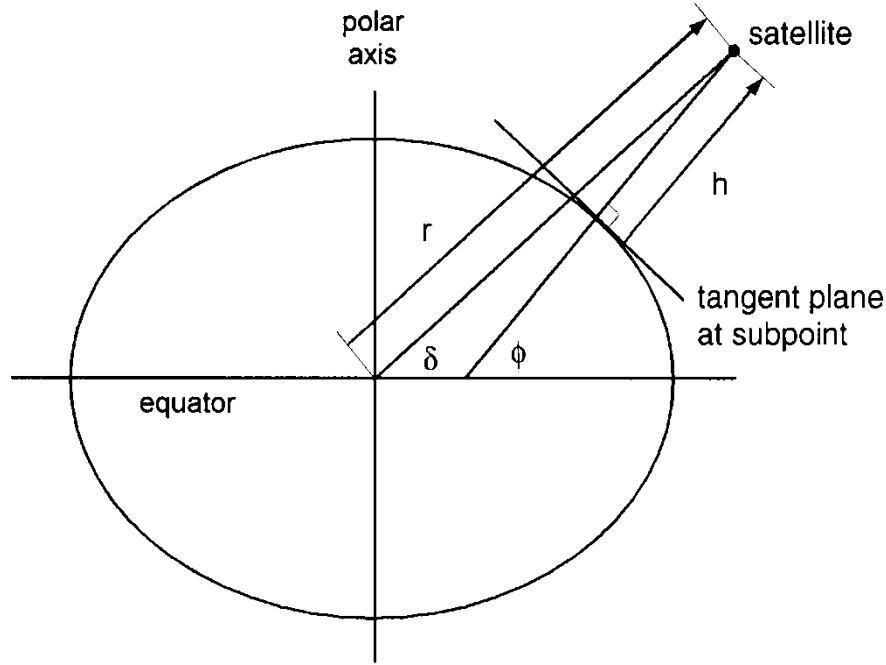
$$\mathbf{v}_R = \begin{bmatrix} -\sin \delta \cos \lambda & -\sin \delta \sin \lambda & \cos \delta \\ -\sin \lambda & \cos \lambda & 0 \\ -\cos \delta \cos \lambda & -\cos \delta \sin \lambda & -\sin \delta \end{bmatrix} \mathbf{v}_{ECF}$$

Please note the two-argument inverse tangent calculation for λ and ψ is a four-quadrant operation.

APPENDIX E

Geodetic Coordinates

The following diagram illustrates the geometric relationship between geocentric and geodetic coordinates.



In this diagram, δ is the geocentric declination, ϕ is the geodetic latitude, r is the geocentric distance, and h is the geodetic altitude. The exact mathematical relationship between geocentric and geodetic coordinates is given by the following system of two nonlinear equations

$$\begin{aligned}(c + h)\cos\phi - r\cos\delta &= 0 \\ (s + h)\sin\phi - r\sin\delta &= 0\end{aligned}$$

where the geodetic constants c and s are given by

$$c = \frac{r_{eq}}{\sqrt{1 - (2f - f^2)\sin^2\phi}} \quad s = c(1 - f)^2$$

and r_{eq} is the Earth equatorial radius (6378.14 kilometers) and f is the flattening factor for the Earth (1/298.257).

In this MATLAB script, the geodetic latitude is determined using the following expression

$$\phi = \delta + \left(\frac{\sin 2\delta}{\rho} \right) f + \left[\left(\frac{1}{\rho^2} - \frac{1}{4\rho} \right) \sin 4\delta \right] f^2$$

which is a series expansion in flattening factor (NASA TN D-7522).

The geodetic altitude is calculated from

$$\hat{h} = (\hat{r} - 1) + \left\{ \left(\frac{1 - \cos 2\delta}{2} \right) f + \left[\left(\frac{1}{4\rho} - \frac{1}{16} \right) (1 - \cos 4\delta) \right] f^2 \right\}$$

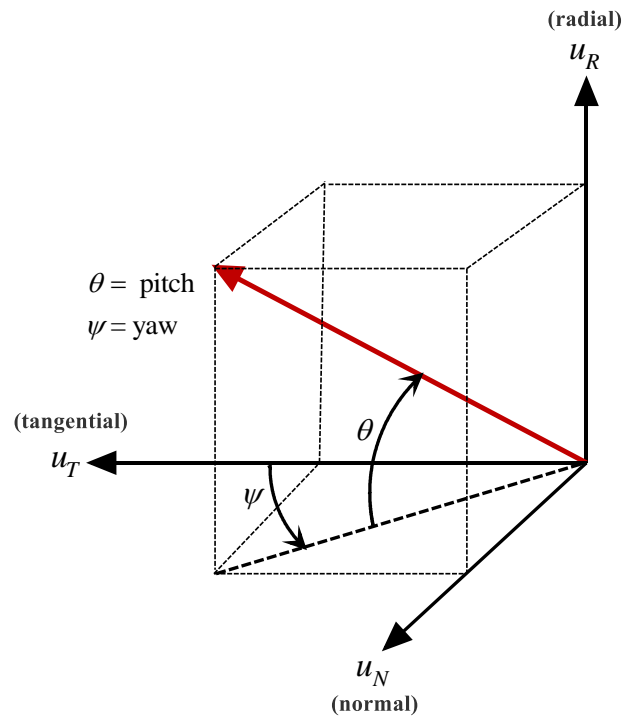
In these equations, ρ is the geocentric distance of the satellite, $\hat{h} = h / r_{eq}$ and $\hat{r} = \rho / r_{eq}$.

APPENDIX F

Pitch and Yaw Angles

The pitch and yaw angles for the de-orbit impulsive maneuver are computed and displayed in the local-vertical-local horizontal (LVLH; also called the radial-tangential-normal RTN) coordinate system. The following diagram illustrates the geometry of the pitch and yaw angles in this system. In this figure, the radial direction is along the geocentric radius vector directed away from the Earth, the tangential direction is tangent to the orbit in the direction of the orbital motion, and the normal direction is along the angular momentum vector of the orbit.

The pitch angle is positive above the local horizontal plane formed by the tangential and normal directions, and the yaw angle is positive in the direction of the angular momentum vector which is perpendicular to the orbit plane. The pitch angle varies between ± 90 degrees and the yaw angle can have a value between ± 180 degrees.



The following is the MATLAB source code for the function that computes the orientation angles using the Earth-centered-inertial (ECI) position and velocity vectors (*reci*, *veci*) at the impulse location and the unit pointing vector of the impulsive delta-v (*ueci*).

```
function [pitch, yaw] = ueci2angles(reci, veci, ueci)

% convert eci unit vector to rtn angles

% input

% reci = eci position vector (kilometers)
% veci = eci velocity vector (kilometers/second)
% ueci = eci unit vector

% output
```

Orbital Mechanics with MATLAB

```
% pitch = pitch angle (radians)
%         positive above the local horizon
% yaw    = yaw angle (radians)
%         positive in the direction of the angular momentum vector

% Orbital Mechanics with MATLAB

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

% compute radial frame unit vectors

rmag = norm(peci);

xrdl = peci / rmag;

zrdl = cross(peci, veci);

hmag = norm(zrdl);

zrdl = zrdl / hmag;

yrdl = cross(zrdl, xrdl);

% unit vector in radial-tangential-normal frame

umee(1) = dot(ueci, xrdl);

umee(2) = dot(ueci, yrdl);

umee(3) = dot(ueci, zrdl);

% pitch angle (radians)

pitch = asin(umee(1));

% yaw angle (radians)

yaw = atan2(umee(3), umee(2));
```

The Hohmann Orbit Transfer

The coplanar circular orbit-to-circular orbit transfer was discovered by the German engineer Walter Hohmann in 1925 and described in his classic report, *The Attainability of Celestial Bodies*. The transfer consists of a velocity impulse on an initial circular orbit, in the direction of motion and collinear with the velocity vector, which propels the space vehicle into an elliptical transfer orbit. At a transfer angle of 180 degrees from the first impulse, a second velocity impulse or ΔV , also collinear and in the direction of motion, places the vehicle into a final circular orbit at the desired final altitude. The impulsive ΔV assumption means that the velocity, but not the position, of the vehicle is changed instantaneously. This is equivalent to a rocket engine with infinite thrust magnitude. The Hohmann formulation is the ideal and minimum energy solution to this type of time-free orbit transfer problem.

Coplanar Equations

For the coplanar Hohmann transfer both velocity impulses are confined to the orbital planes of the initial and final orbits. For a Hohmann transfer from a lower altitude orbit to a higher altitude circular orbit, the first impulse creates an elliptical transfer orbit with a perigee altitude equal to the altitude of the initial circular orbit and an apogee altitude equal to the altitude of the final orbit. The second impulse circularizes the transfer orbit at apogee. Both impulses are *prograde* which means that they are in the direction of orbital motion.

We begin by defining three *normalized* radii as follows:

$$R_1 = \sqrt{2 \frac{r_f}{r_i + r_f}} \quad R_2 = \sqrt{\frac{r_i}{r_f}} \quad R_3 = \sqrt{2 \frac{r_i}{r_i + r_f}}$$

where r_i is the geocentric radius of the initial circular orbit and r_f is the radius of the final circular mission orbit. The relationship between radius and initial orbit altitude h_i and the final orbit altitude h_f is as follows:

$$r_i = r_e + h_i$$

$$r_f = r_e + h_f$$

where r_e is the radius of the Earth.

The magnitude of the first impulse is

$$\Delta V_1 = V_{tc} \sqrt{1 + R_1^2 - 2R_1}$$

and is simply the difference between the speed on the initial orbit and the perigee speed of the transfer orbit. The scalar magnitude of the second impulse is

$$\Delta V_2 = V_{tc} \sqrt{R_2^2 + R_2^2 R_3^2 - 2R_2^2 R_3}$$

which is the difference between the speed on the final orbit and the apogee speed of the transfer ellipse.

In each of these ΔV equations V_{lc} is called the *local circular velocity*. It can be determined from

$$V_{lc} = \sqrt{\frac{\mu}{r_i}}$$

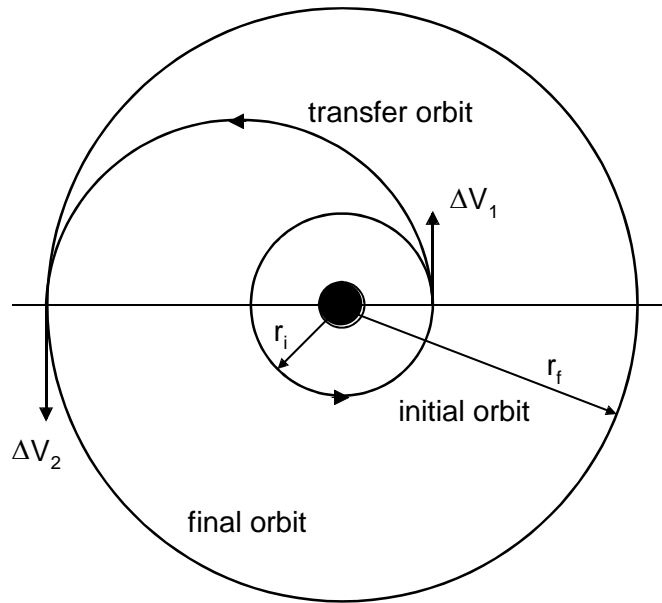
and represents the scalar speed in the initial orbit. In these equations μ is the gravitational constant of the central body. The transfer time from the first impulse to the second is equal to one half the orbital period of the transfer ellipse

$$\tau = \pi \sqrt{\frac{a^3}{\mu}}$$

where a is the semimajor axis of the transfer orbit and is equal to $(r_i + r_f)/2$. The orbital eccentricity of the transfer ellipse is

$$e = \frac{\max(r_i, r_f) - \min(r_i, r_f)}{r_f - r_i}$$

The following diagram illustrates the geometry of the coplanar Hohmann transfer.



Non-coplanar Equations

The non-coplanar Hohmann transfer involves orbital transfer between two circular orbits which have different orbital inclinations. For this problem the propulsive energy is minimized if we optimally partition the total orbital inclination change between the first and second impulses.

The scalar magnitude of the first impulse is

$$\Delta V_1 = V_{lc} \sqrt{1 + R_1^2 - 2R_1 \cos \theta_1}$$

where θ_1 is the plane change associated with the first impulse. The magnitude of the second impulse is

$$\Delta V_2 = V_{lc} \sqrt{R_2^2 + R_3^2 - 2R_2 R_3 \cos \theta_2}$$

where θ_2 is the plane change associated with the second impulse. These two equations are different forms of the law of cosines.

The total ΔV required for the maneuver is the sum of the two impulses as follows

$$\Delta V = \Delta V_1 + \Delta V_2$$

The relationship between the plane change angles is

$$\theta_t = \theta_1 + \theta_2$$

where θ_t is the total plane change angle between the initial and final orbits.

Optimizing the non-coplanar Hohmann transfer involves allocating the total plane change angle between the two maneuvers such that the total ΔV required for the mission is minimized. We can determine this answer by solving for the root of a derivative.

The partial derivative of the total ΔV with respect to the first plane change angle is given by:

$$\frac{\partial \Delta V}{\partial \theta_1} = \frac{R_1 \sin \theta_1}{\sqrt{1 + R_1^2 - 2R_1 \cos \theta_1}} - \frac{R_2^2 R_3 (\sin \theta_t \cos \theta_1 - \cos \theta_t \sin \theta_1)}{\sqrt{R_2^2 + R_3^2 - 2R_2 R_3 \cos(\theta_t - \theta_1)}}$$

If we determine the value of θ_1 which makes this derivative zero, we have found the optimum plane change required at the first impulse. Once θ_1 is calculated we can determine θ_2 from the total plane change angle relationship and the velocity impulses from the previous equations.

Numerical Solution

This numerical algorithm has been implemented in an interactive MATLAB script called `hohmann.m`. This script prompts the user for the initial and final altitudes in kilometers and the initial and final orbital inclinations in degrees. The software then calls the Brent root-finding algorithm to solve the partial derivative equation described above.

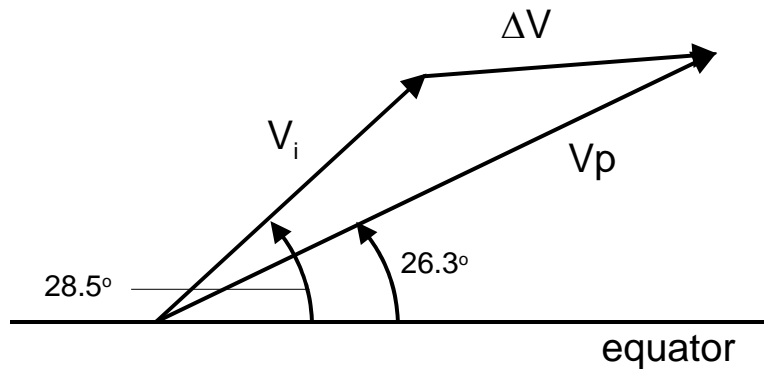
The call to the Brent root-finding algorithm is as follows:

```
[xroot, froot] = brent('hohmfunc', 0, dinc, rtol);
```

where `hohmfunc` is the objective function for this problem. Since we know that the optimum first plane change angle is somewhere between 0 and the total plane change angle `dinc`, we pass these as the bounds of the root. In the parameter list `rtol` is the user-defined root-finding convergence tolerance.

The following is a typical orbit transfer from a low altitude Earth orbit (LEO) at an altitude of 185.2 kilometers and an orbital inclination of 28.5 degrees to a geosynchronous Earth orbit (GSO) at an altitude of 35786.36 kilometers and 0 degrees inclination.

The following is a ΔV diagram for the first maneuver of this orbit transfer example. In this view we are looking along the line of nodes which is the mutual intersection of the park and transfer orbit planes with the equatorial plane.



In this diagram V_i is the speed on the initial park orbit, V_p is the perigee speed of the elliptic transfer orbit, and ΔV is the impulse required for the first maneuver. The inclinations of the park and transfer orbit are also labeled. From this geometry and the law of cosines, the required ΔV is given by

$$\Delta V = \sqrt{V_i^2 + V_p^2 - 2V_i V_p \cos \Delta i}$$

where Δi is the inclination difference or plane change between the park and transfer orbits.

User interaction with the script

The following is a typical user interaction with this MATLAB script. User inputs are in bold font.

```
Hohmann Orbit Transfer Analysis

please input the initial altitude kilometers
? 300

please input the final altitude kilometers
? 35786.2

please input the initial orbital inclination degrees
(0 <= inclination <= 180)
? 28.5

please input the final orbital inclination degrees
(0 <= inclination <= 180)
? 0
```

Orbital Mechanics with MATLAB

The following is the script solution for this example.

```
Hohmann Orbit Transfer Analysis
-----

initial orbit altitude          300.0000 kilometers
initial orbit radius            6678.1363 kilometers
initial orbit inclination       28.5000 degrees
initial orbit velocity          7725.7606 meters/second

final orbit altitude            35786.2000 kilometers
final orbit radius              42164.3363 kilometers
final orbit inclination         0.0000 degrees
final orbit velocity            3074.6540 meters/second

first inclination change        2.2002 degrees
second inclination change       26.2998 degrees
total inclination change        28.5000 degrees

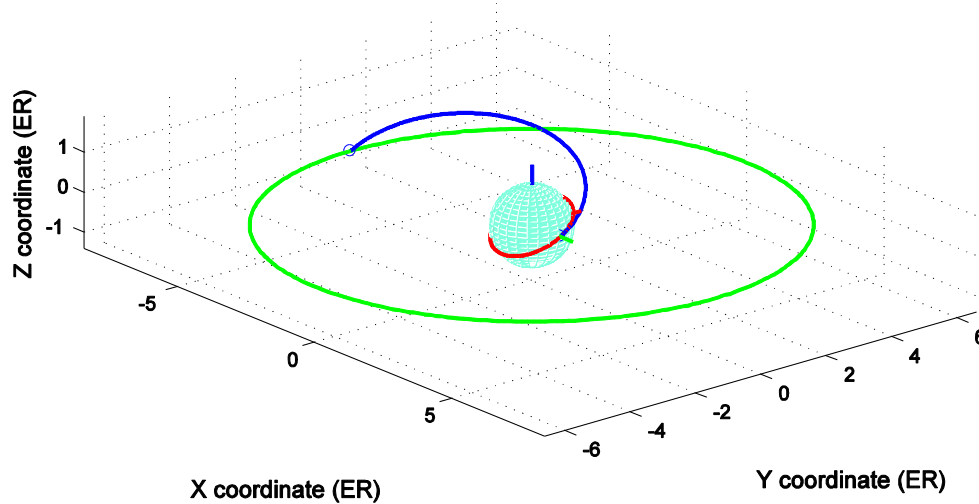
first delta-v                  2449.4551 meters/second
second delta-v                  1781.8532 meters/second
total delta-v                   4231.3083 meters/second

transfer orbit semimajor axis   24421.2363 kilometers
transfer orbit eccentricity     0.72654389
transfer orbit inclination      26.2998 degrees
transfer orbit perigee velocity 10151.4962 meters/second
transfer orbit apogee velocity  1607.8298 meters/second
transfer orbit coast time       18990.3276 seconds
                                316.5055 minutes
                                5.2751 hours
```

This MATLAB script is valid for Hohmann transfers from a high initial circular orbit to a lower final orbit. It also handles the case of transfer to a mission orbit with higher orbital inclination.

The `hohmann` script will also create a graphics display of the initial, transfer and final orbits. The following is the graphics display for this example. The initial orbit trace is red, the transfer orbit is blue and the final mission orbit is green. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis. The location of each impulse is marked with a small blue circle.

Hohmann Transfer: Initial, Transfer and Final Orbits



The interactive graphic features of MATLAB allow the user to rotate and zoom the display. These capabilities allow the user to interactively find the best viewpoint as well as verify basic three-dimensional geometry of the orbital transfer.

The `hohmann` MATLAB script will also create color a Postscript disk file of this graphic image. This image includes a TIFF preview and is created with MATLAB code similar to

```
print -depsc -tiff -r300 hohmann1.eps
```

Primer Vector Analysis

This section summarizes the primer vector analysis included with this MATLAB script. The term primer vector was invented by Derek F. Lawden and represents the adjoint vector for velocity. A technical discussion about primer theory can be found in Lawden's classic text, *Optimal Trajectories for Space Navigation*, Butterworths, London, 1963. Another excellent resource is "Primer Vector Theory and Applications", Donald J. Jezewski, NASA TR R-454, November 1975, along with "Optimal, Multi-burn, Space Trajectories", also by Jezewski.

As shown by Lawden, the following four necessary conditions must be satisfied in order for an impulsive orbital transfer to be *locally optimal*:

- (1) the primer vector and its first derivative are everywhere continuous
- (2) whenever a velocity impulse occurs, the primer is a unit vector aligned with the impulse and has unit magnitude ($\mathbf{p} = \hat{\mathbf{p}} = \hat{\mathbf{u}}_T$ and $\|\mathbf{p}\| = 1$)

(3) the magnitude of the primer vector may not exceed unity on a coasting arc ($\|\mathbf{p}\| = p \leq 1$)

(4) at all interior impulses (not at the initial or final times) $\mathbf{p} \cdot \dot{\mathbf{p}} = 0$; therefore, $d\|\mathbf{p}\|/dt = 0$ at the intermediate impulses

Furthermore, the scalar magnitudes of the primer vector derivative at the initial and final impulses provide information about how to improve the nominal transfer trajectory by changing the endpoint times and/or moving the impulse times. These four cases for non-zero slopes are summarized as follows;

- If $\dot{p}_0 > 0$ and $\dot{p}_f < 0 \rightarrow$ perform an initial coast before the first impulse and add a final coast after the second impulse
- If $\dot{p}_0 > 0$ and $\dot{p}_f > 0 \rightarrow$ perform an initial coast before the first impulse and move the second impulse to a later time
- If $\dot{p}_0 < 0$ and $\dot{p}_f < 0 \rightarrow$ perform the first impulse at an earlier time and add a final coast after the second impulse
- If $\dot{p}_0 < 0$ and $\dot{p}_f > 0 \rightarrow$ perform the first impulse at an earlier time and move the second impulse to a later time

The primer vector analysis of a two impulse orbital transfer involves the following steps.

First partition the two-body state transition matrix as follows:

$$\Phi(t, t_0) = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{r}}{\partial \mathbf{v}_0} \\ \frac{\partial \mathbf{v}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{v}}{\partial \mathbf{v}_0} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix}$$

where

$$\Phi_{11} = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} \end{bmatrix} = \begin{bmatrix} \partial x / \partial x_0 & \partial x / \partial y_0 & \partial x / \partial z_0 \\ \partial y / \partial x_0 & \partial y / \partial y_0 & \partial y / \partial z_0 \\ \partial z / \partial x_0 & \partial z / \partial y_0 & \partial z / \partial z_0 \end{bmatrix}$$

and so forth.

The value of the primer vector at any time t along a two body trajectory is given by

$$\mathbf{p}(t) = \Phi_{11}(t, t_0)\mathbf{p}_0 + \Phi_{12}(t, t_0)\dot{\mathbf{p}}_0$$

and the value of the primer vector derivative is

$$\dot{\mathbf{p}}(t) = \Phi_{21}(t, t_0)\mathbf{p}_0 + \Phi_{22}(t, t_0)\dot{\mathbf{p}}_0$$

which can also be expressed as

$$\begin{Bmatrix} \mathbf{p} \\ \dot{\mathbf{p}} \end{Bmatrix} = \Phi(t, t_0) \begin{Bmatrix} \mathbf{p}_0 \\ \dot{\mathbf{p}}_0 \end{Bmatrix}$$

The primer vector boundary conditions at the initial and final impulses are as follows:

$$\mathbf{p}(t_0) = \mathbf{p}_0 = \frac{\Delta \mathbf{V}_0}{|\Delta \mathbf{V}_0|} \quad \mathbf{p}(t_f) = \mathbf{p}_f = \frac{\Delta \mathbf{V}_f}{|\Delta \mathbf{V}_f|}$$

These two conditions illustrate that at the locations of velocity impulses, the primer vector is a unit vector in the direction of the corresponding impulse.

The value of the primer vector derivative at the initial time is

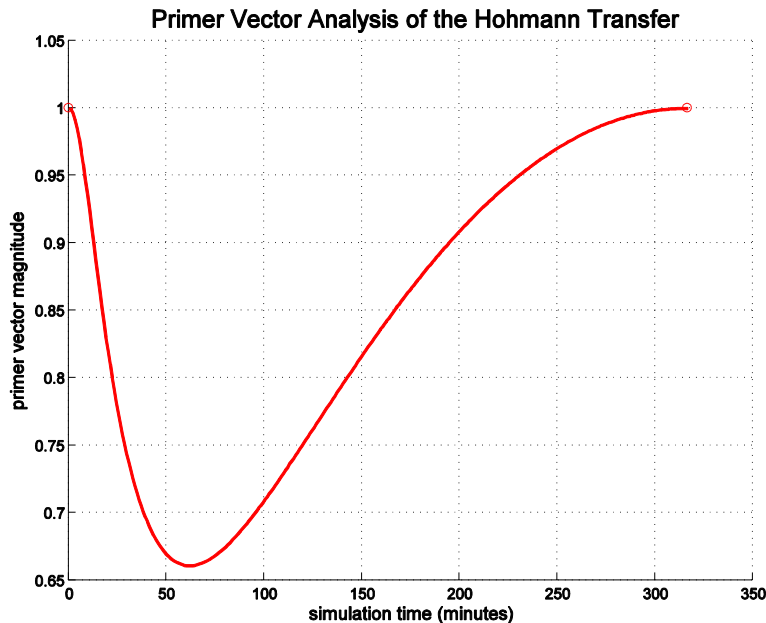
$$\dot{\mathbf{p}}(t_0) = \dot{\mathbf{p}}_0 = \Phi_{12}^{-1}(t_f, t_0) \{ \mathbf{p}_f - \Phi_{11}(t_f, t_0) \mathbf{p}_0 \}$$

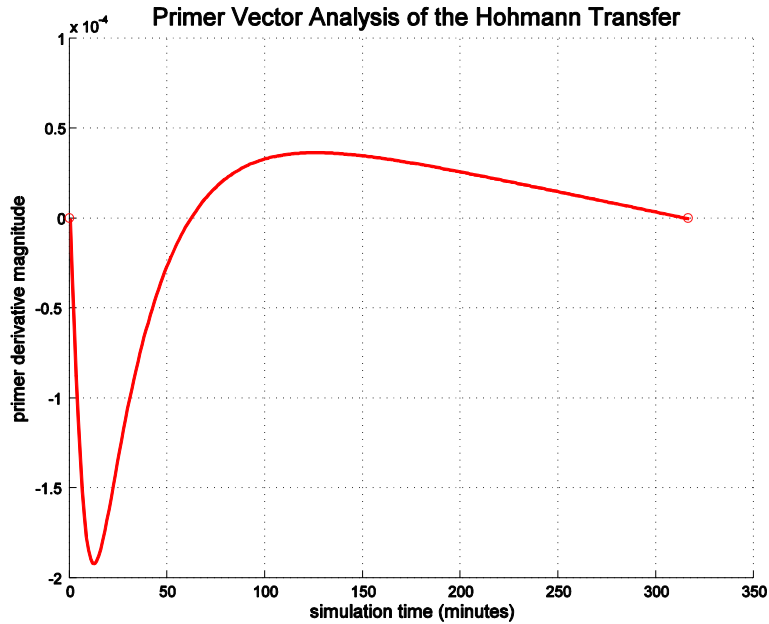
provided the Φ_{12} sub-matrix is non-singular.

The scalar magnitude of the derivative of the primer vector can be determined from

$$\frac{d\|\mathbf{p}\|}{dt} = \frac{d}{dt}(\mathbf{p} \cdot \mathbf{p})^2 = \frac{\dot{\mathbf{p}} \cdot \mathbf{p}}{\|\mathbf{p}\|}$$

The following two graphic images illustrate the behavior of the magnitudes of the primer vector and its derivative for the example given earlier. The location of each impulse is marked with a small red circle.





From the properties of the primer vector and its derivative, we can see that this orbit transfer is optimal.

The `hohmann` MATLAB script will also create color a Postscript disk file of these graphic images. This image includes a TIFF preview and is created with MATLAB source code similar to

```
print -depsc -tiff -r300 primer.eps
```

Algorithm resources

- (1) Walter Hohmann, *Die Erreichbarkeit der Himmelskorper*, Oldenbourg, Munich, 1925. Also, *The Attainability of Heavenly Bodies*, NASA Technical Translation F-44, 1960.
- (2) Jean-Pierre Marec, *Optimal Space Trajectories*, Elsevier, 1979.
- (3) R. P. Brent, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, 1972.
- (4) R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA, 1987.
- (5) D. F. Lawden, *Optimal Trajectories for Space Navigation*, Butterworths, London, 1963.
- (6) John E. Prussing, "Simple Proof of the Global Optimality of the Hohmann Transfer", *AIAA Journal of Guidance, Control and Dynamics*, Vol. 15, No. 4.
- (7) A. Miele, M. Ciarcia, and J. Mathwig, "Reflections on the Hohmann Transfer", *Journal of Optimization Theory and Applications*, Vol. 123, No. 2, pp. 233-253, November 2004.

The Gravity Perturbed Hohmann Transfer

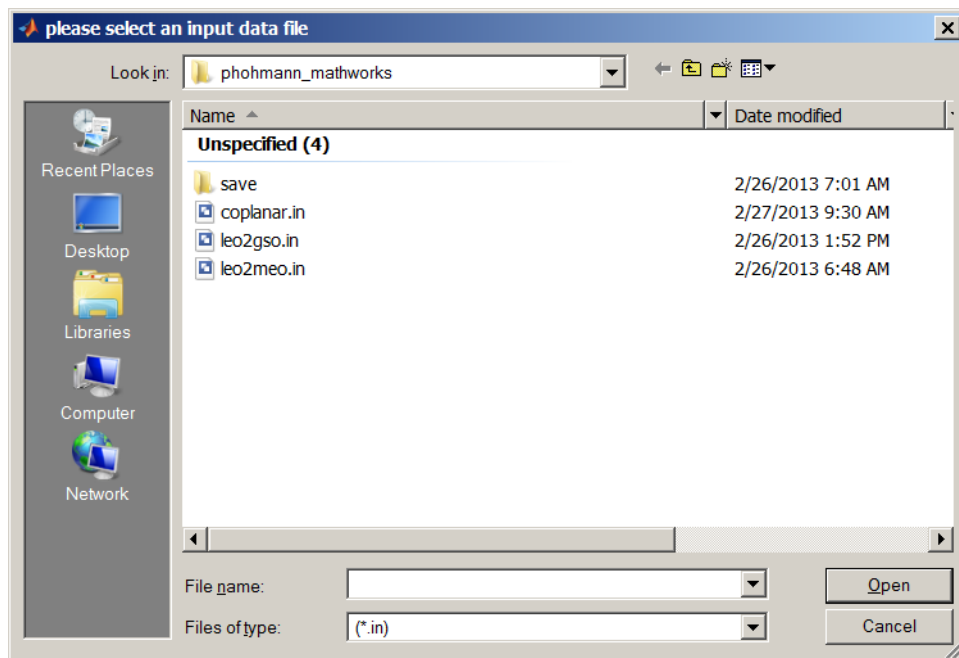
This document is the user's manual for a MATLAB script named `phohmann` which can be used to solve the gravity perturbed Hohmann transfer between coplanar and non-coplanar circular Earth orbits. The algorithm starts with a two-body Hohmann transfer initial guess and then uses the SNOPT nonlinear programming (NLP) method to determine the optimum two impulse orbit transfer subject to non-spherical Earth gravity perturbations. Appendix A summarizes the governing equations for the two-body Hohmann transfer. It also includes information about the MATLAB implementation used to solve this classic astrodynamics problem. This script is valid for "exterior" Hohmann transfers from a lower altitude circular orbit to a higher altitude circular orbit.

The `phohmann` script uses modified equinoctial orbital elements to solve the gravity perturbed orbit transfer "targeting" problem. Additional information about these orbital elements can be found in Appendix B. That appendix also explains how to use components and combinations of these non-singular elements to calculate a variety of final orbital element targets or boundary conditions.

MATLAB versions of SNOPT for several computer platforms can be found at Professor Philip Gill's web site which is located at <http://scicomp.ucsd.edu/~peg/>. Professor Gill's web site also includes a PDF version of the SNOPT software user's guide. A brief introduction to nonlinear programming can be found in Appendix C.

User interaction with the script

The `phohmann` MATLAB script will interactively prompt the user for the name of the simulation definition input data file. This prompt is similar to the following;



The file type defaults to names with a `*.in` filename extension. However, you can select any `phohmann` compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

Input file format and contents

The phohmann software is “data-driven” by a user-created text file. This text file should be simple ASCII format with no special characters.

The following is a typical input file used by this MATLAB script. In the following discussion the actual input file contents are in courier font and all explanations are in *times italic* font. This example is a Hohmann transfer from a low Earth orbit (LEO) to a geosynchronous orbit (GSO). In this data file, user provided inputs are in bold 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.

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

```
*****
* input data file for phohmann MATLAB script
* impulsive LEO-to-GSO orbital transfer
* filename ==> leo2gso.in
*****
```

The first inputs to the program define the initial UTC calendar date and time for the simulation. The data for the calendar year should include all four digits. The calendar date and time are required in order to correctly calculate the tesseral or longitude-dependent components of the Earth’s gravity.

```
initial calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
-----
2, 25, 2013

initial UTC
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
-----
20, 18, 33.0
```

The next three inputs are the altitude, orbital inclination and right ascension of the ascending node (RAAN) of the initial circular orbit.

```
*****
initial orbit
*****

altitude (kilometers; altitude > 0)
-----
185.32

orbital inclination (degrees; 0 <= inclination <= 180)
-----
28.5

right ascension of the ascending node (degrees; 0 <= raan <= 360)
-----
100.0
```

The next two inputs are the altitude and orbital inclination of the final circular orbit.

```
*****
final orbit
*****

altitude (kilometers; altitude > 0)
-----
35788.0955

orbital inclination (degrees; 0 <= inclination <= 180)
-----
0.0
```

The Earth gravitational constant and radius are user-defined by the next two inputs.

```
*****
astrodynamic constants and gravity model
*****

central body gravitational constant (km^3/sec^2)
-----
398600.4415

central body radius (kilometers)
-----
6378.14
```

Finally, the name of the Earth gravity model to use in the simulation and the order and degree of this model are set by the following three inputs.

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

order of the gravity model (zonals)
-----
4

degree of the gravity model (tesserals)
-----
4
```

The following is the phohmann solution for this example. The first part of the display is the two-body Hohmann transfer solution. The second section summarizes the SNOPT iterations and summary. The final section is the perturbed Hohmann transfer solution found during the optimization.

```
two-body Hohmann transfer solution

initial orbit altitude           185.3200 kilometers
initial orbit inclination        28.5000 degrees
initial orbit velocity           7792.9603 meters/second

final orbit altitude             35788.0955 kilometers
final orbit inclination           0.0000 degrees
final orbit velocity              3074.5848 meters/second

first inclination change          2.1645 degrees
```

Orbital Mechanics with MATLAB

```

second inclination change      26.3355 degrees
total inclination change      28.5000 degrees

first delta-v                 2481.9821 meters/second
second delta-v                1790.3423 meters/second
total delta-v                 4272.3244 meters/second

transfer orbit semimajor axis  24364.8478 kilometers
transfer orbit eccentricity    0.73061765
transfer orbit perigee velocity 10251.8686 meters/second
transfer orbit apogee velocity 1595.7727 meters/second
transfer orbit coast time      18924.5926 seconds
                                315.4099 minutes
                                5.2568 hours

```

orbital elements and state vector of the initial orbit

```

-----
          sma (km)          eccentricity          inclination (deg)          argper (deg)
+6.563460000000000e+03 +0.000000000000000e+00 +2.850000000000000e+01 +0.000000000000000e+00

          raan (deg)          true anomaly (deg)          arglat (deg)          period (min)
+1.000000000000000e+02 +0.000000000000000e+00 +0.000000000000000e+00 +8.81980522880484e+01

          rx (km)          ry (km)          rz (km)          rmag (km)
-1.13973286818979e+03 +6.46374629458551e+03 +0.000000000000000e+00 +6.563460000000000e+03

          vx (kps)          vy (kps)          vz (kps)          vmag (kps)
-6.74454148515196e+00 -1.18924463633983e+00 +3.71847929670569e+00 +7.79296034444086e+00

```

orbital elements and state vector of the transfer orbit after the first impulse

```

-----
          sma (km)          eccentricity          inclination (deg)          argper (deg)
+2.436484775000000e+04 +7.30617647713395e-01 +2.63354979218110e+01 +0.000000000000000e+00

          raan (deg)          true anomaly (deg)          arglat (deg)          period (min)
+1.000000000000000e+02 +0.000000000000000e+00 +0.000000000000000e+00 +6.30819751739617e+02

          rx (km)          ry (km)          rz (km)          rmag (km)
-1.13973286818979e+03 +6.46374629458550e+03 +0.000000000000000e+00 +6.563460000000000e+03

          vx (kps)          vy (kps)          vz (kps)          vmag (kps)
-9.04826108087521e+00 -1.59545255705264e+00 +4.54800087376082e+00 +1.02518685807620e+01

```

orbital elements and state vector of the transfer orbit prior to second impulse

```

-----
          sma (km)          eccentricity          inclination (deg)          argper (deg)
+2.436484775000000e+04 +7.30617647713395e-01 +2.63354979218110e+01 +0.000000000000000e+00

          raan (deg)          true anomaly (deg)          arglat (deg)          period (min)
+1.000000000000000e+02 +1.800000000000000e+02 +1.800000000000000e+02 +6.30819751739617e+02

          rx (km)          ry (km)          rz (km)          rmag (km)
+7.32208995364962e+03 -4.15256356357386e+04 +2.29083173533909e-12 +4.216623550000000e+04

          vx (kps)          vy (kps)          vz (kps)          vmag (kps)
+1.40842308946174e+00 +2.48342990924876e-01 -7.07927123702901e-01 +1.59577274464277e+00

```

Orbital Mechanics with MATLAB

orbital elements and state vector of the final orbit

```
-----
          sma (km)          eccentricity          inclination (deg)          argper (deg)
+4.216623550000000e+04 +0.000000000000000e+00 +0.000000000000000e+00 +0.000000000000000e+00

          raan (deg)          true anomaly (deg)          arglat (deg)          period (min)
+1.000000000000000e+02 +1.800000000000000e+02 +1.800000000000000e+02 +1.43617371924092e+03

          rx (km)          ry (km)          rz (km)          rmag (km)
+7.32208995364962e+03 -4.15256356357386e+04 +0.000000000000000e+00 +4.216623550000000e+04

          vx (kps)          vy (kps)          vz (kps)          vmag (kps)
+3.02787492676107e+00 +5.33896043798643e-01 -0.000000000000000e+00 +3.07458477809479e+00
```

initial delta-v vector, magnitude and steering angles

```
-----
x-component of delta-v          -2303.719596 meters/second
y-component of delta-v          -406.207921 meters/second
z-component of delta-v          829.521577 meters/second

delta-v magnitude          2481.982050 meters/second

pitch angle          -0.000000 degrees
yaw angle          -8.975045 degrees
```

final delta-v vector, magnitude and steering angles

```
-----
x-component of delta-v          1619.451837 meters/second
y-component of delta-v          285.553053 meters/second
z-component of delta-v          707.927124 meters/second

delta-v magnitude          1790.342317 meters/second

pitch angle          -0.000000 degrees
yaw angle          49.627319 degrees
```

Nonlinear constraints	5	Linear constraints	1
Nonlinear variables	6	Linear variables	0
Jacobian variables	6	Objective variables	6
Total constraints	6	Total variables	6

The user has defined 0 out of 36 first derivatives

Major	Minors	Step	nCon	Feasible	Optimal	MeritFunction	nS	Penalty		
0	5		1	5.7E+01	8.9E-06	4.2723244E+00	2		r	c
Major	Minors	Step	nCon	Feasible	Optimal	MeritFunction	nS	Penalty		
0	6		1	5.7E+01	8.9E-06	4.2723244E+00	2		r	c
1	2	1.0E+00	2	2.9E-01	1.3E-06	4.2749551E+00		1.3E-07	n r	c
2	2	1.1E-01	3	2.6E-01	1.0E-06	4.2749548E+00	2	1.3E-07	s	c
3	1	1.0E+00	4	6.7E-04	(2.4E-07)	4.2749546E+00	2	1.3E-07		c
4	1	1.0E+00	5	2.6E-05	(1.9E-10)	4.2749546E+00	2	1.3E-07		c
5	1	1.0E+00	6	(2.5E-11)	(9.7E-13)	4.2749546E+00	2	1.3E-07		c

SNOPTA EXIT 0 -- finished successfully

SNOPTA INFO 1 -- optimality conditions satisfied

Problem name

No. of iterations	13	Objective value	4.2749546090E+00
No. of major iterations	5	Linear objective	0.0000000000E+00
Penalty parameter	1.316E-07	Nonlinear objective	4.2749546090E+00
No. of calls to funobj	98	No. of calls to funcon	98
Calls with modes 1,2 (known g)	6	Calls with modes 1,2 (known g)	6
Calls for forward differencing	6	Calls for forward differencing	6
Calls for central differencing	72	Calls for central differencing	72
No. of superbasics	2	No. of basic nonlinears	4
No. of degenerate steps	0	Percentage	.00

Orbital Mechanics with MATLAB

```
Max x          1 2.3E+00    Max pi          5 2.5E+00
Max Primal infeas 0 0.0E+00    Max Dual infeas 5 2.9E-10
Nonlinear constraint violn 8.7E-11
```

gravity-perturbed Hohmann transfer solution

orbital elements and state vector of the initial orbit

```
-----
      sma (km)      eccentricity      inclination (deg)      argper (deg)
+6.563460000000000e+03 +0.000000000000000e+00 +2.850000000000000e+01 +0.000000000000000e+00

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+1.000000000000000e+02 +0.000000000000000e+00 +0.000000000000000e+00 +8.81980522880484e+01

      rx (km)      ry (km)      rz (km)      rmag (km)
-1.13973286818979e+03 +6.46374629458551e+03 +0.000000000000000e+00 +6.563460000000000e+03

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-6.74454148515196e+00 -1.18924463633983e+00 +3.71847929670569e+00 +7.79296034444086e+00
```

orbital elements and state vector of the transfer orbit after the initial delta-v

```
-----
      sma (km)      eccentricity      inclination (deg)      argper (deg)
+2.44593744898382e+04 +7.31658863355877e-01 +2.63335418348091e+01 +3.59907068451479e+02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+1.000000000000000e+02 +9.29315485206550e-02 +0.000000000000000e+00 +6.34494336505482e+02

      rx (km)      ry (km)      rz (km)      rmag (km)
-1.13973286818979e+03 +6.46374629458551e+03 +0.000000000000000e+00 +6.563460000000000e+03

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-9.05235339099006e+00 -1.58903790996246e+00 +4.549053771104437e+00 +1.02549516628076e+01
```

orbital elements and state vector of the transfer orbit prior to the final delta-v

```
-----
      sma (km)      eccentricity      inclination (deg)      argper (deg)
+2.43663822497766e+04 +7.30510103763810e-01 +2.63224626979287e+01 +4.49503355132814e-02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+9.99180887891227e+01 +1.79955049664487e+02 +1.800000000000000e+02 +6.30879346284528e+02

      rx (km)      ry (km)      rz (km)      rmag (km)
+7.26271659870086e+03 -4.15360610054494e+04 -4.54747350886464e-13 +4.216623550000000e+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
+1.40980173813377e+00 +2.43062653381369e-01 -7.07743262330976e-01 +1.59609552335964e+00
```

orbital elements and state vector of the final orbit

```
-----
      sma (km)      eccentricity      inclination (deg)      argper (deg)
+4.21662354999999e+04 +2.56314888882674e-15 +1.01379591428343e-13 +0.000000000000000e+00

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+0.000000000000000e+00 +2.79918088789123e+02 +2.79918088789123e+02 +1.43617371924091e+03

      rx (km)      ry (km)      rz (km)      rmag (km)
+7.26271659870086e+03 -4.15360610054494e+04 -4.54747350886465e-13 +4.216623550000000e+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
+3.02863510092978e+00 +5.29566788146922e-01 +5.44009282066327e-15 +3.07458477809479e+00
```

Orbital Mechanics with MATLAB

initial delta-v vector, magnitude and steering angles

```
-----  
x-component of delta-v      -2307.811906  meters/second  
y-component of delta-v      -399.793274  meters/second  
z-component of delta-v       830.574474  meters/second  
  
delta-v magnitude           2485.092435  meters/second  
  
pitch angle                 0.162032   degrees  
yaw angle                   -8.974637   degrees
```

final delta-v vector, magnitude and steering angles

```
-----  
x-component of delta-v      1618.833363  meters/second  
y-component of delta-v      286.504135  meters/second  
z-component of delta-v      707.743262  meters/second  
  
delta-v magnitude           1789.862174  meters/second  
  
pitch angle                 -0.108656   degrees  
yaw angle                   49.614537   degrees  
  
total delta-v               4274.954609  meters/second  
  
transfer time                18904.205381  seconds  
                             315.070090  minutes  
                             5.251168   hours  
  
degree of gravity model      4  
order of gravity model       4
```

The following is a brief summary of the information provided by this MATLAB script.

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 position vector in kilometers

ry (km) = y-component of the position vector in kilometers

rz (km) = z-component of the position vector in kilometers

rmag (km) = scalar magnitude of the position vector in kilometers

vx (km/sec) = x-component of the velocity vector in kilometers per second

vy (km/sec) = y-component of the velocity vector in kilometers per second

vz (km/sec) = z-component of the velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the velocity vector in kilometers per second

x-component of delta-v = ECI x-component of the impulsive delta-v maneuver in meters per second

y-component of delta-v = ECI y-component of the impulsive delta-v maneuver in meters per second

z-component of delta-v = ECI z-component of the impulsive delta-v maneuver in meters per second

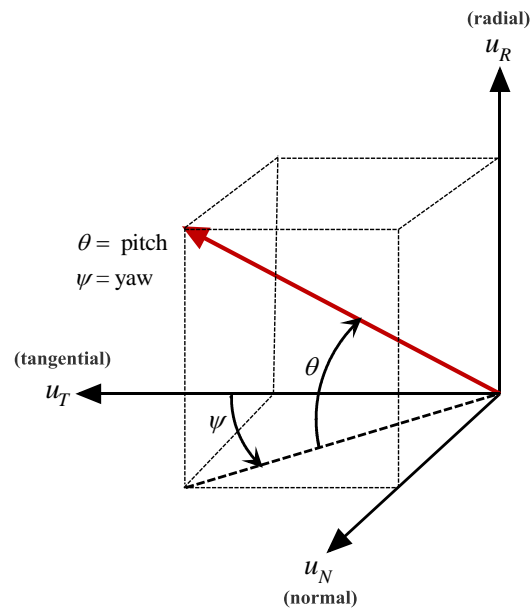
delta-v magnitude = scalar magnitude of the impulsive delta-v maneuver in meters per second

transfer time = time interval between the two impulsive maneuvers in seconds, minutes and hours

Note that ECI implies an Earth-centered-inertial coordinate system.

Additional information about the data displayed by the optimization algorithm can be found in the SNOPT user's manual which is available at Professor Gill's website which is located at <http://scicomp.ucsd.edu/~peg/>.

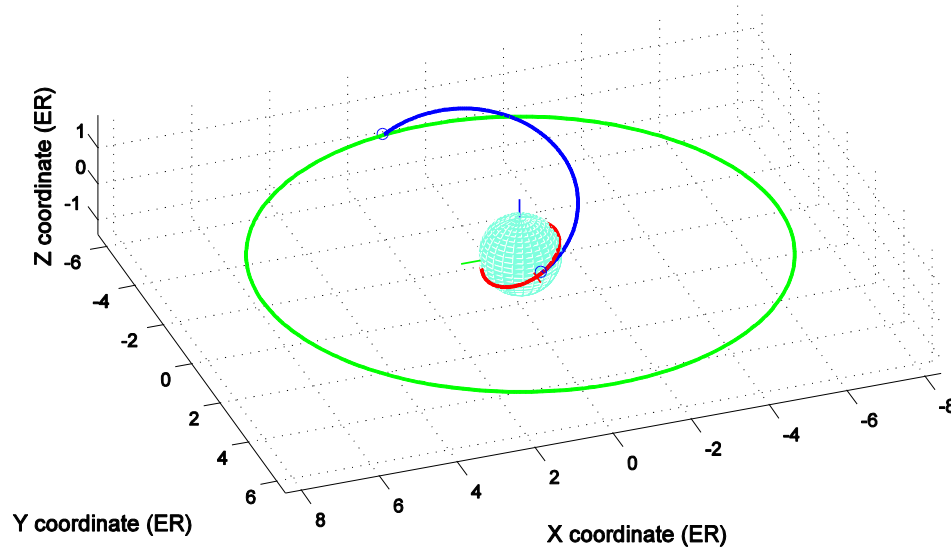
The pitch and yaw angles for each impulsive maneuver are computed and displayed in a local-vertical-local horizontal coordinate system. The following diagram illustrates the geometry of the pitch and yaw angles in this system. In this figure, the radial direction is along the geocentric radius vector directed away from the Earth, the tangential direction is tangent to the orbit in the direction of the orbital motion, and the normal direction is along the angular momentum vector of the orbit. The pitch angle is positive above the local horizontal plane formed by the tangential and normal directions, and the yaw angle is positive in the direction of the angular momentum vector which is perpendicular to the orbit plane.



The phohmann script will also create a graphics display of the initial, transfer and final orbits. The following is the graphics display for this example. The initial orbit trace is red, the transfer orbit is blue and the final mission orbit is green. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis. The location of each impulse is marked with a small blue circle.

Trajectory image files are saved to disk in both encapsulated, color Postscript format and MATLAB `fig` format. The disk file names are `phohmann1.eps` and `phohmann1.fig`. The interactive features of MATLAB graphics allow the user to re-load and manipulate the `fig` version of the trajectory display. These capabilities allow the user to interactively find the best viewpoint as well as verify basic three-dimensional geometry of the orbital maneuver.

Hohmann Transfer: Initial, Transfer and Final Orbits



Technical discussion

In this MATLAB script, the orbital motion is modeled with respect to a *true-of-date* Earth-centered-inertial (ECI) coordinate system. The origin of this system is the center of the Earth and the fundamental plane is the Earth's equator. The x -axis is aligned with the true-of-date Vernal Equinox, the z -axis is aligned with the Earth's spin axis, and the y -axis completes this orthogonal, right-handed coordinate system.

Acceleration due to Earth gravity

This MATLAB script uses a *spherical harmonic* representation of the Earth's geopotential function given by

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

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

The software calculates the 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 Φ , the inertial rectangular cartesian components of the spacecraft's acceleration vector are as follows

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

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

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

where

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

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

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

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

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

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

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

where the first few associated Legendre functions are given by

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

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

The trigonometric arguments are determined from expansions given by

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

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

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

The following are the first 14 lines of the 18 by 18 `egm96.dat` gravity model file included with this script. Column 1 is the degree l , column 2 is the order m , column 3 is the C coefficients and the last column contains the S gravity model coefficients.

2	0	-1.08262668355E-003	0.00000000000E+000
3	0	2.53265648533E-006	0.00000000000E+000
4	0	1.61962159137E-006	0.00000000000E+000
5	0	2.27296082869E-007	0.00000000000E+000
6	0	-5.40681239107E-007	0.00000000000E+000
7	0	3.52359908418E-007	0.00000000000E+000
8	0	2.04799466985E-007	0.00000000000E+000
9	0	1.20616967365E-007	0.00000000000E+000
10	0	2.41145438626E-007	0.00000000000E+000
11	0	-2.44402148325E-007	0.00000000000E+000
12	0	1.88626318279E-007	0.00000000000E+000
13	0	2.19788001661E-007	0.00000000000E+000
14	0	-1.30744533118E-007	0.00000000000E+000

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 δ_{m0} is equal to 1 if m is zero and equal to zero if m is greater than zero.

The following is the MATLAB source code for the function that opens and reads a gravity model file (fname) and creates matrices of the un-normalized coefficients.

```
function [cccoef, scoef] = readgm(fname)

% read gravity model data file

% input

% fname = name of gravity data file

% output

% ccoef, scoef = gravity model coefficients

% data file format (space delimited ascii)

% column 1 is the degree, column 2 is the order, column 3 are the C coefficients
% and the last column contains the S gravity model coefficients. For example,

% 2      0      -1.08262668355E-003      0.00000000000E+000
% 3      0       2.53265648533E-006      0.00000000000E+000
% 4      0       1.61962159137E-006      0.00000000000E+000

% Orbital Mechanics with MATLAB

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

% read the data file

gdata = dlmread(fname);

nrows = size(gdata, 1);

% initialize coefficients

idim = gdata(nrows, 1) + 1;

cccoef = zeros(idim, idim);

scoef = zeros(idim, idim);

% create gravity model coefficients

for n = 1:nrows

    i = gdata(n, 1);

    j = gdata(n, 2);

    ccoef(i + 1, j + 1) = gdata(n, 3);

    scoef(i + 1, j + 1) = gdata(n, 4);

end
```

Solving the trajectory optimization problem

As mentioned earlier, the trajectory optimization uses the two-body Hohmann transfer solution for the initial guess. For this problem, the components of the initial and final impulsive delta-v vector are control variables. The transfer time from the initial to the final impulse is also an “indirect” or implicit control variable as will be explained later. The objective or cost function for this problem is the sum of the scalar magnitude of the two impulses given by

$$f = \|\Delta \mathbf{v}_i\| + \|\Delta \mathbf{v}_f\|$$

This is the scalar value we want to minimize.

The SNOPT algorithm requires initial guesses (`xg`) for the six components of the delta-v vectors as well as lower (`xlwr`) and upper (`xupr`) bounds on each component. It also requires lower (`flow`) and upper (`fupp`) bounds on the objective function and any linear or nonlinear constraints.

The following is the MATLAB source code that sets up this information.

```
% initial guess for components of initial delta-v

xg(1) = vti(1) - vi(1);
xg(2) = vti(2) - vi(2);
xg(3) = vti(3) - vi(3);

% initial guess for components of final delta-v

xg(4) = vf(1) - vtf(1);
xg(5) = vf(2) - vtf(2);
xg(6) = vf(3) - vtf(3);

xg = xg';

% define lower and upper bounds for components of delta-v vectors (kilometers/second)

dvm = norm(xg(1:3));
for i = 1:1:3
    xlwr(i) = xg(i) - 0.01;
    xupr(i) = xg(i) + 0.01;
end

dvm = norm(xg(4:6));
for i = 4:1:6
    xlwr(i) = xg(i) - 0.01;
    xupr(i) = xg(i) + 0.01;
end

xlwr = xlwr';
xupr = xupr';

% bounds on objective function

flow(1) = 0.0d0;
fupp(1) = +Inf;

% enforce final modified equinoctial equality constraints

flow(2) = 0.0d0;
fupp(2) = 0.0d0;
```

```

flow(3) = 0.0d0;
fupp(3) = 0.0d0;

flow(4) = 0.0d0;
fupp(4) = 0.0d0;

flow(5) = 0.0d0;
fupp(5) = 0.0d0;

if (itarget <= 1.0d-8)

    % equatorial orbit constraint

    flow(6) = 0.0d0;
    fupp(6) = 0.0d0;

end

flow = flow';

fupp = fupp';

% read SNOPT specs file
snspec('snopt_specs.txt');

% solve the orbital TPBVP using SNOPT

snscreen on;

[x, f, inform, xmul, fmul] = snopt(xg, xlwr, xupr, flow, fupp, 'tpbvp');

```

The `tpbvp` MATLAB function defines the current value of the objective function and the mission constraints. The function that evaluates the Earth gravity model and the first order equations of motion is called `ceqm1`. Here's the source code for this function.

```

function ydot = ceqm1 (t, y)

% first order form of Cowell's equations of orbital motion

% version for ode45

% input

% t = current simulation time
% y = current eci state vector

% output

% ydot = eci acceleration vector

% Orbital Mechanics with MATLAB

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

% compute non-spherical gravity perturbations

aggrav = gravity(t, y);

% total acceleration vector

ydot = [ y(4)
         y(5)

```

```
y(6)
aggrav(1)
aggrav(2)
aggrav(3)];
```

Notice that the main script will read a SNOPT “specs” file named `snopt_specs.txt` which has the following contents.

```
Begin SNOPT options
  major iterations limit      50
  minor iterations limit     100
  derivative option          0
  major optimality tolerance 1.0d-6
  solution                   Yes
End SNOPT options
```

Additional information about this special file can be found in the SNOPT documentation.

The `tpbvp` objective function starts with the two-body solution for the position and velocity vectors of the maneuver on the initial orbit and the current value for the delta- v vector and numerically integrates the first-order form of the orbital equations to the descending node. At the descending node the software adds the current value of the second delta- v vector to the velocity vector of the transfer orbit to determine the velocity vector on the final mission orbit.

The position vector at the descending node and the resultant velocity vector are used to compute the current modified equinoctial orbital elements of the final mission orbit.

The following is the MATLAB source code within the `tpbvp` function that uses the “event finding” feature of the built-in `ode45` algorithm to predict the descending node conditions. The bound for the search time (`tend`) is 102% of the two-body Hohmann transfer time.

```
% set up options for ode45
options = odeset('RelTol', 1.0e-12, 'AbsTol', 1.0e-12, 'Events', @nc_event);

% solve for nodal crossing condition
rwrk = xi(1:3);
vwrk = xi(4:6);

% maximum search duration = 102% of two-body transfer time (seconds)
tend = 1.02 * tof;

[t, ysol, tevent, yevent, ie] = ode45(@ceqml, [0 tend], [rwrk vwrk], options);
```

The following is the MATLAB source code for descending node objective function `nc_event`. The computed value is simply the current z -component of the unit position vector.

```
function [value, isterminal, direction] = nc_event(t, y)

% nodal crossing event function

% required by phohmann.m

% Orbital Mechanics with MATLAB
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% z-component of the unit position vector
value = y(3) / norm(y(1:3));
isterminal = 1;
direction = -1;

```

Note that setting `direction = -1` ensures a descending node crossing since we want to search for values of the z-component of position that are decreasing. This implementation essentially treats the coast time between the maneuvers as an “indirect” or implicit control variable.

Targeting to the final mission orbit

This section summarizes the technique used to compute and enforce the nonlinear constraints that define the final mission orbit. Since the final orbit of the Hohmann transfer is circular, we enforce the following three modified equinoctial element nonlinear constraints

$$p_t - p_p = 0 \quad f_t - f_p = 0 \quad g_t - g_p = 0$$

In these three equations, the t subscript implies “target” or desired values and the p subscript implies values “predicted” by the optimization process. The targeted values are computed using the position and velocity vectors of the mission orbit determined during the two-body Hohmann transfer solution.

Here’s the MATLAB source code that evaluates these constraints.

```

% enforce semiparameter and circular orbit constraints (p, f = g = 0)
f(2) = mee_final(1) - mee_target(1);
f(3) = mee_final(2) - mee_target(2);
f(4) = mee_final(3) - mee_target(3);

```

If the final mission orbit is equatorial (inclination = 0), we want to enforce the following two constraints.

$$h_t - h_p = 0 \quad k_t - k_p = 0$$

If the orbital inclination of the final mission orbit is non-zero, we enforce the following single nonlinear mission constraint.

$$(h_t^2 + k_t^2) - (h_p^2 + k_p^2) = 0$$

Here’s the MATLAB source code that evaluates this constraint.

```

% enforce mission orbit inclination constraint(s)
if (itarget <= 1.0d-8)
    % equatorial orbit (h = k = 0)
    f(5) = mee_final(4);

```



```
f(6) = mee_final(5);  
  
else  
  
    % non-equatorial mission orbit (h^2 + k^2 = 0)  
  
    f(5) = (mee_final(4)^2 + mee_final(5)^2) - (mee_target(4)^2 + ...  
        mee_target(5)^2);  
  
end
```

Note that we are not enforcing the right ascension of the ascending node (RAAN) or the true anomaly of the final mission orbit.

Additional information about targeting with the modified equinoctial orbital elements can be found in Appendix B.

Algorithm resources

- (1) Walter Hohmann, *Die Erreichbarkeit der Himmelskorper*, Oldenbourg, Munich, 1925. Also, *The Attainability of Heavenly Bodies*, NASA Technical Translation F-44, 1960.
- (2) Jean-Pierre Marec, *Optimal Space Trajectories*, Elsevier, 1979.
- (3) R. P. Brent, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, 1972.
- (4) R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA, 1987.
- (5) D. F. Lawden, *Optimal Trajectories for Space Navigation*, Butterworths, London, 1963.
- (6) John E. Prussing, “Simple Proof of the Global Optimality of the Hohmann Transfer”, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 15, No. 4.
- (7) A. Miele, M. Ciarcia, and J. Mathwig, “Reflections on the Hohmann Transfer”, *Journal of Optimization Theory and Applications*, Vol. 123, No. 2, pp. 233-253, November 2004.

Appendix A

Two-body Hohmann Transfer

The coplanar circular orbit-to-circular orbit transfer was discovered by the German engineer Walter Hohmann in 1925 and described in his classic report, *The Attainability of Celestial Bodies*. The transfer consists of a velocity impulse on an initial circular orbit, in the direction of motion and collinear with the velocity vector, which propels the space vehicle into an elliptical transfer orbit. At a transfer angle of 180 degrees from the first impulse, a second velocity impulse or ΔV , also collinear and in the direction of motion, places the vehicle into a final circular orbit at the desired final altitude. The impulsive ΔV assumption means that the velocity, but not the position, of the vehicle is changed instantaneously. This is equivalent to a rocket engine with infinite thrust magnitude. Therefore, the Hohmann formulation is the ideal and minimum energy solution to this type of orbit transfer problem.

Coplanar Equations

For the coplanar Hohmann transfer both velocity impulses are confined to the orbital planes of the initial and final orbits. For a Hohmann transfer from a lower altitude orbit to a higher altitude circular orbit, the first impulse creates an elliptical transfer orbit with a perigee altitude equal to the altitude of the initial circular orbit and an apogee altitude equal to the altitude of the final orbit. The second impulse circularizes the transfer orbit at apogee. Both impulses are *prograde* which means that they are in the direction of orbital motion.

We begin by defining three *normalized* radii as follows:

$$R_1 = \sqrt{2 \frac{r_f}{r_i + r_f}} \quad R_2 = \sqrt{\frac{r_i}{r_f}} \quad R_3 = \sqrt{2 \frac{r_i}{r_i + r_f}}$$

where r_i is the geocentric radius of the initial circular park orbit and r_f is the radius of the final circular mission orbit. The relationship between radius and initial orbit altitude h_i and the final orbit altitude h_f is as follows:

$$r_i = r_e + h_i$$

$$r_f = r_e + h_f$$

where r_e is the radius of the Earth.

The magnitude of the first impulse is

$$\Delta V_1 = V_{lc} \sqrt{1 + R_1^2 - 2R_1}$$

and is simply the difference between the speed on the initial orbit and the perigee speed of the transfer orbit. The scalar magnitude of the second impulse is

$$\Delta V_2 = V_{lc} \sqrt{R_2^2 + R_2^2 R_3^2 - 2R_2^2 R_3}$$

which is the difference between the speed on the final orbit and the apogee speed of the transfer ellipse.

In each of these ΔV equations V_{lc} is called the *local circular velocity*. It can be determined from

$$V_{lc} = \sqrt{\mu/r_i}.$$

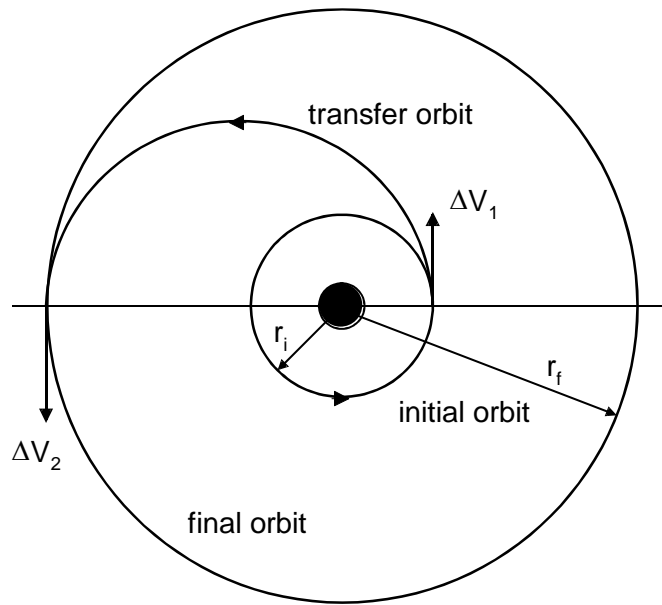
and represents the scalar speed in the initial orbit. In these equations μ is the gravitational constant of the central body. The transfer time from the first impulse to the second is equal to one half the orbital period of the transfer ellipse

$$\tau = \pi \sqrt{\frac{a^3}{\mu}}$$

where a is the semimajor axis of the transfer orbit and is equal to $(r_i + r_f)/2$. The orbital eccentricity of the transfer ellipse is

$$e = \frac{\max(r_i, r_f) - \min(r_i, r_f)}{r_f - r_i}$$

The following diagram illustrates the geometry of the coplanar Hohmann transfer.



Non-coplanar Equations

The non-coplanar Hohmann transfer involves orbital transfer between two circular orbits which have different orbital inclinations. For this problem the propulsive energy is minimized if we optimally partition the total orbital inclination change between the first and second impulses.

The scalar magnitude of the first impulse is

$$\Delta V_1 = V_{lc} \sqrt{1 + R_1^2 - 2R_1 \cos \theta_1}$$

where θ_1 is the plane change associated with the first impulse. The magnitude of the second impulse is

$$\Delta V_2 = V_{ic} \sqrt{R_2^2 + R_2^2 R_3^2 - 2R_2^2 R_3 \cos \theta_2}$$

where θ_2 is the plane change associated with the second impulse. These two equations are different forms of the law of cosines.

The total ΔV required for the maneuver is the sum of the two impulses as follows

$$\Delta V = \Delta V_1 + \Delta V_2$$

The relationship between the plane change angles is

$$\theta_t = \theta_1 + \theta_2$$

where θ_t is the total plane change angle between the initial and final orbits.

Optimizing the non-coplanar Hohmann transfer involves allocating the total plane change angle between the two maneuvers such that the total ΔV required for the mission is minimized. We can determine this answer by solving for the root of a derivative.

The partial derivative of the total ΔV with respect to the first plane change angle is given by:

$$\frac{\partial \Delta V}{\partial \theta_1} = \frac{R_1 \sin \theta_1}{\sqrt{1 + R_1^2 - 2R_1 \cos \theta_1}} - \frac{R_2^2 R_3 (\sin \theta_t \cos \theta_1 - \cos \theta_t \sin \theta_1)}{\sqrt{R_2^2 + R_2^2 R_3^2 - 2R_2^2 R_3 \cos(\theta_t - \theta_1)}}$$

If we determine the value of θ_1 which makes this derivative zero, we have found the optimum plane change required at the first impulse. Once θ_1 is calculated we can determine θ_2 from the total plane change angle relationship and the velocity impulses from the previous equations.

Numerical Solution

This numerical algorithm has been implemented in an interactive MATLAB script called `hohmann.m`. This script prompts the user for the initial and final altitudes in kilometers and the initial and final orbital inclinations in degrees. The software then calls the Brent root-finding algorithm to solve the partial derivative equation described above.

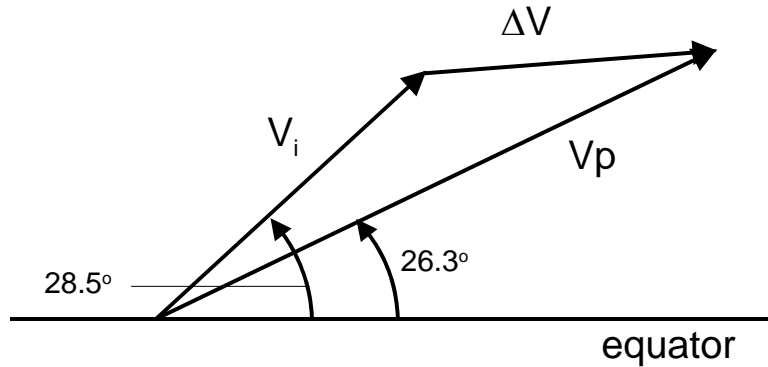
The call to the Brent root-finding algorithm is as follows:

```
[xroot, froot] = brent('hohmfunc', 0, dinc, rtol);
```

where `hohmfunc` is the objective function for this problem. Since we know that the optimum first plane change angle is somewhere between 0 and the total plane change angle `dinc`, we pass these as the bounds of the root. In the parameter list `rtol` is the user-defined root-finding convergence tolerance.

The following discussion pertains to a typical orbit transfer from a low altitude Earth orbit (LEO) at an altitude of 185.2 kilometers and an orbital inclination of 28.5 degrees to a geosynchronous Earth orbit (GSO) at an altitude of 35786.36 kilometers and 0 degrees inclination.

The following is a ΔV diagram for the first maneuver of this orbit transfer example. In this view we are looking along the line of nodes which is the mutual intersection of the park and transfer orbit planes with the equatorial plane.



In this diagram V_i is the speed on the initial park orbit, V_p is the perigee speed of the elliptic transfer orbit, and ΔV is the impulse required for the first maneuver. The inclinations of the park and transfer orbit are also labeled. From this geometry and the law of cosines, the required ΔV is given by

$$\Delta V = \sqrt{V_i^2 + V_p^2 - 2V_i V_p \cos \Delta i}$$

where Δi is the inclination difference or plane change between the park and transfer orbits.

Algorithm resources

- (1) Walter Hohmann, *Die Erreichbarkeit der Himmelskörper*, Oldenbourg, Munich, 1925. Also, *The Attainability of Heavenly Bodies*, NASA Technical Translation F-44, 1960.
- (2) John E. Prussing, "Simple Proof of the Global Optimality of the Hohmann Transfer", *AIAA Journal of Guidance, Control and Dynamics*, Vol. 15, No. 4.
- (3) A. Miele, M. Ciarcia, and J. Mathwig, "Reflections on the Hohmann Transfer", *Journal of Optimization Theory and Applications*, Vol. 123, No. 2, pp. 233-253, November 2004.

Appendix B

Targeting with Modified Equinoctial Orbital Elements

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 *direct* modified equinoctial equations exhibit no singularity for zero eccentricity and orbital inclinations equal to 0 and 90 degrees. However, please note that two of the components are singular for an orbital inclination of 180 degrees.

The classic reference for these elements is “A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

The modified equinoctial elements are defined in terms of the classical orbital elements as follows:

$$p = a(1 - e^2)$$

$$f = e \cos(\omega + \Omega)$$

$$g = e \sin(\omega + \Omega)$$

$$h = \tan(i/2) \cos \Omega$$

$$k = \tan(i/2) \sin \Omega$$

$$L = \theta + \omega + \Omega$$

where

p = semiparameter

a = semimajor axis

e = orbital eccentricity

i = orbital inclination

ω = argument of perigee

Ω = right ascension of the ascending node

θ = true anomaly

L = true longitude

The classical orbital elements can be recovered from the modified equinoctial orbital elements with

semimajor axis

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

orbital eccentricity

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

orbital inclination

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

argument of periapsis

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

$$\sin \omega = \frac{g h - f k}{e \tan(i/2)}$$

$$\cos \omega = \frac{f h + g k}{e \tan(i/2)}$$

right ascension of the ascending node

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

$$\sin \Omega = \frac{k}{\tan(i/2)}$$

$$\cos \Omega = \frac{h}{\tan(i/2)}$$

true anomaly

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

$$\sin \theta = \frac{1}{e} (f \sin L - g \cos L)$$

$$\cos \theta = \frac{1}{e} (f \cos L + g \sin L)$$

In these expressions, an inverse tangent expression of the form $\theta = \tan^{-1}(a, b)$ denotes a four quadrant evaluation where $a = \sin \theta$ and $b = \cos \theta$.

Constraint formulations that enforce both the sine and cosine of a desired orbital element should be used whenever possible. This approach involves a combination of equality and inequality constraints and ensures that the “targeted” orbital element is in the correct quadrant.

To illustrate this technique, here are several examples for different values of argument of perigee and the corresponding mission constraints:

$$0^\circ < \omega < 90^\circ \rightarrow \begin{cases} \sin \omega > 0 \rightarrow gh - fk > 0 \\ fh + gk = e \tan(i/2) \cos \omega \end{cases}$$

$$\omega = 270^\circ \rightarrow \begin{cases} \sin \omega \leq 0 \rightarrow gh - fk \leq 0 \\ \cos \omega = 0 \rightarrow fh + gk = 0 \end{cases}$$

$$\omega = 178^\circ \rightarrow \begin{cases} gh - fk = e \tan(i/2) \sin \omega \\ \cos \omega \leq 0 \rightarrow fh + gk \leq 0 \end{cases}$$

The following is a *sign* table of the sine and cosine for each quadrant.

quadrant	sine	cosine
1	+	+
2	+	−
3	−	−
4	−	+

orbital eccentricity constraint

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

For a circular orbit, $f = g = 0$.

orbital inclination constraint

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

For an equatorial orbit, $h = k = 0$.

argument of perigee constraints

$$gh - fk = e \sin \omega \tan(i/2) \rightarrow \sin \omega = \frac{gh - fk}{e \tan(i/2)}$$

$$fh + gk = e \cos \omega \tan(i/2) \rightarrow \cos \omega = \frac{fh + gk}{e \tan(i/2)}$$

right ascension of the ascending node constraints

$$k = \tan(i/2) \sin \Omega \rightarrow \sin \Omega = \frac{k}{\tan(i/2)}$$

$$h = \tan(i/2) \cos \Omega \rightarrow \cos \Omega = \frac{h}{\tan(i/2)}$$

true anomaly constraints

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

In general,

$$\sin \theta = \frac{1}{e} (f \sin L - g \cos L)$$

$$\cos \theta = \frac{1}{e} (f \cos L + g \sin L)$$

For a circular orbit,

$$\sin \theta = \sin L \cos \Omega - \cos L \sin \Omega$$

$$\cos \theta = \cos L \cos \Omega + \sin L \sin \Omega$$

For a circular, equatorial orbit,

$$\theta = L, \sin \theta = \sin L \text{ and } \cos \theta = \cos L.$$

Also, note that

$$r_p = \frac{p}{1 + \sqrt{f^2 + g^2}} \rightarrow \text{periapsis radius}$$

$$r_a = \frac{p}{1 - \sqrt{f^2 + g^2}} \rightarrow \text{apoapsis radius}$$

Algorithm resources

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, August 20-22, 1990.

Appendix C

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 font.

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 $\mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), t] = 0$.

For any mission time t there are also simple bounds on the state variables

$$\mathbf{y}_l \leq \mathbf{y}(t) \leq \mathbf{y}_u$$

the control variables

$$\mathbf{u}_l \leq \mathbf{u}(t) \leq \mathbf{u}_u$$

and the problem parameters

$$\mathbf{p}_l \leq \mathbf{p}(t) \leq \mathbf{p}_u$$

The basic nonlinear programming problem (NLP) involves the determination of 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.

Algorithm resources

- (1) “Direct Trajectory Optimization Using Nonlinear Programming and Collocation”, C. R. Hargraves and S. W. Paris, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 10, No. 4, July-August, 1987, pp. 338-342.
- (2) “Optimal Finite-Thrust Spacecraft Trajectories Using Direct Transcription and Nonlinear Programming”, Paul J. Enright, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1991.
- (3) “Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.
- (4) “Improved Collocation Methods with Application to Direct Trajectory Optimization”, Albert L. Herman, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1995.
- (5) “Survey of Numerical Methods for Trajectory Optimization”, John T. Betts, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998, pp. 193-207.
- (6) *Practical Optimization*, Philip E. Gill, Walter Murray and Margaret H. Wright, Emerald Group Publishing Limited, 1982.

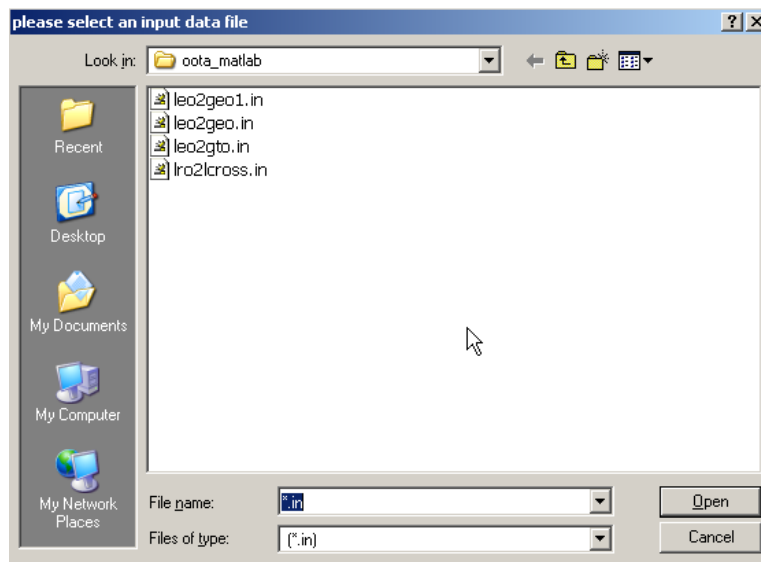
Optimal Impulsive Orbital Transfer

This document describes a MATLAB script called `oota_matlab` that can be used to determine optimum one and two impulse orbital transfers between *non-coplanar* circular and elliptical orbits. The initial and final orbits need not be co-apsidal. The numerical algorithm is based on the orbit transfer and rendezvous work of Gary McCue, Gentry Lee and David Bender, described in “Numerical Investigation of Minimum Impulse Orbital Transfer”, *AIAA Journal*, **3**, 2328-2334 (1965), and “An Analysis of Two-Impulse Orbital Transfer”, *AIAA Journal*, **2**, 1767-1773, October 1964.

The numerical solution of this classic astrodynamics problem involves a combination of one-dimensional root-finding using Brent’s derivative-free method and multi-dimensional unconstrained minimization using the built-in `fminsearch` algorithm provided with MATLAB. The `oota_matlab` MATLAB script uses primer vector theory to determine the optimality of the solution(s) computed by this numerical method.

Interacting with the script

To execute the `oota_matlab` script, log into the directory containing the source code and type `oota_matlab` in the MATLAB command window. This MATLAB script is “data driven” by a simple text file created by the user. The script will prompt the user for the name of the data file with a screen similar to



The file type defaults to names with a `*.in` filename extension. However, you can select any `oota_matlab.m` compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

Data file format and contents

The `oota_matlab` script reads a simple ASCII data file that defines the initial and final orbits along with the algorithm search characteristics. The following is a typical data file named `leo2gso.in` for this application. This example solves the problem of two impulse, non-coplanar orbital transfer from a typical low altitude circular Earth orbit (LEO) to a circular geosynchronous Earth orbit (GSO).

The annotation text in this file can be modified but should not be deleted because the MATLAB function that reads this data (read_oota.m) expects to find exactly 81 lines of text and numeric information. The first two data items define the gravitational constant and radius of the central body. Please note the units and valid range for each input. User inputs are shown in bold font.

```
*****
* input data file for oota_matlab.m MATLAB script
* impulsive LEO-to-GSO orbital transfer
* filename ==> leo2gso.dat
*****
```

```
central body gravitational constant (km^3/sec^2)
398600.4415
```

```
central body radius (kilometers)
6378.1363
```

```
*****
initial orbit
*****
```

```
semimajor axis (kilometers)
(semimajor axis > 0)
6653.14
```

```
orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.0
```

```
orbital inclination (degrees)
(0 <= inclination <= 180)
28.5
```

```
argument of perigee (degrees)
(0 <= argument of perigee <= 360)
0
```

```
right ascension of the ascending node (degrees)
(0 <= raan <= 360)
30
```

```
*****
final orbit
*****
```

```
semimajor axis (kilometers)
(semimajor axis > 0)
42166.2355
```

```
orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0
```

```
orbital inclination (degrees)
(0 <= inclination <= 180)
0
```

```
argument of perigee (degrees)
(0 <= argument of perigee <= 360)
0
```

```
right ascension of the ascending node (degrees)
(0 <= raan <= 360)
0
```

```

*****
algorithm search parameters
*****

initial orbit true anomaly at which to begin the search (degrees)
0

final orbit true anomaly at which to begin the search (degrees)
0

initial orbit true anomaly search increment (degrees)
30

final orbit true anomaly search increment (degrees)
30

number of initial orbit true anomaly search intervals
12

number of final orbit true anomaly search intervals
12

```

The last section of this data file defines the algorithm grid search parameters to use during the optimization. These numbers define the initial true anomaly for the initial and final orbits, the true anomaly search increment for each orbit, and the total number of intervals to analyze. Notice the combination of true anomaly search increments and number of search intervals in this example will encompass the entire true anomaly range for both the initial and final orbits.

Solution information

The `oota_matlab` MATLAB script provides the following types of information about the solution(s).

1. text file summarizing all solutions
2. text file of detailed information about all solutions
3. graphics file of three-dimensional trajectories
4. graphics file of the primer vector characteristics

The following is the summary text file for the `leo2gso` example described in the previous section. The argument of latitude is the sum of the argument of perigee and true anomaly at the impulse location. It describes the location of a maneuver relative to the ascending node of the initial and transfer orbit. For equatorial orbits, the argument of perigee is measured relative to the x-axis of the Earth-centered-inertial (ECI) coordinate system.

input data file ==> `leo2gso.in`

oota solution number	dv1 true anomaly (degrees)	dv2 true anomaly (degrees)	dv1 arg latitude (degrees)	dv2 arg latitude (degrees)	delta-v1 magnitude (m/s)	delta-v2 magnitude (m/s)	total delta-v (m/s)
-----	-----	-----	-----	-----	-----	-----	-----
1	0.0060	179.9979	0.0060	180.0000	2456.4991	1783.6790	4240.1782
2	0.0064	179.9977	0.0064	180.0000	2456.5011	1783.6771	4240.1782
3	0.0065	179.9977	0.0065	180.0000	2456.4997	1783.6784	4240.1782
4	0.0054	179.9981	0.0054	180.0000	2456.5004	1783.6777	4240.1782
5	180.0050	179.9982	180.0050	0.0000	2456.4992	1783.6790	4240.1781
6	180.0064	179.9977	180.0064	0.0000	2456.5011	1783.6771	4240.1782
7	0.0078	179.9973	0.0078	180.0000	2456.5015	1783.6767	4240.1782
8	0.0074	179.9974	0.0074	180.0000	2456.4990	1783.6793	4240.1782
9	180.0080	179.9972	180.0080	0.0000	2456.4999	1783.6783	4240.1782
10	180.0058	179.9980	180.0058	0.0000	2456.5011	1783.6771	4240.1782

Summary text files are saved to disk with a file name consisting of the name of the input data file (minus the filename extension) concatenated with `_summary.txt`. For example, the name of the summary text file for this case is `leo2gso_summary.txt`.

The following is the first solution contained in the detailed solutions text file. The initial part of this file contains the name of the simulation definition input data file, the solution number and information about the minimization algorithm performance.

input data file ==> leo2gso.in

solution number 1

number of function evaluations 286

number of iterations 136

Optimization terminated:

the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-08
and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-08

initial orbit - prior to the first impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+6.653140000000000e+03	+5.53905824337510e-16	+2.850000000000000e+01	+0.000000000000000e+00
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+3.000000000000000e+01	+6.02978812312732e-03	+6.02978812312732e-03	+9.00118574397072e+01
rx (km)	ry (km)	rz (km)	rmag (km)
+5.76148056050912e+03	+3.32710286869169e+03	+3.34094235711526e-01	+6.65313999999999e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-3.40184196051086e+00	+5.89053395030101e+00	+3.69333290837307e+00	+7.74026013232195e+00

transfer orbit - after the first impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.44096861143452e+04	+7.27438527447300e-01	+2.63076547250235e+01	+2.12842711764707e-03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.99994795350325e+01	+4.36350638641582e-03	+6.49193350406289e-03	+6.32561888727922e+02
rx (km)	ry (km)	rz (km)	rmag (km)
+5.76148056050912e+03	+3.32710286869169e+03	+3.34094235712489e-01	+6.65314000000000e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-4.56040306954073e+00	+7.89736266175169e+00	+4.50866270516224e+00	+1.01731830100921e+01

transfer orbit - prior to the second impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.44096877431762e+04	+7.27438549042367e-01	+2.63076547250235e+01	+2.12842723128193e-03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.99994795350325e+01	+1.79997871572769e+02	+1.80000000000000e+02	+6.32561952043141e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-3.65172226388517e+04	-2.10827860347250e+04	+2.28858249950193e-12	+4.21662355000000e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
+7.19306020021783e-01	-1.24621899990843e+00	-7.11392988206719e-01	+1.60516134075115e+00

Orbital Mechanics with MATLAB

final orbit - after the second impulse

```
-----  
          sma (km)          eccentricity          inclination (deg)          argper (deg)  
+4.21662355000000e+04 +1.57009245868378e-16 +0.00000000000000e+00 +0.00000000000000e+00  
  
          raan (deg)          true anomaly (deg)          arglat (deg)          period (min)  
+0.00000000000000e+00 +2.09999479535032e+02 +2.09999479535032e+02 +1.43617371924092e+03  
  
          rx (km)          ry (km)          rz (km)          rmag (km)  
-3.65172226388517e+04 -2.10827860347250e+04 -0.00000000000000e+00 +4.21662355000000e+04  
  
          vx (kps)          vy (kps)          vz (kps)          vmag (kps)  
+1.53726820176287e+00 -2.66268248830778e+00 -0.00000000000000e+00 +3.07458477809479e+00
```

ECI delta-v vectors, magnitudes and LVLH angles

```
-----  
  
delta-v1x          -1158.5611 meters/second  
delta-v1y          2006.8287 meters/second  
delta-v1z          815.3298 meters/second  
  
delta-v1          2456.4991 meters/second  
  
LVLH pitch angle          0.0076 degrees  
LVLH yaw angle          9.1154 degrees  
  
delta-v2x          817.9622 meters/second  
delta-v2y          -1416.4635 meters/second  
delta-v2z          711.3930 meters/second  
  
delta-v2          1783.6790 meters/second  
  
LVLH pitch angle          -0.0051 degrees  
LVLH yaw angle          310.1870 degrees  
  
total delta-v          4240.1782 meters/second  
  
transfer time          18975.8310 seconds  
                      316.2639 minutes  
                      5.2711 hours
```

Detailed text files are saved to disk with a file name consisting of the name of the input data file (minus the filename extension) concatenated with `_solutions.txt`. For example, the name of the summary text file for this test case is `leo2gso_solutions.txt`.

The following is a brief description of the information provided in the detailed text file.

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

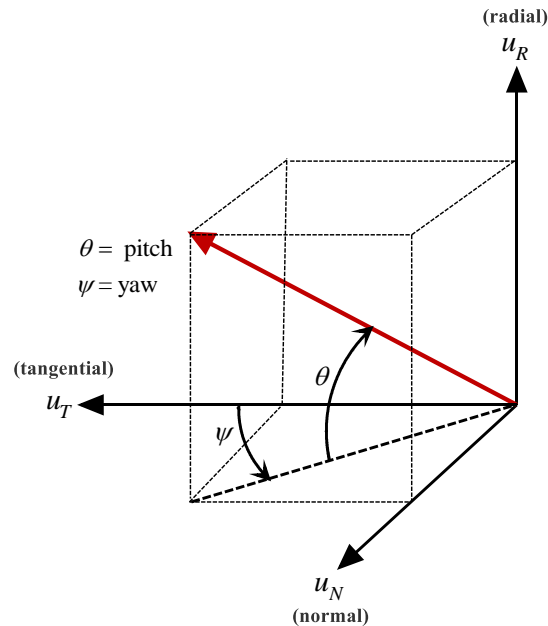
arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (mins) = orbital period in minutes

rx (km) = x-component of the position vector in kilometers

ry (km) = y-component of the position vector in kilometers
rz (km) = z-component of the position vector in kilometers
rmag (km) = scalar magnitude of the position vector in kilometers
vx (kps) = x-component of the velocity vector in kilometers per second
vy (kps) = y-component of the velocity vector in kilometers per second
vz (kps) = z-component of the velocity vector in kilometers per second
vmag (kps) = scalar magnitude of the velocity vector in kilometers per second
transfer time = flight time between the two impulses in seconds, minutes and hours

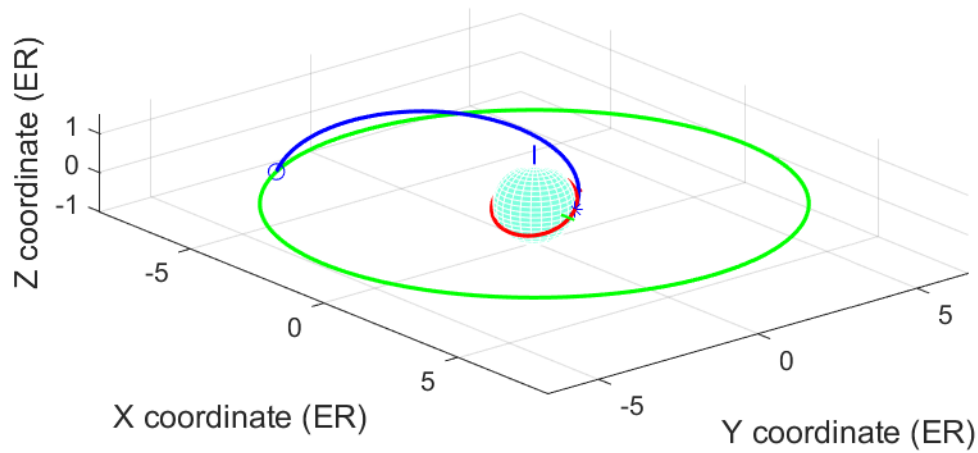
The pitch and yaw angles for each impulsive maneuver are computed and displayed in the local-vertical-local horizontal (LVLH; also called the radial-tangential-normal) coordinate system. The following diagram illustrates the geometry of the pitch and yaw angles in this system. In this figure, the radial direction is along the geocentric radius vector directed away from the Earth, the tangential direction is tangent to the orbit, and the normal direction is along the angular momentum vector of the orbit.



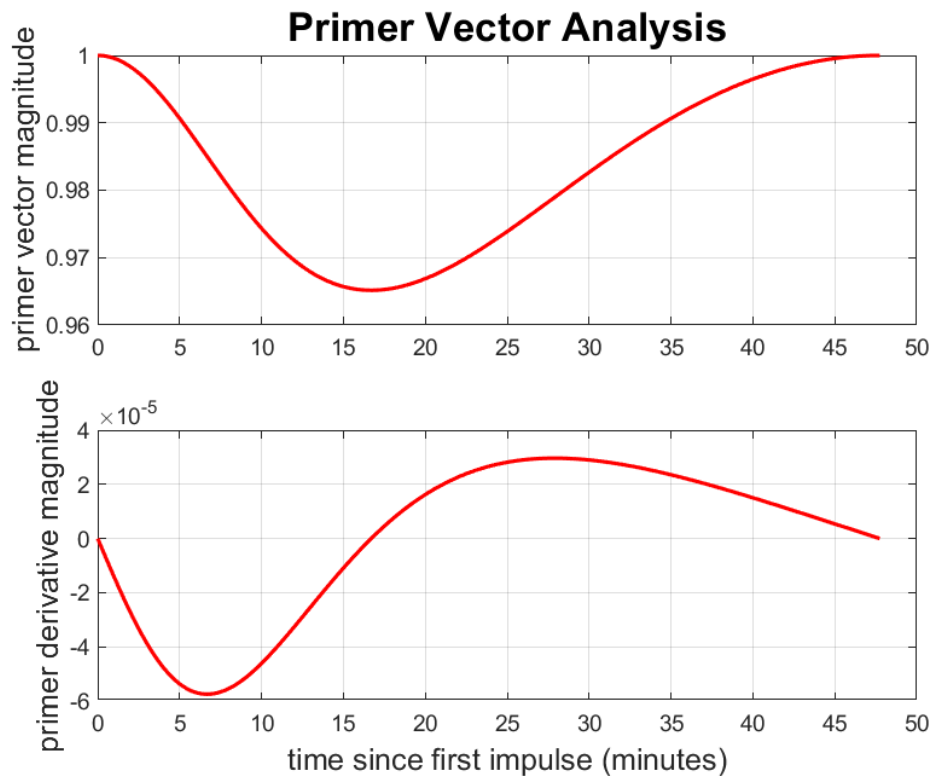
The `oota_matlab` script will also create a graphics display of the initial, transfer and final orbits for each solution. The following is one of the graphic images for this example. The initial orbit trace is red, the transfer orbit is blue and the final mission orbit is green. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis. The initial impulse location is marked with a small blue asterisk.

Trajectory image files are saved to disk in both TIF format and MATLAB `fig` format with a file name consisting of the name of the input data file (minus the filename extension) concatenated with `_traj`, the solution number and either `.tif` or `.fig`. For example, the names of the graphics disk files for the first solution are `leo2gso_traj1.tif` and `leo2gso_traj1.fig`. The interactive features of MATLAB graphics allow the user to manipulate the `fig` version of the trajectory display. This allows the user to interactively find the best viewpoint as well as verify the basic three-dimensional geometry of the orbital transfer.

Initial, Transfer and Final Orbits



The `oota_matlab` script will also create a graphics disk file of the primer vector and its derivative for each solution. The image for one of the solutions for this example is shown below. These plots illustrate the behavior of the scalar magnitudes of the primer vector and its derivative as a function of elapsed time since the first impulse.



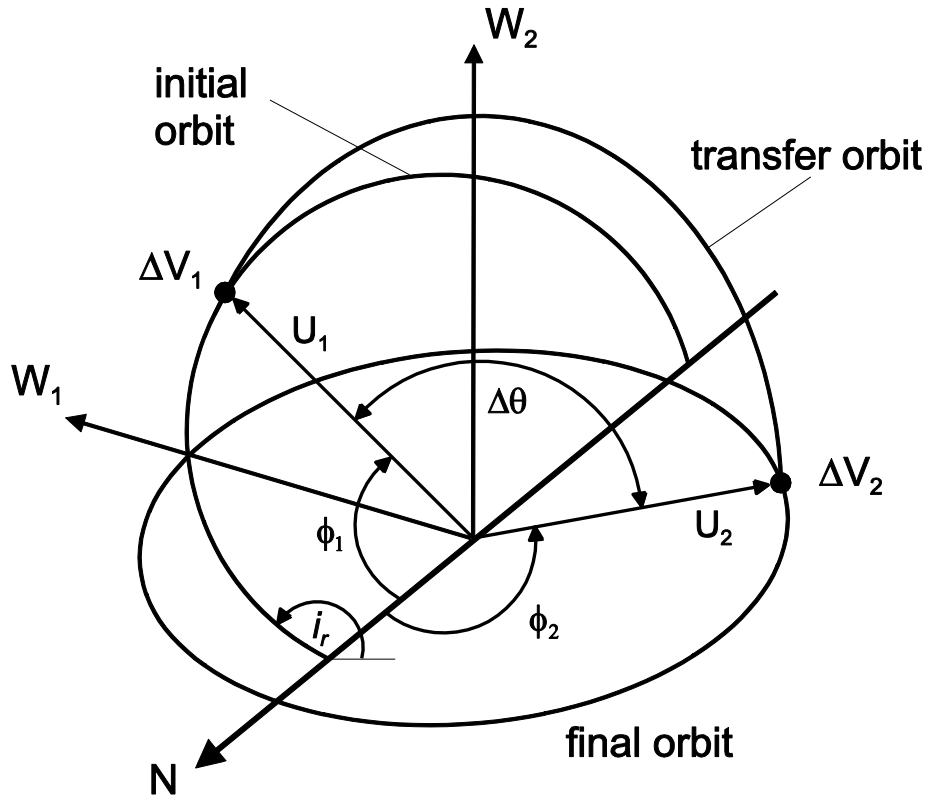
Primer image files are saved to disk with a file name consisting of the name of the input data file (minus the filename extension) concatenated with `_primer`, the solution number and `.tif`. For example, the name of the graphics file for the first solution is `leo2gso_primer1.tif`. Please note these files are also saved in TIF format.

The type of minimization algorithm and grid search implemented in this MATLAB script will often produce duplicate solutions to a particular orbit transfer problem. The summary and detailed text files allow the user to eliminate these duplicate solutions and retain one or more unique solutions.

Technical Discussion

The solution to this important astrodynamics problem is formulated in a *reference coordinate system*. The fundamental reference plane of this coordinate system is the final orbit plane and the x-axis is aligned with the intersection of the planes of the initial and final orbits. The z-axis of this system is aligned with the angular momentum vector of the final orbit and the y-axis completes this orthogonal coordinate system. In the equations which follow, elements of the initial orbit have a subscript of 1 and elements of the final orbit a subscript of 2 . Elements of the transfer orbit will have a subscript of t .

The following diagram illustrates the geometry of a two impulse orbital transfer. The relative inclination between the initial and final orbit planes is i_r and $\Delta\theta$ is the transfer angle which is the angle from the first and second impulse measured in the plane of the transfer orbit. \mathbf{N} corresponds to the x-axis, \mathbf{W}_1 is in the direction of the initial orbit angular momentum vector, and \mathbf{W}_2 is in the direction of the angular momentum vector of the final orbit.



The independent variables for this problem are ϕ_1 , ϕ_2 and p_t , where ϕ_1 is the angle from the \mathbf{N} axis to the first impulse as measured in the initial orbit plane, ϕ_2 is the angle from the \mathbf{N} axis to the second impulse as measured in the final orbit plane, and p_t is the semi-parameter of the transfer orbit. The expression for \mathbf{N} is as follows

$$\mathbf{N} = \frac{\mathbf{W}_2 \times \mathbf{W}_1}{|\mathbf{W}_2 \times \mathbf{W}_1|}$$

where $\mathbf{W}_1 = [0 \quad -\sin i_r \quad \cos i_r]^T$ and $\mathbf{W}_2 = [0 \quad 0 \quad 1]^T$.

The relative inclination between the initial and final orbit planes is determined from

$$i_r = \cos^{-1}(\mathbf{w}_1 \bullet \mathbf{w}_2)$$

where \mathbf{w}_1 is the ECI unit angular momentum vector of the initial orbit given by

$$\mathbf{w}_1 = \begin{bmatrix} \sin \Omega_1 \sin i_1 \\ -\cos \Omega_1 \sin i_1 \\ \cos i_1 \end{bmatrix}$$

and \mathbf{w}_2 is the ECI unit angular momentum vector of the final orbit given by

$$\mathbf{w}_2 = \begin{bmatrix} \sin \Omega_2 \sin i_2 \\ -\cos \Omega_2 \sin i_2 \\ \cos i_2 \end{bmatrix}$$

The unit position vector at the first impulse in the reference coordinate system is

$$\mathbf{U}_1 = \begin{bmatrix} \cos \phi_1 \\ \sin \phi_1 \cos i_r \\ \sin \phi_1 \sin i_r \end{bmatrix}$$

and the unit position vector of the second impulse, also in the reference coordinate system, is determined from

$$\mathbf{U}_2 = \begin{bmatrix} \cos \phi_2 \\ \sin \phi_2 \\ 0 \end{bmatrix}$$

The transfer angle can be computed from the following dot product

$$\Delta\theta = \cos^{-1}(\mathbf{U}_1 \bullet \mathbf{U}_2)$$

The minimum and maximum bounds on the semi-parameter of the transfer orbit can be determined from the following two expressions

$$p_{\min} = \frac{r_1 r_2 - \mathbf{r}_1 \bullet \mathbf{r}_2}{r_1 + r_2 + \sqrt{2(r_1 r_2 + \mathbf{r}_1 \bullet \mathbf{r}_2)}} \quad p_{\max} = \frac{r_1 r_2 - \mathbf{r}_1 \bullet \mathbf{r}_2}{r_1 + r_2 - \sqrt{2(r_1 r_2 + \mathbf{r}_1 \bullet \mathbf{r}_2)}}$$

The partial derivative of the total required ΔV with respect to the semi-parameter of the transfer orbit is as follows

$$\frac{\partial V_t}{\partial p_t} = \frac{1}{2p_t} \left(\frac{\Delta \mathbf{V}_1 \bullet (\mathbf{V} - z\mathbf{U}_1)}{|\Delta \mathbf{V}_1|} - \frac{\Delta \mathbf{V}_2 \bullet (\mathbf{V} + z\mathbf{U}_2)}{|\Delta \mathbf{V}_2|} \right)$$

Part of the optimal orbital transfer solution involves finding the value of p_t which lies between p_{\min} and p_{\max} which makes this partial derivative expression equal to zero.

The $\Delta \mathbf{V}$ vectors in the reference coordinate system are given by the following two expressions

$$\Delta \mathbf{V}_1 = \pm (\mathbf{V} + z\mathbf{U}_1) - \mathbf{V}_1$$

$$\Delta \mathbf{V}_2 = \mathbf{V}_2 \mp (\mathbf{V} - z\mathbf{U}_2)$$

where the upper sign in these two equations corresponds to the short transfer and

$$z = \sqrt{\frac{\mu}{p}} \tan \frac{\Delta \theta}{2}$$

with

$$\mathbf{V} = \sqrt{\mu p_t} \frac{(\mathbf{r}_2 - \mathbf{r}_1)}{|\mathbf{r}_1 \times \mathbf{r}_2|}$$

The velocity vector of the satellite prior to the first impulse with respect to the reference coordinate system is calculated from

$$\mathbf{V}_1 = \sqrt{\frac{\mu}{p_1}} \mathbf{W}_1 \times (\mathbf{e}_1 + \mathbf{U}_1)$$

and prior to the second impulse it is given by

$$\mathbf{V}_2 = \sqrt{\frac{\mu}{p_2}} \mathbf{W}_2 \times (\mathbf{e}_2 + \mathbf{U}_2)$$

In these expressions, \mathbf{e}_1 is the reference coordinate system eccentricity vector of the initial orbit which is given by

$$\mathbf{e}_1 = e_1 [\cos \omega_1 \quad \sin \omega_1 \cos i_r \quad \sin \omega_1 \sin i_r]^T$$

and \mathbf{e}_2 is the eccentricity of the final orbit defined by

$$\mathbf{e}_2 = e_2 [\cos \omega_2 \quad \sin \omega_2 \quad 0]^T$$

where e_1 and e_2 are the scalar eccentricity of the initial and final orbits, respectively.

The total scalar delta-v required for the orbit transfer is given by $\Delta V = |\Delta \mathbf{V}_1| + |\Delta \mathbf{V}_2|$.

In terms of the Earth-centered-inertial (ECI) components of the two $\Delta \mathbf{V}$ vectors, the total scalar ΔV required for the orbital transfer is

$$\Delta V = \sqrt{\Delta V_{1x}^2 + \Delta V_{1y}^2 + \Delta V_{1z}^2} + \sqrt{\Delta V_{2x}^2 + \Delta V_{2y}^2 + \Delta V_{2z}^2}$$

This is the scalar quantity we want to minimize.

Eventually, we want to convert the reference coordinate system solution to ECI vectors and then to classical orbital elements. The transformation of an ECI position or velocity vector \mathbf{X}_{eci} to its corresponding reference coordinate system companion \mathbf{X}_{rcs} is given by the following matrix-vector multiplication

$$\mathbf{X}_{eci} = [\mathbf{T}] \mathbf{X}_{rcs}$$

The conversion of a vector in the reference coordinate system to its corresponding ECI vector involves the transpose of this matrix as follows

$$\mathbf{X}_{rcs} = [\mathbf{T}]^T \mathbf{X}_{eci}$$

The elements of the reference coordinate system-to-ECI transformation matrix $[\mathbf{T}]$ are given by the following nine expressions

$$\begin{aligned} T_{11} &= \cos \Omega_2 \cos \phi - \sin \Omega_2 \cos i_2 \sin \phi \\ T_{12} &= -\cos \Omega_2 \sin \phi - \sin \Omega_2 \cos i_2 \cos \phi \\ T_{13} &= \sin \Omega_2 \cos i_2 \\ T_{21} &= \sin \Omega_2 \cos \phi + \cos \Omega_2 \cos i_2 \sin \phi \\ T_{22} &= -\sin \Omega_2 \sin \phi + \cos \Omega_2 \cos i_2 \cos \phi \\ T_{23} &= -\cos \Omega_2 \sin i_2 \\ T_{31} &= \sin i_2 \sin \phi \\ T_{32} &= \sin i_2 \cos \phi \\ T_{33} &= \cos i_2 \end{aligned}$$

where

$$\phi = -\cos^{-1}(\mathbf{N} \bullet \mathbf{U}) \text{sign}(N_z)$$

and

$$\mathbf{U} = [\cos \Omega_2 \quad \sin \Omega_2 \quad 0]^T$$

The position vector of the initial and transfer orbits at the first impulse in the reference coordinate system is

$$\mathbf{r}_1 = \left(\frac{p_1}{1 + e_1 \cos(\phi_1 - \omega_1)} \right) \mathbf{U}_1$$

and the position vector of the transfer and final orbit at the second impulse is

$$\mathbf{r}_2 = \left(\frac{p_2}{1 + e_2 \cos(\phi_2 - \omega_2)} \right) \mathbf{U}_2$$

In these equations the arguments of perigee ω_1 and ω_2 are with respect to the reference coordinate system. They can be determined with the following three equations

$$\omega_{rcs} = [\mathbf{T}]^T \omega_{eci} \quad \omega_1 = \cos^{-1}(\omega_x) \quad \omega_2 = \tan^{-1}(\omega_y, \omega_z)$$

where the inverse tangent calculation here is a four-quadrant operation.

The ECI argument of perigee vectors at each impulse are given by

$$\omega_{eci_1} = \begin{bmatrix} \cos \omega_1 \cos \Omega_1 - \sin \omega_1 \sin \Omega_1 \cos i_1 \\ \cos \omega_1 \sin \Omega_1 + \sin \omega_1 \cos \Omega_1 \cos i_1 \\ \sin \omega_1 \sin i_1 \end{bmatrix}$$

and

$$\omega_{eci_2} = \begin{bmatrix} \cos \omega_2 \cos \Omega_2 - \sin \omega_2 \sin \Omega_2 \cos i_2 \\ \cos \omega_2 \sin \Omega_2 + \sin \omega_2 \cos \Omega_2 \cos i_2 \\ \sin \omega_2 \sin i_2 \end{bmatrix}$$

where all the orbital elements in these two equations are with respect to the ECI coordinate system.

The semi-parameter of the initial orbit can be determined from

$$p_1 = a_1 (1 - e_1^2)$$

and the semi-parameter of the final orbit is given by

$$p_2 = a_2 (1 - e_2^2)$$

where a_1 and a_2 are the semimajor axes of the initial and final orbits, respectively.

The transfer orbit velocity vectors prior to the first and second impulses in the reference coordinate system are calculated from the next two equations

$$\mathbf{V}_{T_1} = \mathbf{V} + z\mathbf{U}_1 \quad \mathbf{V}_{T_2} = \mathbf{V} - z\mathbf{U}_2$$

The transfer orbit position and velocity vectors can be transformed into the ECI coordinate system using the transpose of the $[\mathbf{T}]$ matrix as described above, and then converted to classical orbital elements.

Time-of-flight

The time of flight between perigee and another true anomaly on an elliptic orbit is given by

$$tof(\theta) = \frac{\tau}{2\pi} \left[2 \tan^{-1} \left\{ \sqrt{\frac{1-e}{1+e}} \tan \frac{\theta}{2} \right\} - \frac{e\sqrt{1-e^2} \sin \theta}{1+e \cos \theta} \right]$$

where

τ = orbital period

e = orbital eccentricity

θ = true anomaly

Therefore, the flight time between any two true anomalies on the same elliptical orbit is given by

$$\Delta t = tof(\theta_1) - tof(\theta_2)$$

This equation is implemented in a MATLAB function named `tofl.m` which is used to compute the transfer time between the first and second impulses for the case of a two-impulse orbit transfer.

Primer Vector Analysis

This section summarizes the primer vector analysis used to determine the optimality of solutions computed by this MATLAB script. The term primer vector was invented by Derek F. Lawden and represents the adjoint vector for velocity in the optimal control theory for space trajectories.

A technical discussion about primer theory can be found in Lawden's classic text, *Optimal Trajectories for Space Navigation*, Butterworths, London, 1963. Another excellent resource is "Primer Vector Theory and Applications", Donald J. Jezewski, NASA TR R-454, November 1975, along with "Optimal, Multi-burn, Space Trajectories", also by Jezewski. As noted by Jezewski, the primer vector is sometimes called the Lagrange multiplier, costate vector or perhaps an adjoint variable.

As shown by D. F. Lawden, the following four necessary conditions must be satisfied in order for an impulsive orbital transfer to be *locally optimal*.

- (1) the primer vector and its first derivative are everywhere continuous
- (2) whenever a velocity impulse occurs, the primer is a unit vector aligned with the impulse and has unit magnitude ($\mathbf{p} = \hat{\mathbf{p}} = \hat{\mathbf{u}}_T$ and $\|\mathbf{p}\| = 1$)
- (3) the magnitude of the primer vector may not exceed unity on a coasting arc ($\|\mathbf{p}\| = p \leq 1$)
- (4) at all interior impulses (not at the initial or final times) $\mathbf{p} \cdot \dot{\mathbf{p}} = 0$; therefore, $d\|\mathbf{p}\|/dt = 0$ at the intermediate impulses

Furthermore, the scalar magnitudes of the primer vector derivative at the initial and final impulses provide information about how to improve the nominal transfer trajectory by changing the endpoint times and/or moving the impulse times. These four cases for non-zero slopes are summarized as

- If $\dot{p}_0 > 0$ and $\dot{p}_f < 0 \rightarrow$ perform an initial coast before the first impulse and add a final coast after the second impulse

- If $\dot{p}_0 > 0$ and $\dot{p}_f > 0 \rightarrow$ perform an initial coast before the first impulse and move the second impulse to a later time
- If $\dot{p}_0 < 0$ and $\dot{p}_f < 0 \rightarrow$ perform the first impulse at an earlier time and add a final coast after the second impulse
- If $\dot{p}_0 < 0$ and $\dot{p}_f > 0 \rightarrow$ perform the first impulse at an earlier time and move the second impulse to a later time

The primer vector analysis of a two impulse orbital transfer involves the following computational steps.

First partition the two-body state transition matrix as follows

$$\Phi(t, t_0) = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{r}}{\partial \mathbf{v}_0} \\ \frac{\partial \mathbf{v}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{v}}{\partial \mathbf{v}_0} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix}$$

where

$$\Phi_{11} = \left[\frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} \right] = \begin{bmatrix} \partial x / \partial x_0 & \partial x / \partial y_0 & \partial x / \partial z_0 \\ \partial y / \partial x_0 & \partial y / \partial y_0 & \partial y / \partial z_0 \\ \partial z / \partial x_0 & \partial z / \partial y_0 & \partial z / \partial z_0 \end{bmatrix}$$

and so forth.

The value of the primer vector at any time t along a *two body* trajectory is given by

$$\mathbf{p}(t) = \Phi_{11}(t, t_0) \mathbf{p}_0 + \Phi_{12}(t, t_0) \dot{\mathbf{p}}_0$$

and the value of the primer vector derivative is

$$\dot{\mathbf{p}}(t) = \Phi_{21}(t, t_0) \mathbf{p}_0 + \Phi_{22}(t, t_0) \dot{\mathbf{p}}_0$$

which can also be expressed as

$$\begin{Bmatrix} \mathbf{p} \\ \dot{\mathbf{p}} \end{Bmatrix} = \Phi(t, t_0) \begin{Bmatrix} \mathbf{p}_0 \\ \dot{\mathbf{p}}_0 \end{Bmatrix}$$

In these equations, t_0 represents the time of the first impulse and t_f is the time of the second impulse.

The primer vector boundary conditions at the initial and final impulses are as follows

$$\mathbf{p}(t_0) = \mathbf{p}_0 = \frac{\Delta \mathbf{V}_0}{|\Delta \mathbf{V}_0|} \quad \mathbf{p}(t_f) = \mathbf{p}_f = \frac{\Delta \mathbf{V}_f}{|\Delta \mathbf{V}_f|}$$

These two conditions illustrate that at the locations of velocity impulses, the primer vector is a unit vector in the direction of the corresponding impulse.

The value of the primer vector derivative at the initial time is

$$\dot{\mathbf{p}}(t_0) = \dot{\mathbf{p}}_0 = \Phi_{12}^{-1}(t_f, t_0) \{ \mathbf{p}_f - \Phi_{11}(t_f, t_0) \mathbf{p}_0 \}$$

provided the Φ_{12} state transition sub-matrix is *non-singular*.

The scalar magnitude of the derivative of the primer vector at any mission elapsed time can be determined from

$$\frac{d\|\mathbf{p}\|}{dt} = \frac{d}{dt}(\mathbf{p} \cdot \mathbf{p})^2 = \frac{\dot{\mathbf{p}} \cdot \mathbf{p}}{\|\mathbf{p}\|}$$

Checking a solution for optimality

An `oota_matlab` solution is deemed locally optimal if all the following primer vector and derivative magnitude conditions are true.

$$|\mathbf{p}(t_0)| \leq 1.001 \quad |\mathbf{p}(t_f)| \leq 1.001$$

$$|\mathbf{p}(t)| \leq 1.001 \text{ for all } t_0 < t < t_f$$

$$|\dot{\mathbf{p}}(t_0)| \leq 0.00001 \quad |\dot{\mathbf{p}}(t_f)| \leq 0.00001$$

The first two equations enforce the primer optimality at the first and second impulses. The second equation checks for primer optimality everywhere along the coast portion of the transfer trajectory.

Finally, the last two equations ensure that the primer derivative conditions at the first and second impulse locations are also satisfied.

The `oota_matlab` script creates 300 equally spaced time values along the transfer trajectory. The scalar magnitude of the primer vector and its derivative are computed at these time points using the equations of the previous section. The maximum value of the primer magnitude along the orbit transfer is determined using the MATLAB `max` statement operating on the vector of primer values.

The following is the snippet of MATLAB source code that performs the optimality check.

```
if (y1(1) <= tol_pv && y1(end) <= tol_pv && max(y1) <= tol_pv ...
    && abs(y2(1)) <= tol_pvd && abs(y2(end)) <= tol_pvd)
```

In this statement, `y1` is the array of primer vector magnitudes, `y2` is the array of primer derivative magnitudes, `tol_pv` is the tolerance on the primer vector magnitude (1.001) and `tol_pvd` is the primer derivative magnitude tolerance (0.0001). This statement checks the absolute value of the primer derivative magnitudes since they may be positive or negative at maneuver locations.

These primer and derivative array values are also used to create the graphics image for a two-impulse orbit transfer.

Appendix A

LEO-to-Molniya Example

This appendix describes the OOTA solution of an optimal two impulse orbital transfer from a low circular Earth orbit (LEO) to an elliptical Molniya repeating ground track orbit.

Here is the data definition input (`leo2moly.in`) for this example.

```
*****
* input data file for oota_matlab.m MATLAB script
* impulsive LEO-to-Molniya orbital transfer
* filename ==> leo2moly.in
*****

central body gravitational constant (km^3/sec^2)
398600.4415

central body radius (kilometers)
6378.14

*****
initial orbit
*****

semimajor axis (kilometers)
(semimajor axis > 0)
6653.14

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.0

orbital inclination (degrees)
(0 <= inclination <= 180)
51.6

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
0

right ascension of the ascending node (degrees)
(0 <= raan <= 360)
0

*****
final orbit
*****

semimajor axis (kilometers)
(semimajor axis > 0)
26553.071184

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.737

orbital inclination (degrees)
(0 <= inclination <= 180)
63.4
```

Orbital Mechanics with MATLAB

```

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270.0

right ascension of the ascending node (degrees)
(0 <= raan <= 360)
100.0

*****
algorithm search parameters
*****

initial orbit true anomaly at which to begin the search (degrees)
0

final orbit true anomaly at which to begin the search (degrees)
0

initial orbit true anomaly search increment (degrees)
60

final orbit true anomaly search increment (degrees)
60

number of initial orbit true anomaly search intervals
6

number of final orbit true anomaly search intervals
6

```

The following is one of the solution summaries for this example.

```

input data file ==> leo2moly.in

solution number      1

number of function evaluations  271

number of iterations           139

Optimization terminated:
  the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-08
  and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-08

initial orbit - prior to the first impulse
-----

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+6.653140000000000e+03 +2.77456789218216e-16 +5.160000000000000e+01 +0.000000000000000e+00

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+0.000000000000000e+00 +3.24412296369100e+02 +3.24412296369100e+02 +9.00118574397072e+01

      rx (km)      ry (km)      rz (km)      rmag (km)
+5.41050425386182e+03 -2.40495037709735e+03 -3.03429221767488e+03 +6.653140000000000e+03

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
+4.50443242893421e+00 +3.90986331092379e+00 +4.93301979512252e+00 +7.74026013232195e+00

transfer orbit - after the first impulse
-----

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+2.11539982192466e+04 +6.85491165777775e-01 +5.13247641350847e+01 +3.24478972417759e+02

```

Orbital Mechanics with MATLAB

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.52533693995149e-01	+3.59775987696247e+02	+3.24254960114006e+02	+5.10326858043352e+02
rx (km)	ry (km)	rz (km)	rmag (km)
+5.41050425386183e+03	-2.40495037709736e+03	-3.03429221767489e+03	+6.65314000000000e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
+5.83483985081196e+00	+5.12828173456787e+00	+6.37462388606419e+00	+1.00489033890523e+01

transfer orbit - prior to the second impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.11539982192467e+04	+6.85491165777776e-01	+5.13247641350847e+01	+3.24478972417759e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.52533693995149e-01	+1.52270145267547e+02	+1.16749117685306e+02	+5.10326858043354e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-1.29049013453925e+04	+1.58564428966570e+04	+1.98804833674103e+04	+2.85166070736183e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-2.95123601438420e+00	+3.88768171330320e-01	+5.01937947519657e-01	+3.01875411499481e+00

final orbit - after the second impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.65530711840000e+04	+7.37000000000000e-01	+6.34000000000000e+01	+2.70000000000000e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+1.00000000000000e+02	+1.41231327743477e+02	+5.12313277434767e+01	+7.17681695832038e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-1.29049013453925e+04	+1.58564428966570e+04	+1.98804833674103e+04	+2.85166070736183e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-1.54031917839349e+00	-5.20034004351135e-01	+3.20954662441303e+00	+3.59780600801229e+00

ECI delta-v vectors, magnitudes and LVLH angles

delta-vlx	1330.4074 meters/second
delta-vly	1218.4184 meters/second
delta-vlz	1441.6041 meters/second
delta-vl	2309.2747 meters/second
LVLH pitch angle	-0.3965 degrees
LVLH yaw angle	1.4744 degrees
delta-v2x	1410.9168 meters/second
delta-v2y	-908.8022 meters/second
delta-v2z	2707.6087 meters/second
delta-v2	3185.5537 meters/second
LVLH pitch angle	13.5026 degrees
LVLH yaw angle	230.9747 degrees
total delta-v	5494.8284 meters/second
transfer time	7374.8151 seconds
	122.9136 minutes
	2.0486 hours

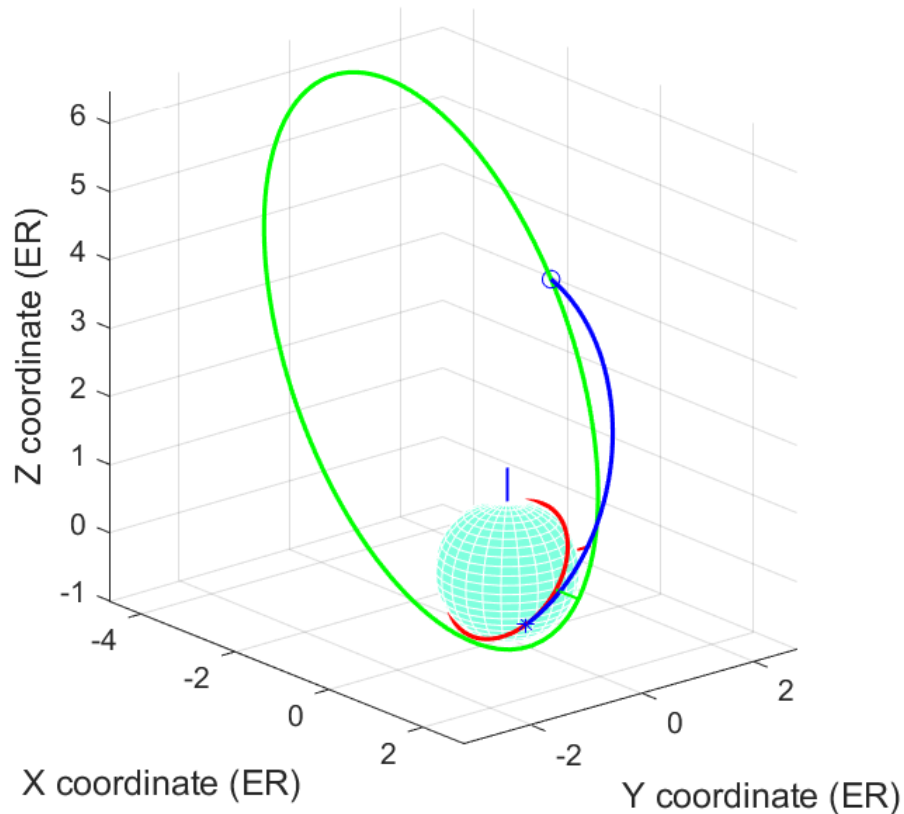
Here is the solution summary for this example.

input data file ==> leo2moly.in

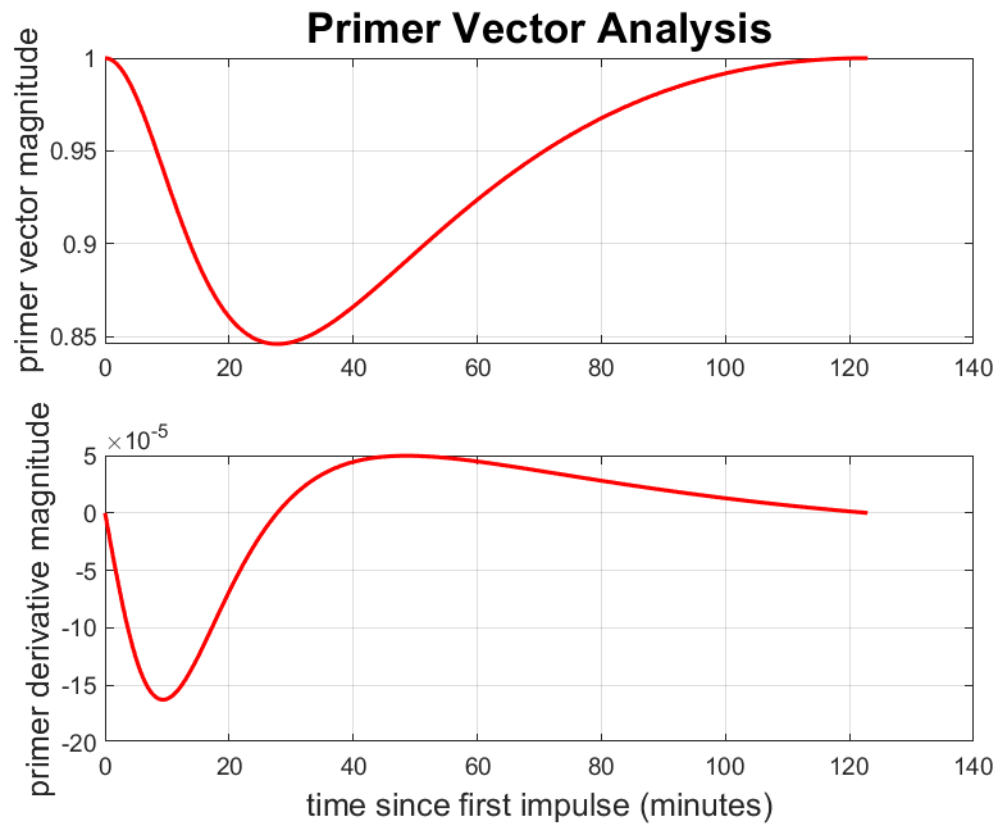
oota solution number	dv1 true anomaly (degrees)	dv2 true anomaly (degrees)	dv1 arg latitude (degrees)	dv2 arg latitude (degrees)	delta-v1 magnitude (m/s)	delta-v2 magnitude (m/s)	total delta-v (m/s)
1	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
2	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
3	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
4	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
5	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
6	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
7	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
8	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
9	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
10	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
11	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
12	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
13	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
14	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
15	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
16	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284
17	324.4123	152.2701	324.4123	116.7491	2309.2747	3185.5537	5494.8284

The following illustrates the trajectory graphics for this example. The initial LEO orbit is red, the elliptical transfer orbit is blue and the final Molniya orbit is green. The asterisk is the orbital location of the first impulse and the small circle is the location of the second impulse.

Initial, Transfer and Final Orbits



Typical behavior of the primer vector and its derivative for this example is illustrated in the next plot.



Gravity Perturbed Optimal Orbital Transfer

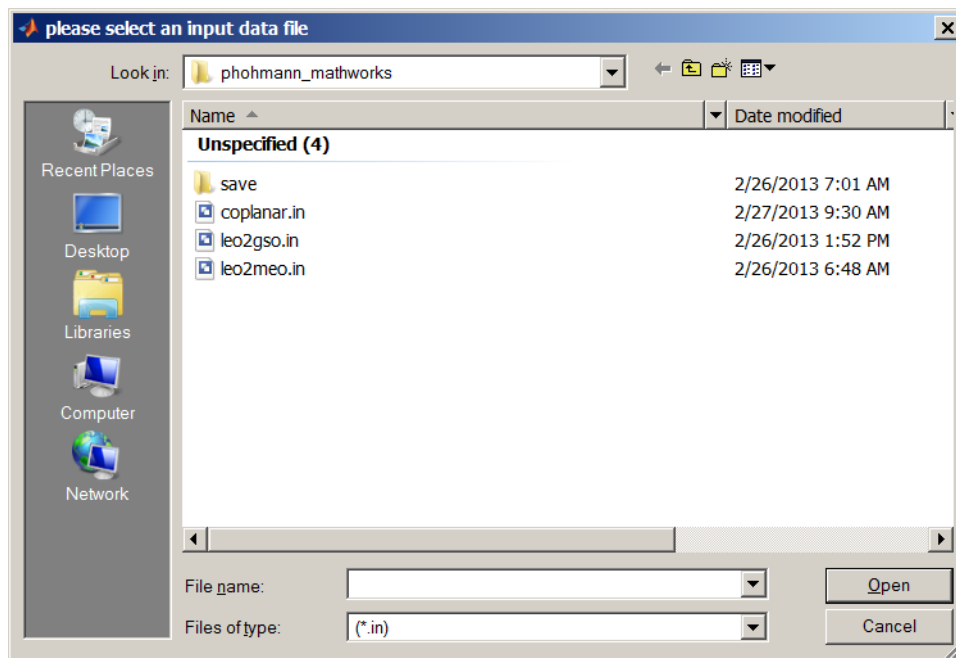
This document is the user's manual for a MATLAB script named `poota` which can be used to solve the optimal two-impulse, non-spherical gravity perturbed orbital transfer between non-coplanar Earth orbits. The algorithm starts with an initial guess computed by the two-body `oota` MATLAB script and then uses the SNOPT nonlinear programming (NLP) method to determine the optimum two impulse orbit transfer subject to non-spherical Earth gravity perturbations. Initial guess conditions can also be extracted from other compatible software simulations. Additional orbit perturbations such as third-body point mass gravity or solar radiation pressure can easily be added to the equations of motion of this MATLAB script.

The `poota` script uses modified equinoctial orbital elements to solve the gravity perturbed orbit transfer "targeting" problem. Additional information about these orbital elements can be found in Appendix A. That appendix also explains how to use components and combinations of these non-singular elements to calculate a variety of final orbital element targets or boundary conditions.

MATLAB versions of SNOPT for several computer platforms can be requested at Professor Philip Gill's web site which is located at <http://ccom.ucsd.edu/~optimizers/>. Professor Gill's web site also includes a PDF version of the SNOPT software user's guide. A brief introduction to nonlinear programming can be found in Appendix B of this document.

User interaction with the script

The `poota` MATLAB script will interactively prompt the user for the name of the simulation definition input data file. This prompt is similar to the following;



The file type defaults to names with a `*.in` filename extension. However, you can select any `poota` compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

Input file format and contents

The `poota` software is “data-driven” by a user-created text file. This text file should be simple ASCII format with no special characters.

The following is a typical input file used by this MATLAB script. In the following discussion the actual input file contents are in courier font and all explanations are in *times italic* font. This example is a transfer from a low Earth orbit (LEO) to a Molniya (from the Russian word for “lightning”) elliptical Earth orbit (EEO). This input file was created using the optimal output from an `oota.m` simulation. In this data file, user provided inputs are in bold 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.

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

```
*****
* input data file for poota.m MATLAB script
* impulsive, gravity-perturbed LEO-to-Molniya orbital transfer
* filename ==> leo2moly.in ==> December 14, 2017
*****
```

The first inputs to the program define the initial UTC calendar date and time for the simulation. The data for the calendar year should include all four digits. The calendar date and time are required to correctly calculate the tesseral or longitude-dependent components of the Earth’s gravity.

```
initial calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
-----
2, 11, 2013

initial UTC
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
-----
20, 18, 19.36
```

The next six inputs are the classical orbital elements of the initial low Earth Orbit (LEO) prior to the first impulsive maneuver. These items can be extracted from an `oota.m` simulation or any other compatible computer program.

```
*****
initial orbit - prior to the first impulse
*****

semimajor axis (kilometers)
(semimajor axis > 0)
6653.14

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.0
```

```
orbital inclination (degrees)
(0 <= inclination <= 180)
51.6

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
0.0

right ascension of the ascending node (degrees)
(0 <= raan <= 360)
0.0

true anomaly (degrees)
(0 <= true anomaly <= 360)
324.412296733363
```

The next three inputs define the Earth-centered-inertial (ECI) delta-v guess for the first impulsive maneuver in the units of meters/second.

```
*****
initial guess for the first ECI delta-v vector
*****

delta-vx (meters/second)
1330.4075

delta-vy (meters/second)
1218.4185

delta-vz (meters/second)
1441.6042
```

The next three inputs summarize the ECI delta-v guess for the second impulsive maneuver also in the units of meters/second.

```
*****
initial guess for the second ECI delta-v vector
*****

delta-vx (meters/second)
1410.9169

delta-vy (meters/second)
-908.8023

delta-vz (meters/second)
2707.6089
```

The next set of six classical orbital elements inputs correspond to the final orbit “targets”.

```
*****
final orbit "targets"
*****

semimajor axis (kilometers)
(semimajor axis > 0)
26553.071184

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.737

orbital inclination (degrees)
(0 <= inclination <= 180)
63.4
```

Orbital Mechanics with MATLAB

```
argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270.0

right ascension of the ascending node (degrees)
(0 <= raan <= 360)
100.0

true anomaly (degrees)
(0 <= true anomaly <= 360)
141.231327743477
```

This next input is the two-body initial guess for the orbit transfer time in seconds.

```
*****
two-body orbit transfer time (seconds)
*****
7374.8151
```

The Earth gravitational constant and radius are user-defined by the next two inputs.

```
*****
astrodynamic constants and gravity model
*****

central body gravitational constant (km^3/sec^2)
-----
398600.4415

central body radius (kilometers)
-----
6378.14
```

Finally, the name of the Earth gravity model to use in the simulation and the order and degree of this model are set by the following three inputs.

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

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

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

The following is the `poota` solution for this example. The first part of the display summarizes useful information about the user-defined initial guess. The second section summarizes the SNOPT iterations and summary. The final section of the output is the gravity perturbed orbit transfer solution found during the optimization.

Gravity-perturbed, Two-impulse Orbit Transfer Analysis

```
user-defined orbital elements and state vector of the park orbit
-----
```

sma (km)	eccentricity	inclination (deg)	argper (deg)
+6.653140000000000e+03	+0.000000000000000e+00	+5.160000000000000e+01	+0.000000000000000e+00
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+0.000000000000000e+00	+3.24412296733363e+02	+3.24412296733363e+02	+9.00118574397073e+01

Orbital Mechanics with MATLAB

rx (km)	ry (km)	rz (km)	rmag (km)
+5.41050427847703e+03	-2.40495035573127e+03	-3.03429219071760e+03	+6.65314000000000e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
+4.50443238891588e+00	+3.90986332871179e+00	+4.93301981756539e+00	+7.74026013232195e+00

initial guess for orbital elements and state vector of the transfer orbit after the first impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.11540016846597e+04	+6.85491217299906e-01	+5.13247641745961e+01	+3.24478972781318e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.52533654168549e-01	+3.59775987721832e+02	+3.24254960503150e+02	+5.10326983444715e+02
rx (km)	ry (km)	rz (km)	rmag (km)
+5.41050427847703e+03	-2.40495035573127e+03	-3.03429219071760e+03	+6.65314000000000e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
+5.83483988891588e+00	+5.12828182871179e+00	+6.37462401756540e+00	+1.00489035426409e+01

initial guess for orbital elements and state vector of the transfer orbit prior to second impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.11540016846598e+04	+6.85491217299906e-01	+5.13247641745960e+01	+3.24478972781318e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.52533654168549e-01	+1.52270140234892e+02	+1.16749113016209e+02	+5.10326983444715e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-1.29049002337334e+04	+1.58564447485915e+04	+1.98804856918629e+04	+2.85166092208052e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-2.95123603224056e+00	+3.88768603587956e-01	+5.01938485784781e-01	+3.01875427761895e+00

initial guess for orbital elements and state vector of the transfer orbit after the second impulse

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.65530827342780e+04	+7.37000036687095e-01	+6.34000067825626e+01	+2.70000015036282e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+1.00000001165420e+02	+1.41231311447734e+02	+5.12313264840160e+01	+7.17682164106908e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-1.29049002337334e+04	+1.58564447485915e+04	+1.98804856918629e+04	+2.85166092208052e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-1.54031913224056e+00	-5.20033696412044e-01	+3.20954738578478e+00	+3.59780662295074e+00

user-defined orbital elements and state vector of the mission orbit

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.65530711840000e+04	+7.37000000000000e-01	+6.34000000000000e+01	+2.70000000000000e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+1.00000000000000e+02	+1.41231327743477e+02	+5.12313277434770e+01	+7.17681695832038e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-1.29049013453926e+04	+1.58564428966570e+04	+1.98804833674105e+04	+2.85166070736184e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-1.54031917839348e+00	-5.20034004351151e-01	+3.20954662441301e+00	+3.59780600801227e+00

Orbital Mechanics with MATLAB

initial guess for initial delta-v vector, magnitude and steering angles

```
-----
x-component of delta-v      1330.407500  meters/second
y-component of delta-v      1218.418500  meters/second
z-component of delta-v      1441.604200  meters/second
```

```
delta-v magnitude          2309.274870  meters/second
```

```
pitch angle                -0.396455  degrees
yaw angle                  -1.474411  degrees
```

initial guess for final delta-v vector, magnitude and steering angles

```
-----
x-component of delta-v      1410.916900  meters/second
y-component of delta-v      -908.802300  meters/second
z-component of delta-v      2707.608900  meters/second
```

```
delta-v magnitude          3185.553967  meters/second
```

```
pitch angle                13.502600  degrees
yaw angle                  129.025342  degrees
```

```
Nonzero derivs  Jij      55
Non-constant    Jij's    55      Constant Jij's          0
```

```
SNJAC  EXIT 100 -- finished successfully
SNJAC  INFO 102 -- Jacobian structure estimated
```

Scale option 0

```
Nonlinear constraints      7      Linear constraints      1
Nonlinear variables        7      Linear variables        0
Jacobian variables         7      Objective variables    6
Total constraints          8      Total variables      7
```

```
Itn      0: Feasible linear rows
Itn      0: PP1. Minimizing Norm(x-x0)

Itn      0: PP1. Norm(x-x0) approximately minimized (0.00E+00)
```

The user has defined 0 out of 55 first derivatives

```
Itn      0: Hessian set to a scaled identity matrix
Itn      4: Infeasible subproblem. Elastic mode started with weight = 2.3E+05
```

```
Major Minors      Step    nCon Feasible    Optimal    MeritFunction      nS Penalty
      0      4          1 2.9E-03  1.0E+00  5.4948288E+00
Search exit 5 -- step too small. Major itn = 0
Itn      4 -- Central differences invoked. Small step length.
```

```
Major Minors      Step    nCon Feasible    Optimal    MeritFunction      nS Penalty
      0      0 1.0E+00      1 2.9E-03  1.0E+00  6.2492568E+00      3.3E-03  r i c
Itn      4: Hessian set to a scaled identity matrix
      1      0 1.0E+00      2 4.6E-06  3.4E-02  5.4964753E+00      3.3E-03  r i
Search exit 5 -- step too small. Major itn = 1
Itn      4 -- Central differences invoked. Small step length.
      1      0 1.0E+00      2 4.6E-06  3.4E-02  5.4976636E+00      2.0E+00  r i c
Itn      4: Elastic weight increased to 2.350E+06
Itn      4: Elastic weight increased to 2.350E+08
Itn      4: Elastic weight increased to 2.350E+10
      2      0 1.0E+00      3 (3.8E-11) (3.9E-11) 5.4976627E+00      2.0E+00  r i c
```

```
SNOPTA EXIT 0 -- finished successfully
SNOPTA INFO 1 -- optimality conditions satisfied
```

```
Problem name          matlabMx
No. of iterations      4      Objective          5.4976631940E+00
```

Orbital Mechanics with MATLAB

No. of major iterations	2	Linear obj. term	0.0000000000E+00
Penalty parameter	2.026E+00	Nonlinear obj. term	5.4976631940E+00
User function calls (total)	45	Calls with modes 1,2 (known g)	3
Calls for forward differencing	21	Calls for central differencing	14
No. of degenerate steps	0	Percentage	0.00
Max x	7 7.4E+03	Max pi	4 2.3E+10
Max Primal infeas	12 2.0E-12	Max Dual infeas	3 9.3E+01
Nonlinear constraint violn	2.8E-07		

Solution printed on file 9

Time for MPS input	0.00 seconds
Time for solving problem	1.37 seconds
Time for solution output	0.00 seconds
Time for constraint functions	1.40 seconds
Time for objective function	0.00 seconds

Gravity-perturbed Two-impulse Orbit Transfer Analysis

 predicted orbital elements and state vector of the transfer orbit after the initial delta-v

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.11671023071891e+04	+6.85685412823596e-01	+5.13366592810150e+01	+3.24424997756782e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.41569275048433e-01	+3.59836813556265e+02	+3.24261811313046e+02	+5.10801123250296e+02
rx (km)	ry (km)	rz (km)	rmag (km)
+5.41050427847703e+03	-2.40495035573127e+03	-3.03429219071760e+03	+6.65314000000000e+03
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
+5.83871587822862e+00	+5.12500634248986e+00	+6.37462401756540e+00	+1.00494837918453e+01

 predicted orbital elements and state vector of the transfer orbit prior to the final delta-v

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.11418028453867e+04	+6.85332159815510e-01	+5.13298884384500e+01	+3.24489090032656e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+2.02185017854783e-01	+1.52289343782529e+02	+1.16778433815185e+02	+5.09885613105625e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-1.29007200838770e+04	+1.58578532716098e+04	+1.98717713621579e+04	+2.85094262681018e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-2.95089496123111e+00	+3.89427513695840e-01	+4.99613504467329e-01	+3.01812002983457e+00

 predicted orbital elements and state vector of the mission orbit

sma (km)	eccentricity	inclination (deg)	argper (deg)
+2.65530711858840e+04	+7.37000000029211e-01	+6.34000000005345e+01	+2.7000000000315e+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
+9.9999999989945e+01	+1.41218027730079e+02	+5.12180277303945e+01	+7.17681695908418e+02
rx (km)	ry (km)	rz (km)	rmag (km)
-1.29007200838770e+04	+1.58578532716098e+04	+1.98717713621579e+04	+2.85094262681018e+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-1.54092132791403e+00	-5.19293981387735e-01	+3.21047420530623e+00	+3.59878443378630e+00

Orbital Mechanics with MATLAB

predicted initial delta-v vector, magnitude and steering angles

```
-----  
x-component of delta-v      1334.283489  meters/second  
y-component of delta-v      1215.143014  meters/second  
z-component of delta-v      1441.604200  meters/second  
  
delta-v magnitude          2309.785194  meters/second  
  
pitch angle                 27.789418   degrees  
yaw angle                   100.512247   degrees
```

predicted final delta-v vector, magnitude and steering angles

```
-----  
x-component of delta-v      1409.973633  meters/second  
y-component of delta-v      -908.721495  meters/second  
z-component of delta-v      2710.860701  meters/second  
  
delta-v magnitude          3187.878000  meters/second  
  
pitch angle                 13.534348   degrees  
yaw angle                   129.042135   degrees  
  
total delta-v              5497.663194  meters/second  
  
transfer time               7374.815100  seconds  
                           122.913585  minutes  
                           2.048560   hours  
  
degree of gravity model     8  
order of gravity model      8
```

The following is a brief summary of the information provided by this MATLAB script.

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 position vector in kilometers

ry (km) = y-component of the position vector in kilometers

rz (km) = z-component of the position vector in kilometers

rmag (km) = scalar magnitude of the position vector in kilometers

vx (km/sec) = x-component of the velocity vector in kilometers per second

vy (km/sec) = y-component of the velocity vector in kilometers per second

vz (km/sec) = z-component of the velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the velocity vector in kilometers per second

x-component of delta-v = ECI x-component of the impulsive delta-v maneuver in meters per second

y-component of delta-v = ECI y-component of the impulsive delta-v maneuver in meters per second

z-component of delta-v = ECI z-component of the impulsive delta-v maneuver in meters per second

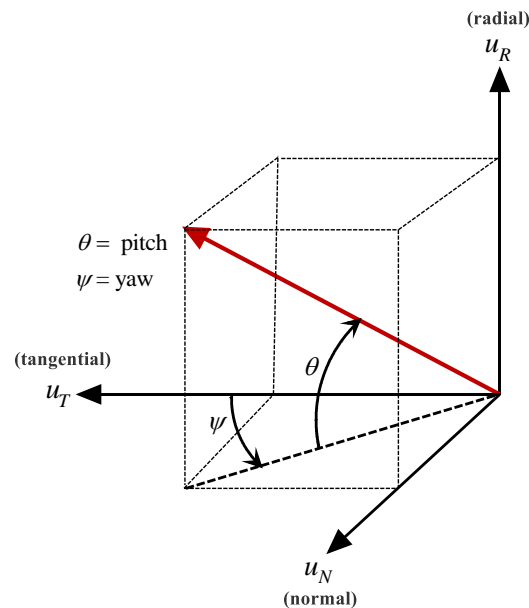
delta-v magnitude = scalar magnitude of the impulsive delta-v maneuver in meters per second

transfer time = time interval between the two impulsive maneuvers in seconds, minutes and hours

Note that ECI implies an Earth-centered-inertial coordinate system.

Additional information about the data displayed by the optimization algorithm can be found in the SNOPT user's manual which is available at Professor Gill's website which is located at <http://scicomp.ucsd.edu/~peg/>.

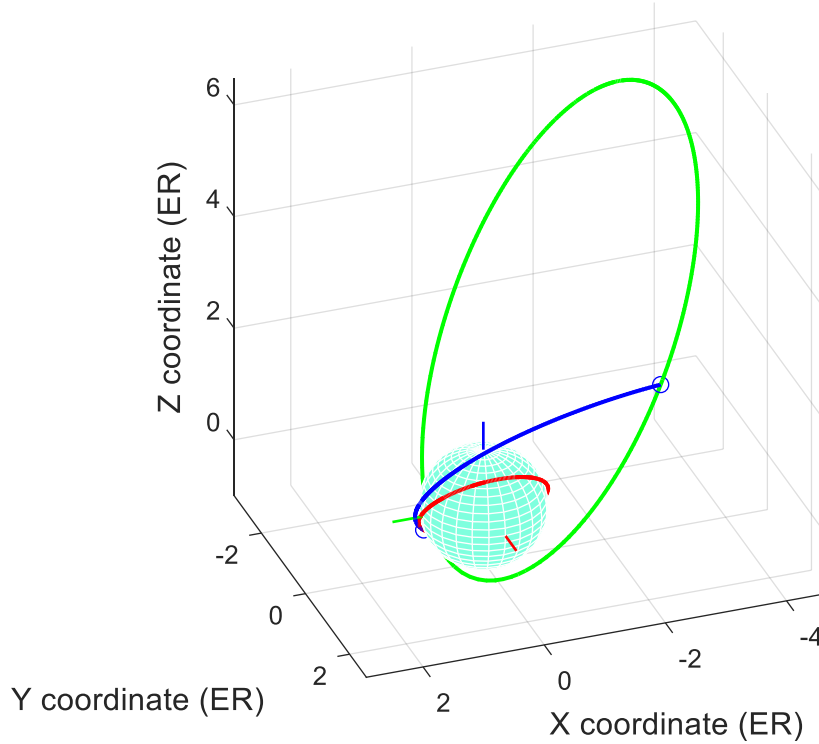
The pitch and yaw angles for each impulsive maneuver are computed and displayed in a local-vertical-local horizontal coordinate system. The following diagram illustrates the geometry of the pitch and yaw angles in this system. In this figure, the radial direction is along the geocentric radius vector directed away from the Earth, the tangential direction is tangent to the orbit in the direction of the orbital motion, and the normal direction is along the angular momentum vector of the orbit. The pitch angle is positive above the local horizontal plane formed by the tangential and normal directions, and the yaw angle is positive in the direction of the angular momentum vector which is perpendicular to the orbit plane.



The `poota` script will also create a graphics display of the initial, transfer and final orbits. The following is the graphics display for this example. The initial orbit trace is red, the transfer orbit is blue and the final mission orbit is green. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis. The location of each impulse is marked with a small blue circle.

Trajectory image files are saved to disk in both encapsulated, color Postscript format and MATLAB `fig` format. The disk file names are `pootal.eps` and `pootal.fig`. The interactive features of MATLAB graphics allow the user to re-load and manipulate the `fig` version of the trajectory display. These capabilities also allow the user to interactively find the best viewpoint as well as verify basic three-dimensional geometry of the orbital maneuver.

Orbit Transfer: Initial, Transfer and Final Orbits



Technical discussion

In this MATLAB script, the orbital motion is modeled with respect to a *true-of-date* Earth-centered-inertial (ECI) coordinate system. The origin of this system is the center of the Earth and the fundamental plane is the Earth's equator. The x -axis is aligned with the true-of-date Vernal Equinox, the z -axis is aligned with the Earth's spin axis, and the y -axis completes this orthogonal, right-handed coordinate system.

Acceleration due to Earth gravity

This MATLAB script uses a *spherical harmonic* representation of the Earth's geopotential function given by

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

where ϕ is the geocentric latitude, λ is the geocentric east longitude and $r = |\mathbf{r}| = \sqrt{x^2 + y^2 + z^2}$ is the geocentric distance. In this expression the S 's and C 's are *unnormalized* harmonic coefficients of the

geopotential, and the P 's are associated Legendre polynomials of degree n and order m with argument $u = \sin \phi$.

The software calculates the 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 Φ , the inertial rectangular cartesian components of the spacecraft's acceleration vector are as follows

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

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

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

where

R = radius of the Earth

r = geocentric distance

S_n^m, C_n^m = harmonic coefficients

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

λ = longitude = $\alpha - \alpha_g$

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

α_g = right ascension of Greenwich

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

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

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

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

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

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

where the first few associated Legendre functions are given by

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

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

The trigonometric arguments are determined from expansions given by

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

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

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

The following are the first 14 lines of the 18 by 18 `egm96.dat` gravity model file included with this script. Column 1 is the degree l , column 2 is the order m , column 3 is the C coefficients and the last column contains the S gravity model coefficients.

2	0	-1.08262668355E-003	0.00000000000E+000
3	0	2.53265648533E-006	0.00000000000E+000
4	0	1.61962159137E-006	0.00000000000E+000
5	0	2.27296082869E-007	0.00000000000E+000
6	0	-5.40681239107E-007	0.00000000000E+000
7	0	3.52359908418E-007	0.00000000000E+000
8	0	2.04799466985E-007	0.00000000000E+000
9	0	1.20616967365E-007	0.00000000000E+000
10	0	2.41145438626E-007	0.00000000000E+000
11	0	-2.44402148325E-007	0.00000000000E+000
12	0	1.88626318279E-007	0.00000000000E+000
13	0	2.19788001661E-007	0.00000000000E+000
14	0	-1.30744533118E-007	0.00000000000E+000

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 δ_{m0} is equal to 1 if m is zero and equal to zero if m is greater than zero.

The following is the MATLAB source code for the function that opens and reads a gravity model file (fname) and creates matrices of the un-normalized coefficients.

```
function [cccoef, scoef] = readgm(fname)

% read gravity model data file

% input

% fname = name of gravity data file

% output

% ccoef, scoef = gravity model coefficients

% data file format (space delimited ascii)

% column 1 is the degree, column 2 is the order, column 3 are the C coefficients
% and the last column contains the S gravity model coefficients. For example,

% 2      0      -1.08262668355E-003      0.00000000000E+000
% 3      0       2.53265648533E-006      0.00000000000E+000
% 4      0       1.61962159137E-006      0.00000000000E+000

% Orbital Mechanics with MATLAB

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

% read the data file

gdata = dlmread(fname);

nrows = size(gdata, 1);

% initialize coefficients

idim = gdata(nrows, 1) + 1;

cccoef = zeros(idim, idim);

scoef = zeros(idim, idim);

% create gravity model coefficients

for n = 1:nrows

    i = gdata(n, 1);

    j = gdata(n, 2);

    ccoef(i + 1, j + 1) = gdata(n, 3);

    scoef(i + 1, j + 1) = gdata(n, 4);

end
```

Solving the trajectory optimization problem

As mentioned earlier, the trajectory optimization uses a two-body orbital transfer solution generated with the `oota.m` MATLAB script for the initial guess. For this problem, the components of the initial and final impulsive delta-v vector and the orbit transfer time are control variables. The objective or cost function for this problem is the sum of the scalar magnitude of the two impulses given by

$$f = \|\Delta \mathbf{v}_i\| + \|\Delta \mathbf{v}_f\|$$

This is the scalar value we want to minimize.

The SNOPT algorithm requires initial guesses (`xg`) for the six components of the delta-v vectors as well as lower (`xlwr`) and upper (`xupr`) bounds on each component. It also requires lower (`flow`) and upper (`fupp`) bounds on the objective function and any linear or nonlinear constraints.

The following is the MATLAB source code that sets up this information.

```
% initial guess for components of initial delta-v
xg(1) = dv1(1);
xg(2) = dv1(2);
xg(3) = dv1(3);

% initial guess for components of final delta-v
xg(4) = dv2(1);
xg(5) = dv2(2);
xg(6) = dv2(3);

% initial guess for orbit transfer time
xg(7) = ttransfer;

xg = xg';

% define lower and upper bounds for components of delta-v vectors (kilometers/second)
for i = 1:1:3
    xlwr(i) = xg(i) - 0.01;
    xupr(i) = xg(i) + 0.01;
end
for i = 4:1:6
    xlwr(i) = xg(i) - 0.01;
    xupr(i) = xg(i) + 0.01;
end

% lower and upper bounds on transfer time (seconds)
xlwr(7) = 0.95 * ttransfer;
xupr(7) = 1.05 * ttransfer;
```

```
% bounds on objective function

flow(1) = 0.0e0;
fupp(1) = +Inf;

% semimajor axis ==> enforce final modified equinoctial "p" equality constraint

flow(2) = 0.0;
fupp(2) = 0.0;

% orbital eccentricity ==> enforce final modified equinoctial "sqrt(f^ + g^2)" equality
constraint

flow(3) = 0.0;
fupp(3) = 0.0;

% orbital inclination ==> enforce final modified equinoctial "sqrt(h^2 + k^2)" equality
constraint

flow(4) = 0.0;
fupp(4) = 0.0;

% argument of perigee constraints

flow(5) = sin(oev_mo(4));
fupp(5) = sin(oev_mo(4));
flow(6) = cos(oev_mo(4));
fupp(6) = cos(oev_mo(4));

% raan constraints

flow(7) = sin(oev_mo(5));
fupp(7) = sin(oev_mo(5));
flow(8) = cos(oev_mo(5));
fupp(8) = cos(oev_mo(5));

flow = flow';
fupp = fupp';

% solve the orbital TPBVP using SNOPT

snscreen on;

xmul = zeros(7, 1);
xstate = zeros(7, 1);
fmul = zeros(8, 1);
fstate = zeros(8, 1);

% solve the orbital TPBVP using SNOPT

snscreen on;

[x, f, inform, xmul, fmul] = snopt(xg, xlwr, xupr, flow, fupp, 'tpbvp');
```

The `tpbvp` MATLAB function defines the current value of the objective function and the mission constraints. The function that evaluates the Earth gravity model and the first order equations of motion is called `ceqm1`. Here's the source code for this function.

```
function ydot = ceqm1 (t, y)

% first order form of Cowell's equations of orbital motion

% version for ode45

% input

% t = current simulation time
% y = current eci state vector

% output

% ydot = eci acceleration vector

% Orbital Mechanics with MATLAB

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

% compute non-spherical gravity perturbations

agrav = gravity(t, y);

% total acceleration vector

ydot = [ y(4)
         y(5)
         y(6)
         agrav(1)
         agrav(2)
         agrav(3)];
```

The `tpbvp` objective function starts with the two-body solution for the position and velocity vectors of the maneuver on the initial orbit and the current value for the delta-v vector and numerically integrates the first-order form of the orbital equations for the predicted transfer time. At the final time the software adds the current value of the second delta-v vector to the velocity vector of the transfer orbit to determine the velocity vector on the final mission orbit. This predicted state vector is used to compute the current modified equinoctial orbital elements of the final mission orbit.

The following is the MATLAB source code for the `tpbvp` function.

```
function [f, g] = tpbvp(x)

% two point boundary value objective function and
% and modified equinoctial element constraints

% input

% x(1:3) = current first delta-v vector (kilometers/second)
% x(4:6) = current second delta-v vector (kilometers/second)
% x(7)   = current transfer time (seconds)

% output

% f(1) = objective function (total delta-v magnitude)
% f(2) = predicted semiparameter difference
% f(3) = predicted orbital eccentricity difference
% f(4) = predicted orbital inclination difference
```

Orbital Mechanics with MATLAB

```
% f(5) = predicted mission orbit sine of argument of perigee
% f(6) = predicted mission orbit cosine of argument of perigee
% f(7) = predicted mission orbit sine of raan
% f(8) = predicted mission orbit cosine of raan

% Orbital Mechanics with MATLAB

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

global emu rpo vpo ritop vitop rftop vftop

global mee_target mee_predict

% load current state vector of transfer orbit after first impulse

ritop(1) = rpo(1);
ritop(2) = rpo(2);
ritop(3) = rpo(3);

vitop(1) = vpo(1) + x(1);
vitop(2) = vpo(2) + x(2);
vitop(3) = vpo(3) + x(3);

% load current orbit transfer time (seconds)

ttime = x(7);

% set up options for ode45

options = odeset('RelTol', 1.0e-8, 'AbsTol', 1.0e-8);

% propagate to end of current transfer orbit time

[~, ysol] = ode45(@ceqm1, [0.0 ttime], [ritop, vitop], options);

% current "predicted" state vector prior to second impulse

rftop(1:3) = ysol(end, 1:3);

vftop(1:3) = ysol(end, 4:6);

% current "predicted" mission orbit position vector

rmop = rftop;

% current "predicted" mission orbit velocity vector after second delta-v

vmop(1) = vftop(1) + x(4);
vmop(2) = vftop(2) + x(5);
vmop(3) = vftop(3) + x(6);

% compute current modified equinoctial and classical orbital elements
% of "predicted" mission orbit

mee_predict = eci2mee(emu, rmop, vmop);
oev_predict = eci2orbl(emu, rmop, vmop);

% objective function (total delta-v magnitude)

f(1) = norm(x(1:3)) + norm(x(4:6));

% mission orbit semimajor axis difference (kilometers)

f(2) = mee_predict(1) - mee_target(1);
```



```
% mission orbit orbital eccentricity difference

f(3) = sqrt(mee_predict(2)^2 + mee_predict(3)^2) ...
      - sqrt(mee_target(2)^2 + mee_target(3)^2);

% mission orbit orbital inclination difference

f(4) = sqrt(mee_predict(4)^2 + mee_predict(5)^2) ...
      - sqrt(mee_target(4)^2 + mee_target(5)^2);

% mission orbit sine and cosine of argument of perigee

f(5) = (mee_predict(3) * mee_predict(4) - mee_predict(2) * mee_predict(5)) ...
      / (oev_predict(2) * tan(0.5 * oev_predict(3)));

f(6) = (mee_predict(2) * mee_predict(4) + mee_predict(3) * mee_predict(5)) ...
      / (oev_predict(2) * tan(0.5 * oev_predict(3)));

% mission orbit sine and cosine of raan

f(7) = mee_predict(5) / (tan(0.5 * oev_predict(3)));

f(8) = mee_predict(4) / (tan(0.5 * oev_predict(3)));

% transpose objective function/constraints vector

f = f';

% no derivatives

g = [];
```

Targeting to the final mission orbit

This section summarizes the technique used to compute and enforce the nonlinear constraints that define the final mission or “targeted” orbit. The semimajor axis constraint is simply

$$p_p - p_t = 0$$

The orbital eccentricity constraint is formulated as follows

$$\sqrt{f_p^2 + g_p^2} = \sqrt{f_t^2 + g_t^2}$$

The orbital inclination constraint is coded according to

$$\sqrt{h_p^2 + k_p^2} = \sqrt{h_t^2 + k_t^2}$$

The argument of perigee constraints are formulated as

$$\sin \omega_t = \frac{g_p h_p - f_p k_p}{e_p \tan(i_p/2)} \quad \text{and} \quad \cos \omega_t = \frac{f_p h_p + g_p k_p}{e_p \tan(i_p/2)}$$

The right ascension of the ascending node constraints are implemented by

$$\sin \Omega_t = \frac{k_p}{\tan(i_p/2)} \quad \text{and} \quad \cos \Omega_t = \frac{h_p}{\tan(i_p/2)}$$

In these equations, the t subscript implies “target” or desired values and the p subscript implies values “predicted” by the optimization process. The targeted values are computed using the position and velocity vectors of the mission orbit determined during the transfer orbit solution. Note that true anomaly of the final or mission orbit is not enforced explicitly since it results from the numerical integration for the predicted orbital transfer time.

Additional information about targeting with the modified equinoctial orbital elements can be found in Appendix B.

Algorithm resources

- (1) Walter Hohmann, *Die Erreichbarkeit der Himmelskörper*, Oldenbourg, Munich, 1925. Also, *The Attainability of Heavenly Bodies*, NASA Technical Translation F-44, 1960.
- (2) Jean-Pierre Marec, *Optimal Space Trajectories*, Elsevier, 1979.
- (3) R. P. Brent, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, 1972.
- (4) R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA, 1987.
- (5) D. F. Lawden, *Optimal Trajectories for Space Navigation*, Butterworths, London, 1963.
- (6) John E. Prussing, “Simple Proof of the Global Optimality of the Hohmann Transfer”, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 15, No. 4.
- (7) A. Miele, M. Ciarcia, and J. Mathwig, “Reflections on the Hohmann Transfer”, *Journal of Optimization Theory and Applications*, Vol. 123, No. 2, pp. 233-253, November 2004.

Appendix A

Targeting with Modified Equinoctial Orbital Elements

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 *direct* modified equinoctial equations exhibit no singularity for zero eccentricity and orbital inclinations equal to 0 and 90 degrees. However, please note that two of the components are singular for an orbital inclination of 180 degrees.

The classic reference for these elements is “A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

The modified equinoctial elements are defined in terms of the classical orbital elements as follows:

$$p = a(1 - e^2)$$

$$f = e \cos(\omega + \Omega)$$

$$g = e \sin(\omega + \Omega)$$

$$h = \tan(i/2) \cos \Omega$$

$$k = \tan(i/2) \sin \Omega$$

$$L = \theta + \omega + \Omega$$

where

p = semiparameter

a = semimajor axis

e = orbital eccentricity

i = orbital inclination

ω = argument of perigee

Ω = right ascension of the ascending node

θ = true anomaly

L = true longitude

The classical orbital elements can be recovered from the modified equinoctial orbital elements with

semimajor axis

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

orbital eccentricity

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

orbital inclination

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

argument of periapsis

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

$$\sin \omega = \frac{g h - f k}{e \tan(i/2)}$$

$$\cos \omega = \frac{f h + g k}{e \tan(i/2)}$$

right ascension of the ascending node

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

$$\sin \Omega = \frac{k}{\tan(i/2)}$$

$$\cos \Omega = \frac{h}{\tan(i/2)}$$

true anomaly

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

$$\sin \theta = \frac{1}{e} (f \sin L - g \cos L)$$

$$\cos \theta = \frac{1}{e} (f \cos L + g \sin L)$$

In these expressions, an inverse tangent expression of the form $\theta = \tan^{-1}(a, b)$ denotes a four quadrant evaluation where $a = \sin \theta$ and $b = \cos \theta$.

Constraint formulations that enforce both the sine and cosine of a desired orbital element should be used whenever possible. This approach involves a combination of equality and inequality constraints and ensures that the “targeted” orbital element is in the correct quadrant.

To illustrate this technique, here are several examples for different values of argument of perigee and the corresponding mission constraints:

$$0^\circ < \omega < 90^\circ \rightarrow \begin{cases} \sin \omega > 0 \rightarrow gh - fk > 0 \\ fh + gk = e \tan(i/2) \cos \omega \end{cases}$$

$$\omega = 270^\circ \rightarrow \begin{cases} \sin \omega \leq 0 \rightarrow gh - fk \leq 0 \\ \cos \omega = 0 \rightarrow fh + gk = 0 \end{cases}$$

$$\omega = 178^\circ \rightarrow \begin{cases} gh - fk = e \tan(i/2) \sin \omega \\ \cos \omega \leq 0 \rightarrow fh + gk \leq 0 \end{cases}$$

The following is a *sign* table of the sine and cosine for each quadrant.

quadrant	sine	cosine
1	+	+
2	+	−
3	−	−
4	−	+

orbital eccentricity constraint

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

For a circular orbit, $f = g = 0$.

orbital inclination constraint

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

For an equatorial orbit, $h = k = 0$.

argument of perigee constraints

$$gh - fk = e \sin \omega \tan(i/2) \rightarrow \sin \omega = \frac{gh - fk}{e \tan(i/2)}$$

$$fh + gk = e \cos \omega \tan(i/2) \rightarrow \cos \omega = \frac{fh + gk}{e \tan(i/2)}$$

right ascension of the ascending node constraints

$$k = \tan(i/2) \sin \Omega \rightarrow \sin \Omega = \frac{k}{\tan(i/2)}$$

$$h = \tan(i/2) \cos \Omega \rightarrow \cos \Omega = \frac{h}{\tan(i/2)}$$

true anomaly constraints

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

In general,

$$\sin \theta = \frac{1}{e} (f \sin L - g \cos L)$$

$$\cos \theta = \frac{1}{e} (f \cos L + g \sin L)$$

For a circular orbit,

$$\sin \theta = \sin L \cos \Omega - \cos L \sin \Omega$$

$$\cos \theta = \cos L \cos \Omega + \sin L \sin \Omega$$

For a circular, equatorial orbit,

$$\theta = L, \sin \theta = \sin L \text{ and } \cos \theta = \cos L.$$

Also, note that

$$r_p = \frac{p}{1 + \sqrt{f^2 + g^2}} \rightarrow \text{periapsis radius}$$

$$r_a = \frac{p}{1 - \sqrt{f^2 + g^2}} \rightarrow \text{apoapsis radius}$$

Algorithm resources

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, August 20-22, 1990.

Appendix B

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 font.

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 $\mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), t] = 0$.

For any mission time t there are also simple bounds on the state variables

$$\mathbf{y}_l \leq \mathbf{y}(t) \leq \mathbf{y}_u$$

the control variables

$$\mathbf{u}_l \leq \mathbf{u}(t) \leq \mathbf{u}_u$$

and the problem parameters

$$\mathbf{p}_l \leq \mathbf{p}(t) \leq \mathbf{p}_u$$

The basic nonlinear programming problem (NLP) involves the determination of 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.

Algorithm resources

- (1) “Direct Trajectory Optimization Using Nonlinear Programming and Collocation”, C. R. Hargraves and S. W. Paris, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 10, No. 4, July-August, 1987, pp. 338-342.
- (2) “Optimal Finite-Thrust Spacecraft Trajectories Using Direct Transcription and Nonlinear Programming”, Paul J. Enright, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1991.
- (3) “Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.
- (4) “Improved Collocation Methods with Application to Direct Trajectory Optimization”, Albert L. Herman, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1995.
- (5) “Survey of Numerical Methods for Trajectory Optimization”, John T. Betts, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998, pp. 193-207.
- (6) *Practical Optimization*, Philip E. Gill, Walter Murray and Margaret H. Wright, Emerald Group Publishing Limited, 1982.

Program rendezvous_ocs

Finite-Burn, Earth Orbit Rendezvous Trajectory Optimization

This document is the user's manual for a Fortran computer program called `rendezvous_ocs` that uses the *Sparse Optimization Suite* distributed by [Applied Mathematical Analysis](#) to solve the classic Earth orbit rendezvous trajectory optimization problem. The software models the trajectory as a mission consisting of one or more maneuvers separated by coasting periods. The propulsive phases are simulated as variable thrust, finite-burn propulsive maneuvers. This computer program attempts to maximize the spacecraft mass at the end of the propulsive maneuvers.

The important features of this scientific simulation are as follows:

- finite-burn orbital maneuvers
- variable attitude steering during all maneuvers
- user-defined throttle bounds
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified flyby or rendezvous final orbit constraints
- fixed or free final flyby or rendezvous time

The *Sparse Optimization 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 *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010 (www.siam.org).

The `rendezvous_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 *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `rendezvous_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:

```
rendezvous_ocs leo2meo_10k.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 rendezvous_ocs      *
*                                  *
*      finite burn earth orbit     *
*      rendezvous optimization     *
*                                  *
*      April 10, 2012              *
*****

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.

The screen output created by the `rendezvous_ocs` computer program can be re-directed to a text file with a command line similar to

```
rendezvous_ocs leo2meo_10k.in >leo2meo_10k.txt
```

To create a DOS command window in Windows 7, select **start**, then **All Programs**, then **Accessories** and finally **Command Prompt**. The size, font and other characteristics of the screen can be controlled by the user with the **c:**\ icon in the upper left corner of the window. To log into the subdirectory created during the installation of the Fortran executable and support files, type **root:**\ and then **cd subdirectory** from the DOS command line where **root** is the name of the root directory, usually **c:**, and **subdirectory** is the name of the subdirectory created by the user.

The DOS command line prompt looks similar to **C:\rendezvous_ocs>_**.

Input file format and contents

The `rendezvous_ocs` software is “data-driven” by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in *times italic* font. This example attempts to optimize the maneuvers required to perform a rendezvous between a spacecraft in a circular low Earth orbit (LEO) and a second spacecraft in a typical medium altitude Earth orbit (MEO).

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.

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 earth-orbit rendezvous
** trajectory optimization
** program rendezvous_ocs
** leo2meo_10k.in - April 10, 2012
*****
```

The first input is an integer that tells the simulation what type of trajectory to model.

```
trajectory type (1 = flyby, 2 = rendezvous)
2
```

The next three inputs define an initial guess, lower bound and upper bound for the total simulation duration in minutes. Identical values for the lower and upper bounds will create a fixed time mission.

```
initial guess for total simulation duration (minutes)
85.0

lower bound for total simulation duration (minutes)
88.0

upper bound for total simulation duration (minutes)
88.0
```

The next input is the initial mass of the entire spacecraft in kilograms.

```
initial spacecraft mass (kilograms)
8000.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

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

The next four inputs define the thrust magnitude and the specific impulse of the upper stage or spacecraft propulsion system, and the user's initial guess for either the delta-v or thrust duration for the first maneuver.

```
-----
first propulsive maneuver
-----
thrust magnitude (newtons)
10000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
2925.0

initial guess for thrust duration (seconds)
170.0
```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
8000.0

orbital eccentricity (non-dimensional)
0.015

orbital inclination (degrees)
28.5

argument of perigee (degrees)
100.0

right ascension of the ascending node (degrees)
20.0

true anomaly (degrees)
30.0
```

The next six inputs define the classical orbital elements of the final mission orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
*****
* FINAL ORBIT *
*****

semimajor axis (kilometers)
10000.0

orbital eccentricity (non-dimensional)
0.05

orbital inclination (degrees)
40.0

argument of perigee (degrees)
200.0

right ascension of the ascending node (degrees)
55.0

true anomaly (degrees)
120.0
```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```
*****
* type of gravity model *
-----
1 = spherical Earth
2 = oblate gravity model
-----
2
```

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 options *
*****
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 binary initial guess data file
leo2meo_10k.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 data file (yes or no)
no
```

This next input specifies the type of solution data file to create.

```
*****
* 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 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
25
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
leo2heo_10k.csv
```

The next series of program inputs are algorithm control options and parameters for the Sparse Optimization Suite. 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 input defines the relative error in the objective function.

```

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.

```

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

```

10000

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

```

maximum number of algorithm iterations

```

10000

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

```

*****
sparse NLP iteration output
-----

```

```

1 = none
2 = terse
3 = standard
4 = interpretive
5 = diagnostic
-----

```

2

The level of output from the 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 Sparse Optimization Suite 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 Sparse Optimization Suite 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 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 rendezvous_ocs software. Please consult Appendix C.

```

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

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

name of optimal control configuration file
leo2meo_10k_config.txt

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

name of optimal control configuration file
leo2meo_10k_config1.txt

```

Optimal control solution

The following is the optimal control solution for this example. The output includes the time and orbital characteristics at the beginning and end of each mission phase.

```

program rendezvous_ocs
=====

input file ==> leo2meo_10k.in

rendezvous trajectory

oblate earth gravity model

-----
beginning of maneuver phase
-----

mission elapsed time      00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.8000000000000D+04  0.1500000000000D-01  0.2850000000000D+02  0.1000000000000D+03

```

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.200000000000D+02	0.300000000000D+02	0.130000000000D+03	0.118684693004D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.658713032027D+04	0.325905620287D+04	0.288604970418D+04	0.789563272225D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.381028378016D+01	-.564780744839D+01	-.217399960475D+01	0.715138208606D+01

end of maneuver phase			

mission elapsed time		00:54:37.540	
sma (km)	eccentricity	inclination (deg)	argper (deg)
0.999778599805D+04	0.500760702724D-01	0.399921786418D+02	0.199752487246D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.550628877938D+02	0.467623343061D+02	0.246514821552D+03	0.165811819507D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.335362513193D+04	-.702990130399D+04	-.568340909816D+04	0.964196312202D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.515138258024D+01	0.360501455968D+01	-.181069217135D+01	0.654304811253D+01
spacecraft mass	3339.96227073119	kilograms	
propellant mass	4660.03772926881	kilograms	
phase duration	3277.53953053337	seconds	
	54.6256588422229	minutes	
delta-v	2998.07572342380	meters/second	

The following program output is the spacecraft mass, the propellant mass consumed, the actual thrust duration for all the maneuvers, and the accumulated delta-v for the mission.

spacecraft mass	3339.96227073119	kilograms
propellant mass	4660.03772926881	kilograms
phase duration	3277.53953053337	seconds
	54.6256588422229	minutes
delta-v	2998.07572342380	meters/second

This section of the numeric results summarizes the time and orbital conditions at the beginning and end of the transfer orbit coast.

beginning of coast phase			

mission elapsed time		00:54:37.540	
sma (km)	eccentricity	inclination (deg)	argper (deg)
0.999778599805D+04	0.500760702724D-01	0.399921786418D+02	0.199752487246D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)

0.550628877938D+02	0.467623343061D+02	0.246514821552D+03	0.165811819507D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.335362513193D+04	-.702990130399D+04	-.568340909816D+04	0.964196312202D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.515138258024D+01	0.360501455968D+01	-.181069217135D+01	0.654304811253D+01

end of coast phase			

mission elapsed time		01:27:60.000	
sma (km)	eccentricity	inclination (deg)	argper (deg)
0.100000000000D+05	0.500000000000D-01	0.400000000000D+02	0.200000000000D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.550000000000D+02	0.120000000000D+03	0.320000000000D+03	0.165866900904D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.862186499607D+04	0.353038894193D+04	-.422710739886D+04	0.102307692308D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.459681283666D+00	0.541422590670D+01	0.292176275871D+01	0.616942839083D+01
coast duration	2002.46046946663	seconds	
	33.3743411577771	minutes	

After the simulation is complete, the software will display a simulation summary similar to the following;

```

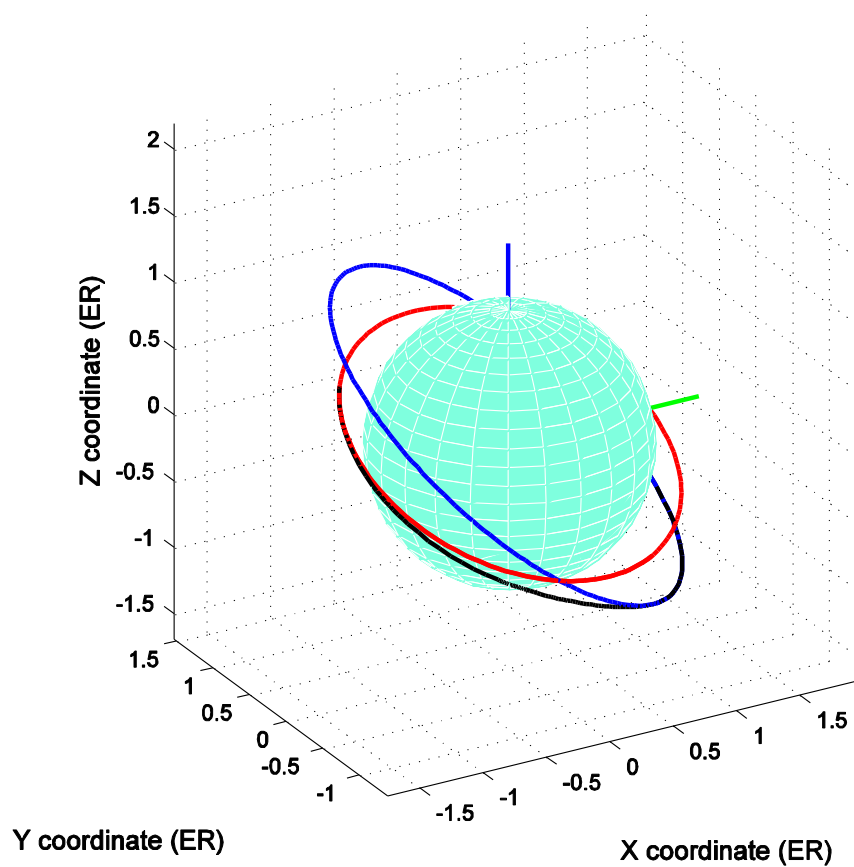
SIMULATION SUMMARY
=====
initial mass          8000.00000000000    kilograms
total propellant mass 4660.03772926881     kilograms
final spacecraft mass 3339.96227073119     kilograms
total delta-v         2998.07572342380     meters/second
total sim duration    5280.00000000000     seconds
                    88.0000000000000     minutes

```

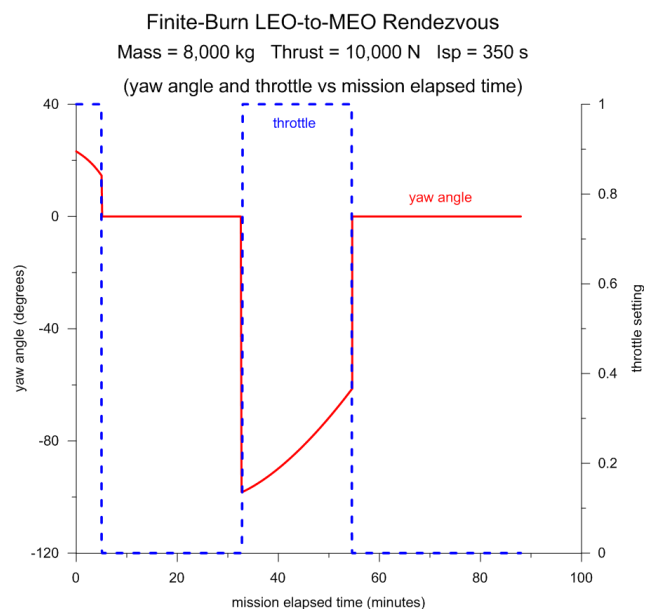
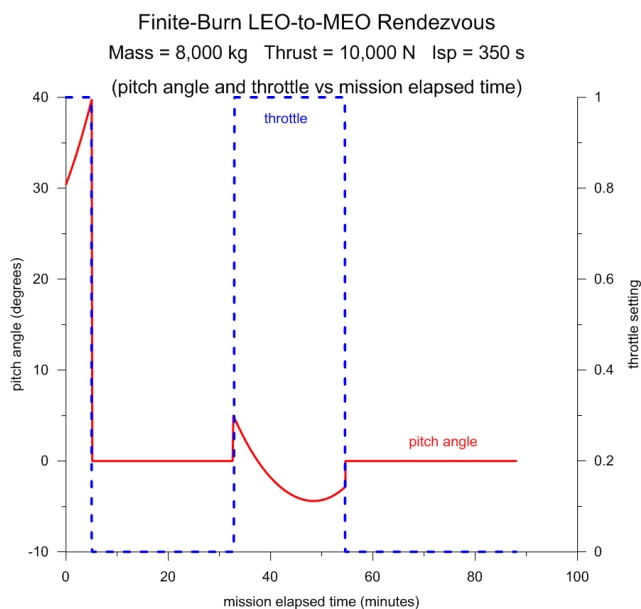
The rendezvous_ocs computer program will also create an output file called orbits.csv. This file contains the Earth-centered inertial position and velocity vectors of the park orbit and final mission orbit. The rendezvous_ocs software package includes a MATLAB script called oplot.m that can be used to create trajectory graphic displays using this data file. 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 orbital transfer.

The following is the graphics display for this example. The initial orbit trace is red, the final orbit is blue and the transfer orbit is black. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis.

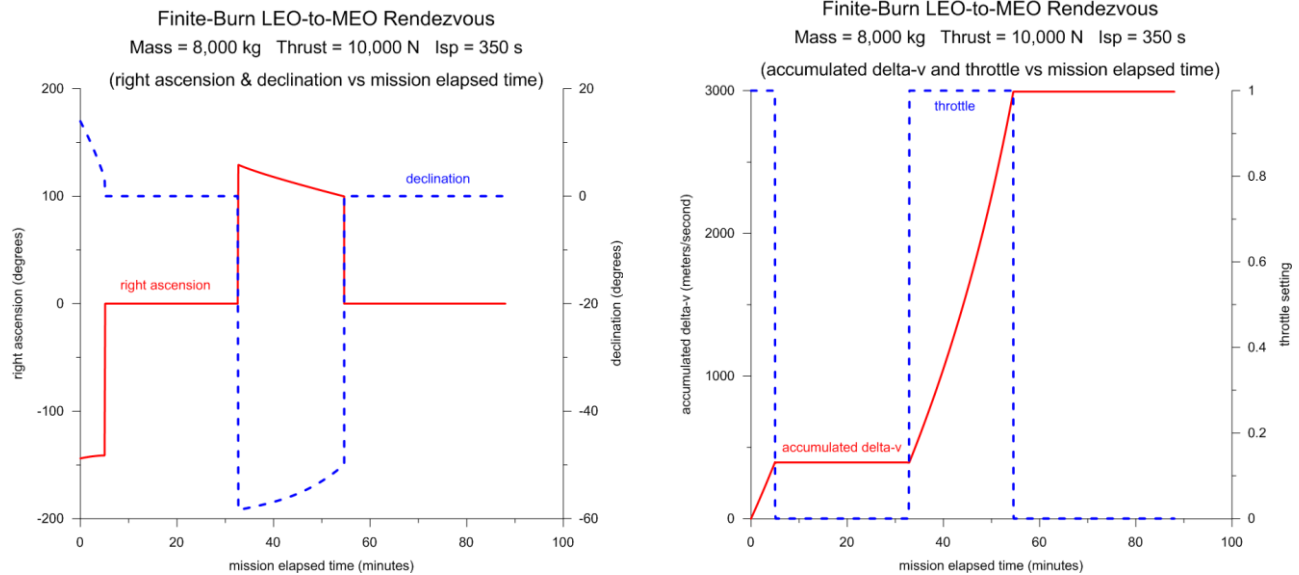
Initial, Transfer and Final Orbits



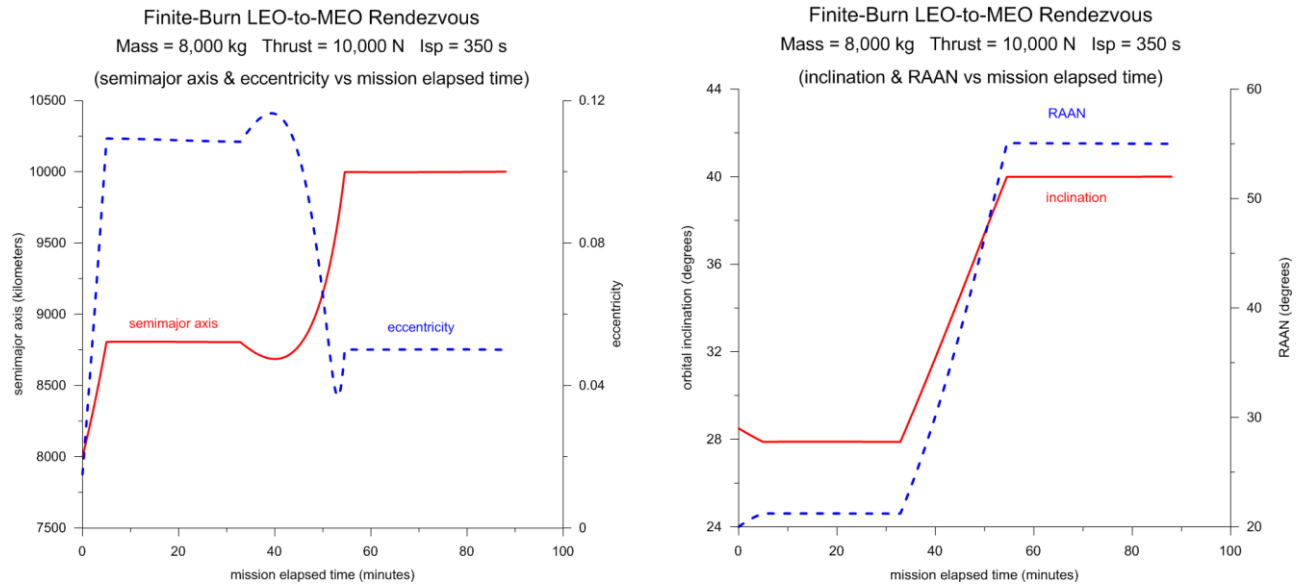
The following two plots illustrate the evolution of the pitch and yaw steering angles and the throttle setting during the two propulsive maneuvers determined by the software.



The next two plots illustrate the behavior of the inertial right ascension and declination, and the accumulated delta-v and throttle setting during the simulation.



The next pair of plots illustrate the behavior of the semimajor axis, eccentricity, inclination and the right ascension of the ascending node (RAAN) during the simulation.



Verification of the optimal control solution

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

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

verification of optimal control solution
=====

final mass	3339.96216840498	kilograms
propellant mass	4660.03783159502	kilograms
delta-v	2998.07650071517	meters/second

final mission orbit

mission elapsed time 01:27:60.000

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.100000037673D+05	0.500000524679D-01	0.400000018124D+02	0.199999582004D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.549999947879D+02	0.120000292543D+03	0.319999874547D+03	0.165866994633D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.862187310660D+04	0.353037110635D+04	-.422712123100D+04	0.102307756263D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.459669452953D+00	0.541422750064D+01	0.292175590011D+01	0.616942566002D+01

Creating an initial guess

The software allows the user to input either a delta-v or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. An estimate of the thrust duration can be determined from the following expression:

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

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

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

In these equations

m_i = initial mass

m_f = final mass

m_p = propellant mass

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

I_{sp} = specific impulse

ΔV = impulsive velocity increment

F = thrust

g = acceleration of gravity

The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting, the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = [0 \ 1 \ 0]^T$.

The dynamic variables and control variables at each grid point are determined by the *Sparse Optimization Suite* by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2`. These program options create an initial guess from the numerical integration of the equations programmed in the `oderhs` subroutine. The number and location of the initial collocation nodes are determined from the variable step-size numerical integration. The `INIT(2) = 2` option tells the program to use a Dormand-Prince numerical integration method.

Please note that this algorithm creates a *coplanar* initial guess.

If the software cannot find a feasible solution, try increasing the guess for the thrust duration or the value for the magnitude of the delta-v.

Binary restart data files can also be used to initialize a `rendezvous_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 may be necessary to make successive small changes in the mission definition and run multiple simulations to eventually reach the final desired solution.

Problem setup

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

(1) Point functions – initial orbit constraints

For this two-point boundary value problem (TPBVP), both lower and upper bounds for all modified equinoctial elements are set equal to the user-defined initial modified equinoctial orbital elements as follows:

$$\begin{aligned} p_L &= p_U = p_i & f_L &= f_U = f_i \\ g_L &= g_U = g_i & h_L &= h_U = h_i \\ k_L &= k_U = k_i & L_L &= L_U = L_i \end{aligned}$$

In *Sparse Optimization Suite* 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 the end of the mission. This is simply

$$J = m_f$$

The value of the `maxmin` indicator tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

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

For variable attitude steering, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows:

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

(4) *Point functions – final mission orbit constraints*

The final mission constraints enforced by the software are determined by the trajectory type. For the flyby trajectory option, the final orbit position vector is constrained to the values corresponding to the user-defined final orbit. The rendezvous trajectory option adds the final three components of the inertial velocity vector to this constraint set.

The computation of inertial position and velocity vectors from the modified equinoctial orbital elements is described in the Technical Discussion section later in this document.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05m_{sc_i} \leq m_{sc} \leq 1.05m_{sc_i} \quad 100p_f \leq p \leq 0.8p_i$$

$$-1 \leq f \leq +1 \quad -1 \leq g \leq +1$$

$$-1 \leq h \leq +1 \quad -1 \leq k \leq +1$$

where m_{sc_i} is the initial spacecraft mass.

For variable attitude steering, the three components of the unit thrust vector are constrained as follows:

$$-1.1 \leq u_r \leq +1.1$$

$$-1.1 \leq u_t \leq +1.1$$

$$-1.1 \leq u_n \leq +1.1$$

Technical Discussion

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

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

The relationship between classical and modified equinoctial orbital elements is summarized as follows:

semimajor axis

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

orbital eccentricity

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

orbital inclination

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

argument of periapsis

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

right ascension of the ascending node

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

true anomaly

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

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

position vector

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

velocity vector

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

where

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

and

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

The total *non-two-body* acceleration vector is given by $\mathbf{P} = \Delta_r \hat{\mathbf{i}}_r + \Delta_t \hat{\mathbf{i}}_t + \Delta_n \hat{\mathbf{i}}_n$.

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

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

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:

$$\begin{aligned}\Delta_{J_{2r}} &= -\frac{3\mu J_2 R_e^2}{2r^4} \left[1 - \frac{12(h \sin L - k \cos L)^2}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2t}} &= -\frac{12\mu J_2 R_e^2}{r^4} \left[\frac{(h \sin L - k \cos L)(h \cos L + k \sin L)}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2n}} &= -\frac{6\mu J_2 R_e^2}{r^4} \left[\frac{(1 - h^2 - k^2)(h \sin L - k \cos L)}{(1 + h^2 + k^2)^2} \right]\end{aligned}$$

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

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

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = [u_{T_r} \quad u_{T_t} \quad u_{T_n}]^T$ 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.*

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 and \dot{m} is the propellant flow rate.

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

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

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

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the “local horizontal” and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four quadrant operation.

The `rendezvous_ocs` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. The following section summarizes the inertial-to/from-modified equinoctial coordinate transformations and the calculation of the inertial unit thrust vector in terms of right ascension and declination angles.

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

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

where

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

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 `rendezvous_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

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

where the calculation for right ascension is a four quadrant inverse tangent operation.

Algorithm Resources

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“Optimal Interplanetary Orbit Transfers by Direct Transcription”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

“Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, 20-22 August 1990.

“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.

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 the CSV data files produced by the `rendezvous_ocs` software.

The simulation summary screen display contains the following information:

mission elapsed time = simulation time since beginning of the mission (hh:mm:ss.sss)

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 spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

spacecraft mass = current spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

phase duration = current phase duration in seconds and minutes

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The accumulated delta-v is determined using a cubic spline integration of the thrust acceleration data at each collocation node.

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

time (sec) = simulation time since ignition in seconds
semimajor axis (km) = semimajor axis in kilometers
eccentricity = orbital eccentricity (non-dimensional)
inclination (deg) = orbital inclination in degrees
argument 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
period (min) = orbital period in minutes
mass = spacecraft mass in kilograms
thracc = thrust acceleration in meters/second**2
yaw = thrust vector yaw angle in degrees
pitch = thrust vector pitch angle in degrees
perigee altitude = perigee altitude in kilometers
apogee altitude = apogee altitude in kilometers
ut-radial = radial component of unit thrust vector
ut-tangential = tangential component of unit thrust vector
ut-normal = normal component of unit thrust vector
semi-parameter = orbital semiparameter in kilometers
f equinoctial element = modified equinoctial orbital element
g equinoctial element = modified equinoctial orbital element
h equinoctial element = modified equinoctial orbital element
k equinoctial element = modified equinoctial orbital element
true longitude = true longitude in degrees
rx (km) = x-component of the spacecraft's position vector in kilometers
ry (km) = y-component of the spacecraft's position vector in kilometers
rz (km) = z-component of the spacecraft's position vector in kilometers
fpa (deg) = flight path angle in degrees
deltav (mps) = accumulative delta-v in meters per second

The `orbits.csv` file contains the following information:

time (seconds) = simulation time since ignition in seconds
rp1-x (er) = x-component of the initial orbit position vector in earth radii
rp1-y (er) = y-component of the initial orbit position vector in earth radii
rp1-z (er) = z-component of the initial orbit position vector in earth radii
rp2-x (er) = x-component of the final orbit position vector in earth radii
rp2-y (er) = y-component of the final orbit position vector in earth radii

rp2-z (er) = z-component of the final orbit position vector in earth radii

APPENDIX B

Example Flyby Trajectory Analysis

This appendix summarizes a flyby mission from LEO to MEO. This example starts and ends at the same orbits as the previous example. However, the propulsive thrust is 5000 Newtons and the total simulation time is fixed to 100 minutes. Furthermore, since this is a flyby trajectory, the orbit transfer only matches the three components of the position vector of the final mission orbit at the final time.

Here's the initial part of the simulation definition input file for this example.

```
*****
** finite-burn earth-orbit rendezvous
** trajectory optimization
** program rendezvous_ocs
** leo2meo_flyby.in - April 10, 2012
*****

trajectory type (1 = flyby, 2 = rendezvous)
1

initial guess for total simulation duration (minutes)
95.0

lower bound for total simulation duration (minutes)
100.0

upper bound for total simulation duration (minutes)
100.0

initial spacecraft mass (kilograms)
8000.0

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

-----
propulsive maneuver
-----

thrust magnitude (newtons)
5000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
2925.0

initial guess for thrust duration (seconds)
170.0

lower bound for throttle setting
0.0

upper bound for throttle setting
1.0
```

Here's the program output for this example.

```

program rendezvous_ocs
=====

input file ==> leo2meo_flyby.in

flyby trajectory

oblate earth gravity model

-----
beginning of maneuver phase
-----

mission elapsed time      00:00:00.000


      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.800000000000D+04    0.150000000000D-01    0.285000000000D+02    0.100000000000D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.200000000000D+02    0.300000000000D+02    0.130000000000D+03    0.118684693004D+03

      rx (km)      ry (km)      rz (km)      rmag (km)
-0.658713032027D+04    0.325905620287D+04    0.288604970418D+04    0.789563272225D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-0.381028378016D+01    -0.564780744839D+01    -0.217399960475D+01    0.715138208606D+01

-----
end of maneuver phase
-----

mission elapsed time      01:10:11.736


      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.898711286539D+04    0.235466089640D+00    0.350916114708D+02    0.903365417492D+02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.625248414158D+02    0.172979511791D+03    0.263316053540D+03    0.141315573076D+03

      rx (km)      ry (km)      rz (km)      rmag (km)
0.739228594994D+04    -0.529738113942D+04    -0.632512076131D+04    0.110776934037D+05

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.298144849526D+01    0.430206674637D+01    -0.463960881703D+00    0.525471912855D+01


spacecraft mass      3343.59238906710      kilograms
propellant mass      4656.40761093290      kilograms
phase duration      4211.73581241822      seconds
                   70.1955968736369      minutes
delta-v      2994.34791421085      meters/second

-----
beginning of coast phase
-----

mission elapsed time      01:10:11.736

```

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.898711286539D+04	0.235466089640D+00	0.350916114708D+02	0.903365417492D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.625248414158D+02	0.172979511791D+03	0.263316053540D+03	0.141315573076D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.739228594994D+04	-.529738113942D+04	-.632512076131D+04	0.110776934037D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.298144849526D+01	0.430206674637D+01	-.463960881703D+00	0.525471912855D+01

end of coast phase

mission elapsed time 01:40:00.000

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.898867629134D+04	0.235360368321D+00	0.351008512173D+02	0.903369639672D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.624745498836D+02	0.223728548707D+03	0.314065512674D+03	0.141352450221D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.862186499607D+04	0.353038894193D+04	-.422710739886D+04	0.102307692308D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.192043174636D+01	0.473402281402D+01	0.273461673498D+01	0.579458012140D+01

coast duration	1788.26418758178	seconds
	29.8044031263631	minutes

SIMULATION SUMMARY

=====

initial mass	8000.000000000000	kilograms
total propellant mass	4656.40761093290	kilograms
final spacecraft mass	3343.59238906710	kilograms
total delta-v	2994.34791421085	meters/second
total sim duration	6000.000000000000	seconds
	100.000000000000	minutes

verification of optimal control solution

=====

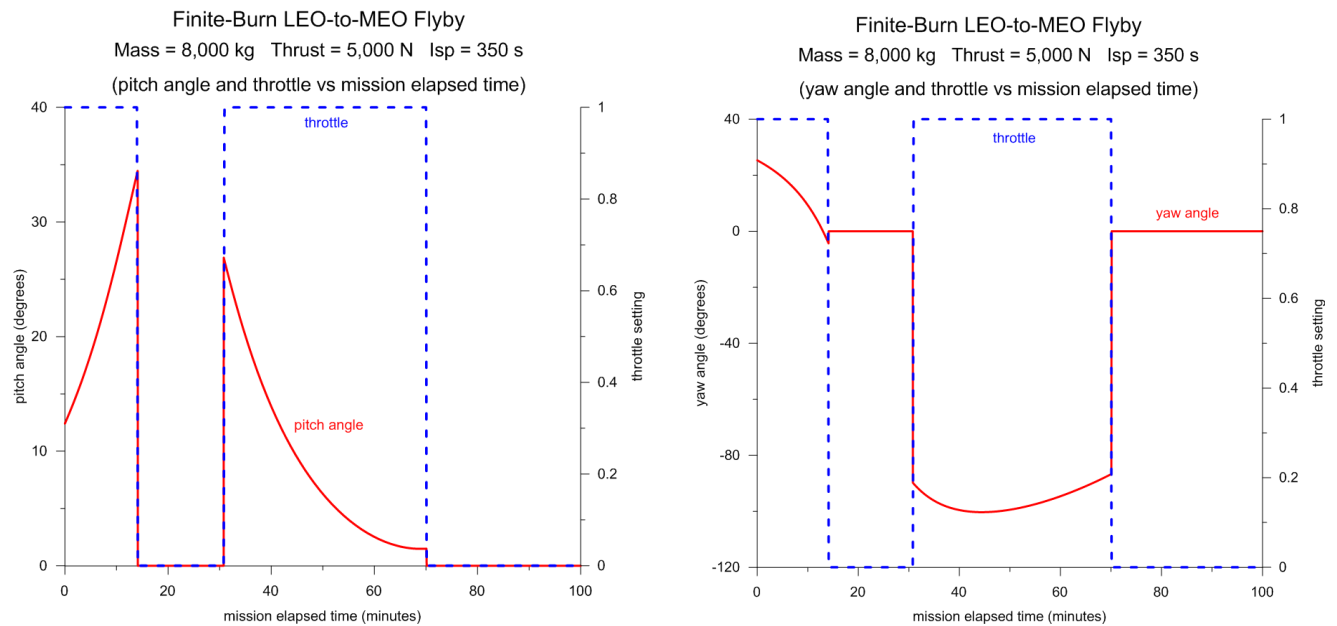
final mass	3343.59248030948	kilograms
propellant mass	4656.40751969052	kilograms
delta-v	2994.34782001675	meters/second

final mission orbit

mission elapsed time 01:40:00.000

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.898867644662D+04	0.235360359825D+00	0.351008505110D+02	0.903369671277D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.624745489581D+02	0.223728544739D+03	0.314065511867D+03	0.141352453884D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.862186536286D+04	0.353038874660D+04	-.422710749938D+04	0.102307695140D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.192043150637D+01	0.473402286160D+01	0.273461656678D+01	0.579458000135D+01

Here are plots of the behavior of the control variables and throttle setting for this example.



APPENDIX C

Typical Sparse Optimization Suite Configuration File

The `rendezvous_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 *Sparse Optimization Suite* user's manual. Please note that the `rendezvous_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 `rendezvous_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
TOLMSA=0.1490116119384766D-07
TOLMR=0.1490116119384766D-07
IDSCPH=0
IDSCND=0
IDSCVR=0
IDSCFN=0
IDTSFD=-1
IPFAUX=0
IPFSFD=0
IPRSFD=1
IPGRD=0
IPNLP=10
IODE=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.rsb
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:RHO1WR=0.100000000000000D+03
INSNLP:IMAXMU=10
INSNLP:MUCALC=3
INSNLP:MXQPIT=1

Low-Thrust Orbital Transfer with Solar-Electric Propulsion

This document describes an interactive MATLAB script (`sep_ltot.m`) which can be used to determine the characteristics of continuous, low-thrust orbital transfer between two *nonplanar* circular orbits using solar-electric propulsion (SEP). The numerical method used in this script is described in Chapter 14 of the book *Orbital Mechanics* by V. Chobotov and the technical paper “The Reformulation of Edelbaum's Low-thrust Transfer Problem Using Optimal Control Theory” by Jean. A. Kechichian, AIAA-92-4576-CP. The original Edelbaum algorithm is described in “Propulsion Requirements for Controllable Satellites”, ARS Journal, August 1961, pp. 1079-1089.

This algorithm is valid for total inclination changes Δi given by $0 < \Delta i < 114.6^\circ$. This algorithm assumes that the thrust acceleration magnitude and spacecraft mass are both constant during the orbit transfer. Earth shadow effects on the orbital transfer are ignored in this MATLAB script.

The propulsive thrust provided by an SEP system is given by

$$T = \frac{2\eta P}{gI_{sp}}$$

where η is the non-dimensional propulsive efficiency, P is the input power in kilowatts, g is the acceleration of gravity in meters/second and I_{sp} is the specific impulse in seconds. The quantity gI_{sp} is also called the exhaust velocity. Note that with these metric units the thrust will be in milli-newtons. The thrust acceleration required in the equations to follow is equal to $a_T = T/m$ where m is the mass of the spacecraft.

The initial thrust vector yaw angle β_0 is given by the following expression

$$\tan \beta_0 = \frac{\sin\left(\frac{\pi}{2}\Delta i\right)}{\frac{V_0}{V_f} - \cos\left(\frac{\pi}{2}\Delta i\right)}$$

where the speed on the initial circular orbit is $V_0 = \sqrt{\mu/r_0}$ and the speed on the final circular orbit is $V_f = \sqrt{\mu/r_f}$. In these equations $r_0 = r_e + h_0$ is the geocentric radius of the initial orbit, $r_f = r_e + h_f$ is the geocentric radius of the final orbit, r_e is the radius of the Earth and μ is the gravitational constant of the Earth. The initial circular orbit altitude is h_0 , the final circular orbit altitude is h_f , and Δi is the total orbital inclination change.

The total velocity change required for a low-thrust orbit transfer is given by

$$\Delta V = V_0 \cos \beta_0 - \frac{V_0 \sin \beta_0}{\tan\left(\frac{\pi}{2} \Delta i + \beta_0\right)}$$

The total transfer time is given by $t = \Delta V / a_T$ where a_T is the thrust acceleration. The time evolution of the out-of-plane yaw angle, speed and inclination change are given by the following three expressions:

$$\beta(t) = \tan^{-1}\left(\frac{V_0 \sin \beta_0}{V_0 \cos \beta_0 - a_T t}\right)$$

$$V(t) = \sqrt{V_0^2 - 2V_0 a_T t \cos \beta_0 + a_T^2 t^2}$$

$$\Delta i(t) = \frac{2}{\pi} \left[\tan^{-1}\left(\frac{a_T t - V_0 \cos \beta_0}{V_0 \sin \beta_0}\right) + \frac{\pi}{2} - \beta_0 \right]$$

Finally, the propellant mass m_p required for the maneuver can be determined from the ideal rocket equation as follows:

$$m_p = m_i \left(1 - e^{-\Delta V / g I_{sp}}\right)$$

where m_i is the initial spacecraft mass.

This MATLAB script will prompt you for the initial and final altitudes and orbital inclinations, and the SEP propulsive characteristics. The following is a typical user interaction with this script. It illustrates an orbital transfer from a low Earth orbit (LEO) with an inclination of 28.5° to a geosynchronous Earth orbit (GSO) with an orbital inclination of 0° .

SEP Low-thrust Orbit Transfer Analysis

```

please input the initial altitude (kilometers)
? 621.86

please input the final altitude (kilometers)
? 35787.86

please input the initial orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the final orbital inclination (degrees)
(0 <= inclination <= 180)
? 0
    
```


Orbital Mechanics with MATLAB

please input the initial spacecraft mass (kilograms)
? 1147.732571

please input the SEP propulsive efficiency (non-dimensional)
? .65

please input the SEP input power (kilowatts)
? 10

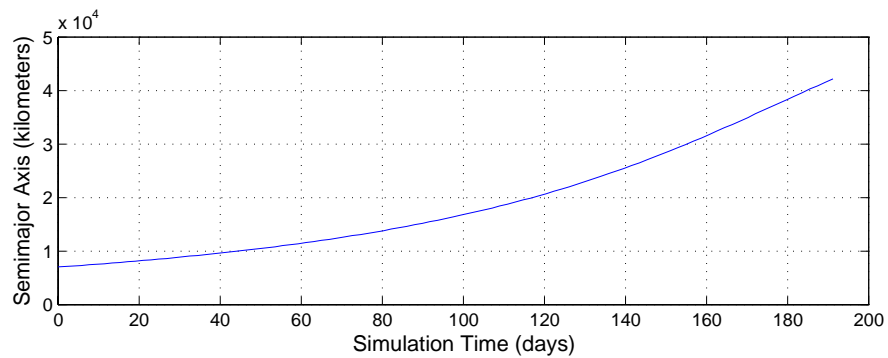
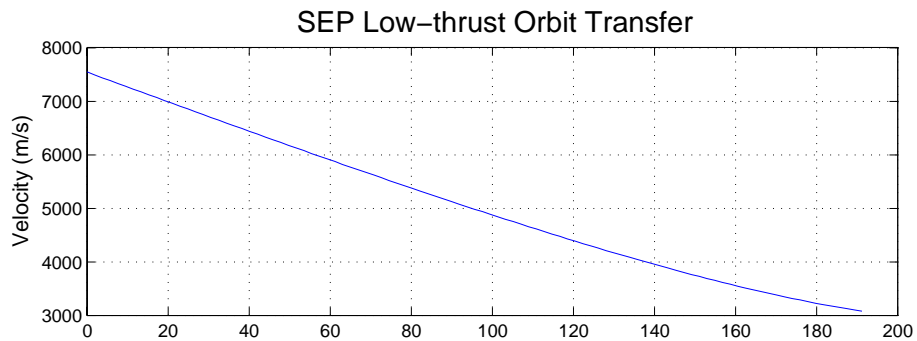
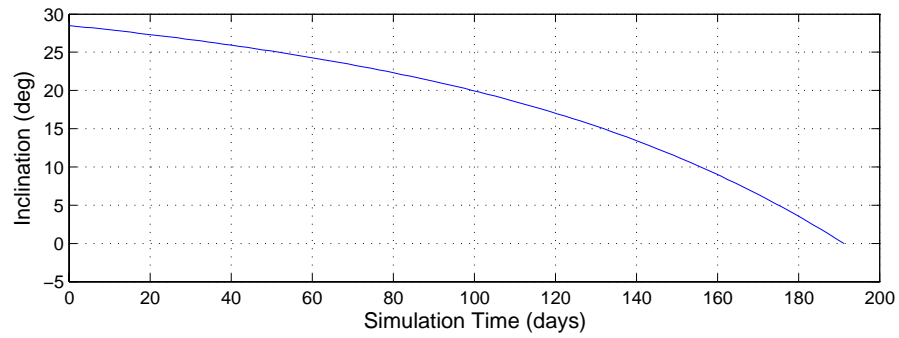
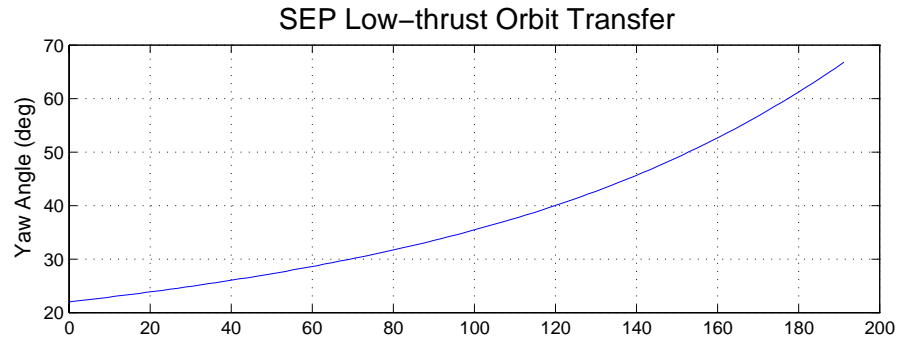
please input the SEP specific impulse (seconds)
? 3300

The following is the output created for this example.

initial orbit altitude	621.8600 kilometers
initial orbit inclination	28.5000 degrees
initial orbit velocity	7546.0538 meters/second
final orbit altitude	35787.8600 kilometers
final orbit inclination	0.0000 degrees
final orbit velocity	3074.5936 meters/second
propulsive efficiency	0.6500
input power	10.0000 kilowatts
specific impulse	3300.0000 seconds
thrust	0.4017 newtons
initial spacecraft mass	1147.7326 kilograms
final spacecraft mass	959.8933 kilograms
propellant mass	187.8393 kilograms
total inclination change	28.5000 degrees
total delta-v	5783.7751 meters/second
thrust duration	191.2624 days
initial yaw angle	21.9850 degrees
thrust acceleration	0.000350 meters/second^2

Orbital Mechanics with MATLAB

The software will also graphically display the time evolution of the thrust vector yaw angle, spacecraft speed, inclination change and semimajor axis. The graphics for this example are as follows:



Solar Sail Trajectory Analysis with MATLAB

This document describes an interactive MATLAB script named `ss2d_opt.m` that can be used to analyze and optimize two-dimensional, heliocentric solar sail trajectories between the orbits of Earth and Venus and between the Earth and Mars. In this script, the heliocentric planet orbits are assumed to be circular and coplanar. The optimal steering angles for minimum transfer time are modeled as piecewise-linear variations as suggested in “Near Minimum-Time Trajectories for Solar Sails”, by Michiel Otten and Colin R. McInnes, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 24, No. 3.

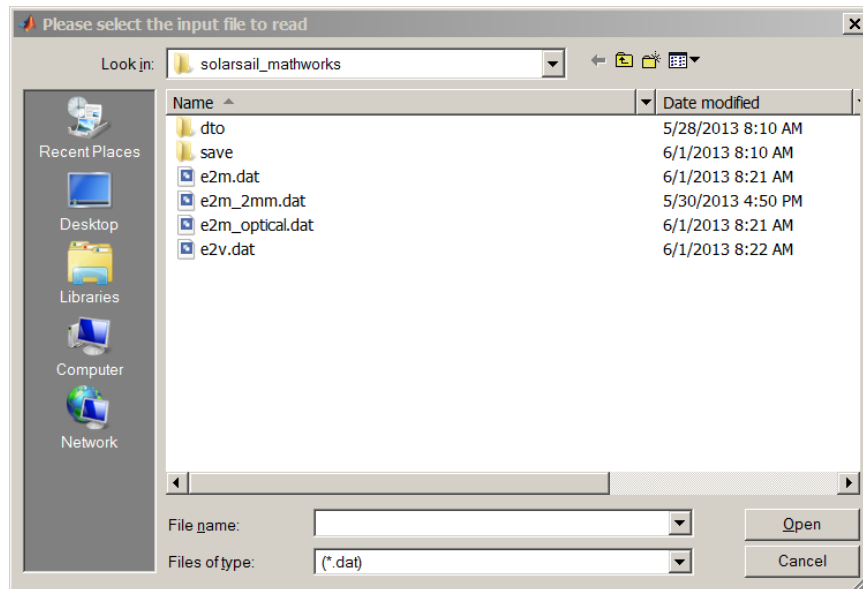
The optimization of these steering angles is performed using the SNOPT nonlinear programming (NLP) algorithm. MATLAB versions of SNOPT for several computer platforms can be found at Professor Philip Gill’s web site which is located at <http://scicomp.ucsd.edu/~peg/>. Professor Gill’s web site also includes a PDF version of the SNOPT software user’s guide.

This MATLAB script solves the problem of transferring from the Earth’s orbit and arriving at the orbit of either Venus or Mars with the proper distance and velocity. It does not attempt to solve the rendezvous problem between the actual or ephemeris locations of each planet. The `ss2d_opt` script provides both numerical and graphical information about the trajectory analysis.

Interacting with the script

To execute the `ss2d_opt` script, log into the directory containing the source code and type `ss2d_opt` in the MATLAB command window.

The `ss2d_opt` MATLAB script is “data driven” by a simple text file created by the user. The script will prompt the user for the name of the data file with a screen similar to



The file type defaults to names with a `*.dat` filename extension. However, you can select any compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

Orbital Mechanics with MATLAB

The following are the contents of a typical `ss2d_opt` compatible input data file. Please note the proper units. User input is denoted in bold font.

```
*****
* input data file for ss2d_opt.m
* Earth-to-Mars trajectory - ideal sail
*****

number of trajectory segments
25

characteristic acceleration (meters/second**2)
0.001

optical force model coefficient b1 (b1 = 0 = ideal)
0.0

optical force model coefficient b2 (b2 = 1 = ideal)
1.0

optical force model coefficient b3 (b3 = 0 = ideal)
0.0

target planet (1 = Venus, 2 = Mars)
2

initial guess for mission duration (days)
400

lower bound for mission duration (days)
250

upper bound for mission duration (days)
500

initial guess for steering angles (degrees)
20.0

lower bound for steering angles (degrees)
0.0

upper bound for steering angles (degrees)
90.0
```

The following is the numerical information output by the script for this example.

```
program ss2d_opt - Earth-to-Mars

number of segments 25

initial state vector

radius                1.00000000 (AU)
radial velocity       0.00000000 (AU/day)
transverse velocity   1.00000000 (AU/day)

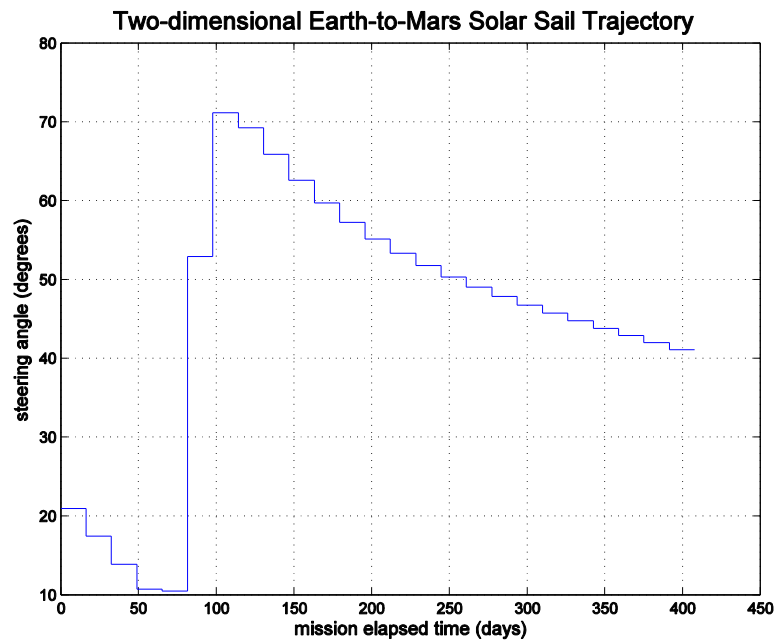
final state vector

radius                1.52368000 (AU)
```

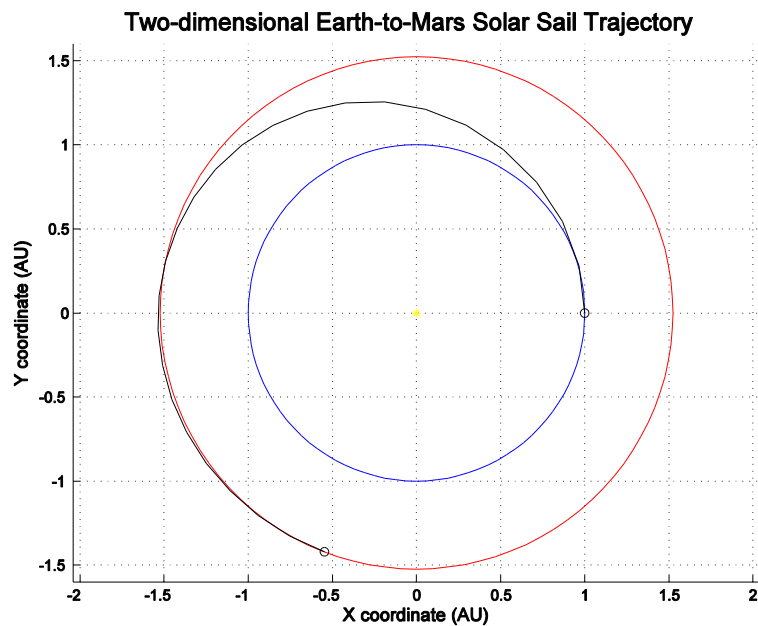
Orbital Mechanics with MATLAB

```
radial velocity      0.00000000 (AU/day)
transverse velocity  0.81012702 (AU/day)
total transfer time  7.01572813 (non-dimensional)
total transfer time  407.84140084 days
```

The following is a plot of the optimal piecewise-linear steering angles during the transfer.

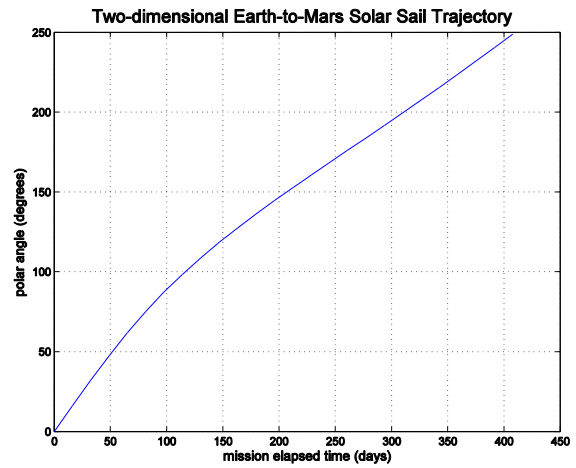
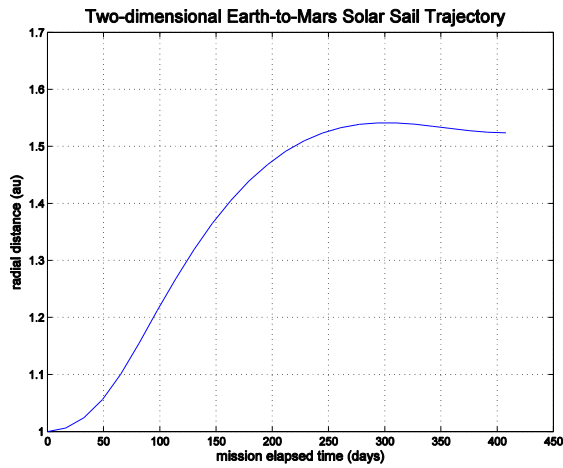


This plot illustrates the transfer trajectory between the Earth (blue) and Mars (red). The beginning and end of the transfer is marked with a small **o**. Please note the scale for the x- and y-coordinates are Astronomical Units.

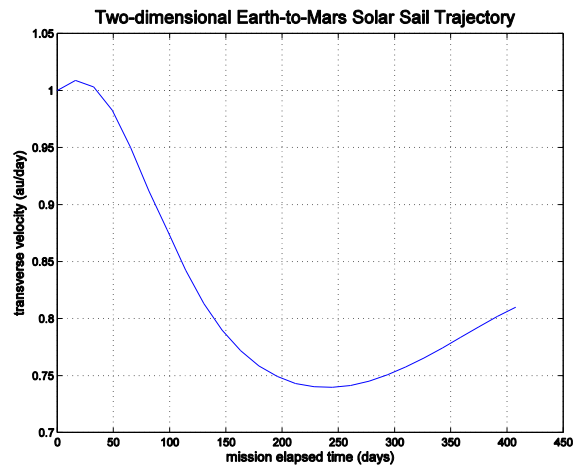
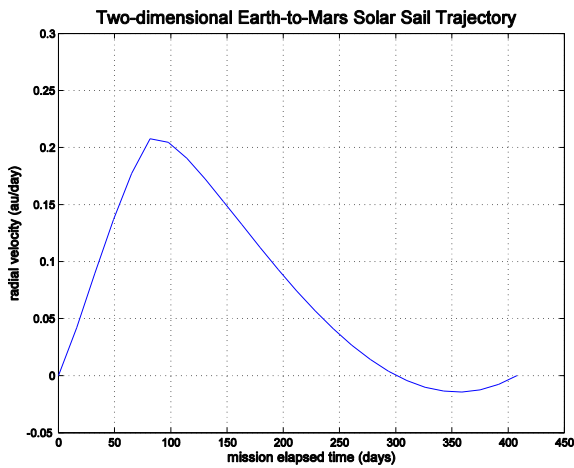


Orbital Mechanics with MATLAB

The following two plots illustrate the behavior of the radial distance and polar angle during the mission. The scale for the radial distance is Astronomical Units and the polar angle scale is degrees.



These next two plots illustrate the behavior of the radial and tangential or transverse velocities during the transfer. The scale for both velocity plots is AU/day.



The `ss2d_opt` MATLAB script will create color Postscript disk files of these graphic images. These images include a TIFF preview and are created with MATLAB code similar to

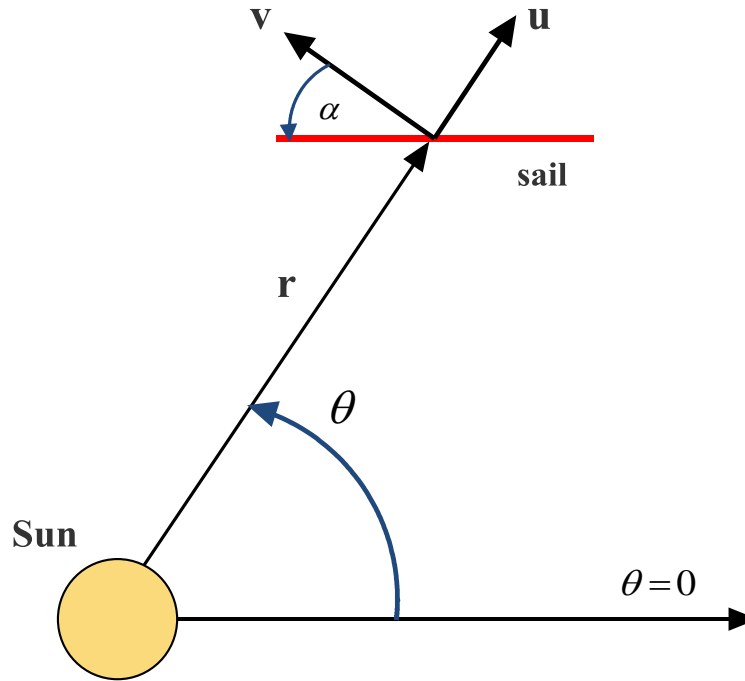
```
print -depsc -tiff -r300 polar_angle.eps
```

Additional script examples can be found in Appendix A.

Technical Discussion

This trajectory optimization problem is modeled in a two-dimensional polar coordinate system. The planets are assumed to be in circular and coplanar heliocentric orbits. No allowance is made for the eccentricity or orbital inclination of the planetary orbits. Therefore, all orbital motion is confined to the ecliptic plane.

The following diagram illustrates the geometry of this coordinate system along with the orientation of the steering angle.



In this diagram, \mathbf{r} is the heliocentric distance of the solar sail, θ is the polar angle, \mathbf{v} is the transverse or tangential component of the velocity, \mathbf{u} is the radial component of the solar sail's velocity and α is the steering or solar sail orientation angle. The steering angle is measured relative to the tangential direction and is positive in the counter-clockwise direction.

Solar radiation pressure

Space travel by solar sail is made possible by solar radiation pressure (SRP). The solar radiation force results from the impingement of photons on the reflective, Sun-facing surface of the solar sail.

The solar radiation pressure at any heliocentric distance r is given by

$$P = P_{\oplus} \left(\frac{r_{\oplus}}{r} \right)^2 = \frac{S_0}{c} \left(\frac{r_{\oplus}}{r} \right)^2$$

where

$$P_{\oplus} = 4.563 \mu\text{N}/\text{m}^2 = \text{SRP at 1 AU}$$

$$S_0 = 1368 \text{ W}/\text{m}^2 \text{ at 1 AU}$$

$$r_{\oplus} = \text{distance at 1 AU}$$

$$c = \text{speed of light}$$

Dimensional analysis

To “streamline” the numerical calculations, the fundamental distance, velocity and time in the equations of motion are normalized. In this MATLAB script, all heliocentric distances are normalized with respect to the Astronomical Unit which is equal to 149597870.691 kilometers. Likewise, all velocity values are normalized with respect to the “local circular velocity” at the heliocentric distance of the Earth’s circular orbit, r_0 . Therefore, the velocity unit is $\sqrt{\mu_{\oplus}/r_0} = \sqrt{\mu_{\oplus}}$ since r_0 is equal to 1 AU.

Finally, all time values are normalized with respect to $\sqrt{r_0^3/\mu_{\oplus}} = \sqrt{1/\mu_{\oplus}}$ since again r_0 is equal to 1 AU. In these equations, μ_{\oplus} is the gravitational constant of the Sun. The corresponding value for this astronomical constant is $\mu_{\oplus} = 0.0002959122082855912 \text{ AU}^3/\text{day}^2$.

Equations of motion

The two-dimensional equations of motion in the polar coordinate system are given by

$$\dot{r} = u$$

$$\dot{\theta} = \frac{v}{r}$$

$$\dot{u} = \frac{v^2}{r} - \frac{1}{r^2} + \left(\frac{\tilde{a}}{r^2} \right) \cos \alpha (b_1 + b_2 \cos^2 \alpha + b_3 \cos \alpha)$$

$$\dot{v} = -\frac{uv}{r} + \left(\frac{\tilde{a}}{r^2} \right) \cos \alpha \sin \alpha (b_2 \cos \alpha + b_3)$$

where

r = radial distance

θ = polar angle

u = radial velocity

v = transverse (tangential) velocity

α = solar sail steering angle

\tilde{a} = acceleration ratio

b_1, b_2, b_3 = sail optical properties

The acceleration ratio is the ratio of the of the acceleration due to SRP and the acceleration due to the point-mass gravity of the Sun, both evaluated at a distance of 1 AU. The SRP acceleration is also called the *characteristic* acceleration which is defined to be the acceleration experienced by an ideal solar sail oriented perpendicular to the direction of the Sun at a heliocentric distance of 1 AU.

Therefore, the acceleration ratio $\tilde{\alpha}$ is equal to a_c/a_{\oplus} where a_c is the characteristic acceleration and a_{\oplus} is the solar acceleration. Values for the characteristic acceleration are typically 0.5 to 1.0 millimeters per second². The acceleration due to the Sun is equal to μ/r^2 . If we convert the characteristic acceleration in meters per second to normalized units according to

$$a_c \rightarrow \frac{m}{s^2} \cdot \left[\frac{(86400 s/day)^2}{1000 m/km \cdot 149597870.691 km/AU} \right]$$

we will have characteristic acceleration in units of AU per day. Likewise, we can convert the solar acceleration to the same unit according to

$$a_{\oplus} = \frac{\mu}{r^2} \rightarrow \frac{(km^3/s^2)}{km^2} = \left(\frac{km}{s^2} \right) \cdot \frac{(86400 s/day)^2}{149597870.691 km/AU}$$

The sail optical properties are a function of the method and material used to manufacture the sail. For an ideal solar sail, $b_1 = b_3 = 0$ and $b_2 = 1$. According to “Solar sail trajectories with piecewise-constant steering laws”, by Giovanni Mengali and Alessandro A. Quarta, *Aerospace Science and Technology*, 13 (2009) 431-441, the values for a solar sail with a highly reflective aluminum-coated front side and a highly emissive chromium-coated back side are $b_1 = 0.0864$, $b_2 = 0.8272$ and $b_3 = -0.00545$.

Trajectory optimization

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 $\mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), t] = 0$.

For any mission time t there are also simple bounds on the state variables

$$\mathbf{y}_l \leq \mathbf{y}(t) \leq \mathbf{y}_u$$

the control variables

$$\mathbf{u}_l \leq \mathbf{u}(t) \leq \mathbf{u}_u$$

and the problem parameters

$$\mathbf{p}_l \leq \mathbf{p}(t) \leq \mathbf{p}_u$$

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.

In this MATLAB script, the total transfer time is the objective function which we are attempting to minimize. The control variables are the steering angles in each time segment. The final boundary conditions or equality constraints are the heliocentric distance and velocities at the destination planet.

The initial conditions are fixed to the normalized values $r_0 = 1, u_0 = 0, v_0 = 1, \theta_0 = 0$. The final boundary conditions are $r_f = 0.723331$ for Venus, $r_f = 1.52368$ for Mars, $u_f = 0, v_f = \sqrt{1/r_f}$. The final polar angle is not constrained since we are solving a minimum time orbital transfer problem.

The following is the MATLAB source code that initializes the optimization problem. It establishes the proper initial and final conditions, calculates initial guesses for the steering angles and objective function, and also sets lower and upper bounds on the final dynamic variables and objective function.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initial and final times and states
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% define state vector at initial time

xinitial(1) = 1.0d0;

xinitial(2) = 0.0d0;

xinitial(3) = 1.0d0;

xinitial(4) = 0.0;

% final conditions

if (iplanet == 1)

    % Venus

    xfinal(1) = 0.723331;
    
```

```
else

    % Mars

    xfinal(1) = 1.52368d0;

end

xfinal(2) = 0.0d0;

xfinal(3) = sqrt(1.0d0 / xfinal(1));

% initial guess for non-dimensional transfer time

xg(1) = time_g / tfactor;

% initial guess for steering angles (radians)

xg(2:nsegments + 1) = alpha_g;

% transpose initial guess

xg = xg';

% upper and lower bounds for non-dimensional transfer time

xlb(1) = time_lb / tfactor;

xub(1) = time_ub / tfactor;

% upper and lower bounds for steering angles (radians)

xlb(2:nsegments + 1) = alpha_lb;

xub(2:nsegments + 1) = alpha_ub;

% transpose bounds

xlb = xlb';

xub = xub';

% define lower and upper bounds on objective function (transfer time)

flow(1) = 0.0d0;

fupp(1) = +Inf;

% define bounds on final state vector equality constraints

flow(2) = xfinal(1);
fupp(2) = xfinal(1);

flow(3) = xfinal(2);
fupp(3) = xfinal(2);

flow(4) = xfinal(3);
fupp(4) = xfinal(3);
```

```
flow = flow';
```

```
fupp = fupp';
```

The following is the call to the SNOPT algorithm to solve the problem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% solve solar sail shooting problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

snscreen('on');

[x, f, inform, xmul, fmul] = snopt(xg, xlb, xub, flow, fupp, 'ss2d_shoot');
```

The following is the MATLAB source code for the function that performs the shooting calculations. This function starts with the initial conditions and integrates the equations of motion along each trajectory segment using the current values of the steering angles for each segment.

```
function [f, g] = ss2d_shoot (x)

% objective function and equality constraints

% simple shooting method

% inputs

% x(1) = current value of transfer time (objective function)
% x(2, nsegments) = current values of steering angle alpha

% outputs

% f = vector of equality constraints and
%     objective function evaluated at x

% Orbital Mechanics with MATLAB

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

global tfactor nsegments alpha_wrk xinitial

% compute duration of each time interval (non-dimensional)

deltat = x(1) / nsegments;

% specify number of differential equations

neq = 4;

% truncation error tolerance

tetol = 1.0e-10;

% initialize initial time

ti = -deltat;

% total non-dimensional time of flight
```

```
tof = x(1);

% set initial conditions
yi(1) = xinitial(1);
yi(2) = xinitial(2);
yi(3) = xinitial(3);
yi(4) = xinitial(4);

% step size guess (non-dimensional time)
h = (1200.0 / 86400.0) / tfactor;

% integrate for all segments
for i = 1:1:nsegments
    alpha_wrk = x(i + 1);

    % increment initial and final times
    ti = ti + deltat;
    tf = ti + deltat;

    % integrate from current ti to tf
    yfinal = rkf78('ss2d_eqm_opt', neq, ti, tf, h, tetol, yi);

    % reset integration vector
    yi = yfinal;

    % check for end of simulation
    if (tf >= tof)
        break;
    end
end

% objective function (minimize non-dimensional transfer time)
f(1) = x(1);

% compute equality constraints (final state boundary conditions)
f(2) = yfinal(1);
f(3) = yfinal(2);
f(4) = yfinal(3);

% transpose
```

```
f = f';  
  
% no derivatives  
  
g = [];
```

The following is the MATLAB function that evaluates the two-dimensional equations of motion.

```
function ydot = ss2d_eqm_opt (t, y)  
  
% two-dimensional solar sail polar equations of motion  
  
% required by ss2d_opt.m  
  
% input  
  
% t      = non-dimensional simulation time  
% y(1) = radial distance (r)  
% y(2) = radial component of velocity (u)  
% y(3) = tangential component of velocity (v)  
% y(4) = polar angle (radians)  
  
% output  
  
% ydot(1) = r-dot  
% ydot(2) = u-dot  
% ydot(3) = v-dot  
% ydot(4) = theta-dot  
  
% Orbital Mechanics with MATLAB  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
global b1 b2 b3 acc_srp alpha_wrk  
  
% evaluate equations of motion at current conditions  
  
r = y(1);  
  
u = y(2);  
  
v = y(3);  
  
afactor = (acc_srp / r^2) * cos(alpha_wrk);  
  
% r-dot  
  
ydot(1) = u;  
  
% u-dot  
  
ydot(2) = (v^2 / r) - (1.0 / r^2) + afactor * (b1 + b2 * cos(alpha_wrk)^2 ...  
          + b3 * cos(alpha_wrk));  
  
% v-dot  
  
ydot(3) = -(u * v / r) + afactor * sin(alpha_wrk) * (b2 * cos(alpha_wrk) + b3);
```

```
% theta-dot  
  
ydot(4) = v / r;
```

Algorithm resources

Colin R. McInnes, *Solar Sailing: Technology, Dynamics and Mission Applications*, Springer-Verlag, Berlin, 1999.

J. L. Wright, *Space Sailing*, Gordon and Breach, New York, 1992.

C. G. Sauer, Jr., “Optimum Solar Sail Interplanetary Trajectories”, AIAA Paper 76-0792, 1976.

Kirill Simon and Yuri Zakharov, “Optimization of Interplanetary Trajectories with Solar Sail”, IAF-95-A.2.08.

Bernd Dachwald, “Optimal Solar Sail Trajectories for Mission to the Outer Solar System”, AIAA *Journal of Guidance, Control and Dynamics*, Vol. 28, No. 1, 2005, pp. 173-177.

Giovanni Mengali and Alessandro A. Quarta, “Semi-Analytical Method for the Analysis of Solar Sail Heliocentric Orbit Raising”, AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 1, January-February 2012.

Giovanni Mengali and Alessandro A. Quarta, “Optimal Three-Dimensional Interplanetary Rendezvous Using Nonideal Solar Sail”, AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 1, January-February 2005.

Bernd Dachwald and Malcolm Macdonald, “Parametric Model and Optimal Control of Solar Sails with Optical Degradation”, AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 5, September-October 2006.

Guido Colasurdo and Lorenzo Casalino, “Optimal Control for Interplanetary Trajectories with Nonideal Solar Sail”, AIAA *Journal of Spacecraft and Rockets*, Vol. 40, No. 2, March-April 2003.

Victoria L. Coverstone and John E. Prussing, “Technique for Escape from Geosynchronous Transfer Orbit Using a Solar Sail”, AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 4, July-August 2003.

Appendix A

Additional Script Examples

This appendix summarizes input data files and script results for two additional examples. The first example is an Earth-to-Venus transfer with an ideal solar sail and the second example illustrates an Earth-to-Mars mission with a non-ideal solar sail.

Earth-to-Venus with ideal solar sail

For this example, the steering angles are negative since they are bounded by $-90^\circ \leq \alpha \leq 0^\circ$. These bounds force the script to fly an interplanetary transfer to an inner planet.

```
*****
* input data file for ss2d_opt.m
* Earth-to-Venus trajectory - ideal sail
*****

number of trajectory time segments
35

characteristic acceleration (meters/second**2)
0.001

optical force model coefficient b1 (b1 = 0 = ideal)
0.0

optical force model coefficient b2 (b2 = 1 = ideal)
1.0

optical force model coefficient b3 (b3 = 0 = ideal)
0.0

target planet (1 = Venus, 2 = Mars)
1

initial guess for mission duration (days)
200

lower bound for mission duration (days)
150

upper bound for mission duration (days)
250

initial guess for steering angles (degrees)
-20.0

lower bound for steering angles (degrees)
-90.0

upper bound for steering angles (degrees)
0.0
```

The following is the script output for this example.

```
program ss2d_opt - Earth-to-Venus

number of segments 35
```


Orbital Mechanics with MATLAB

initial state vector

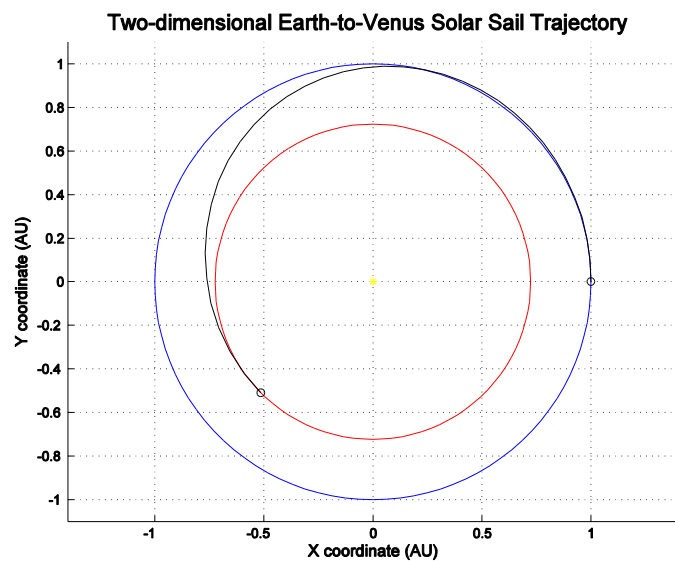
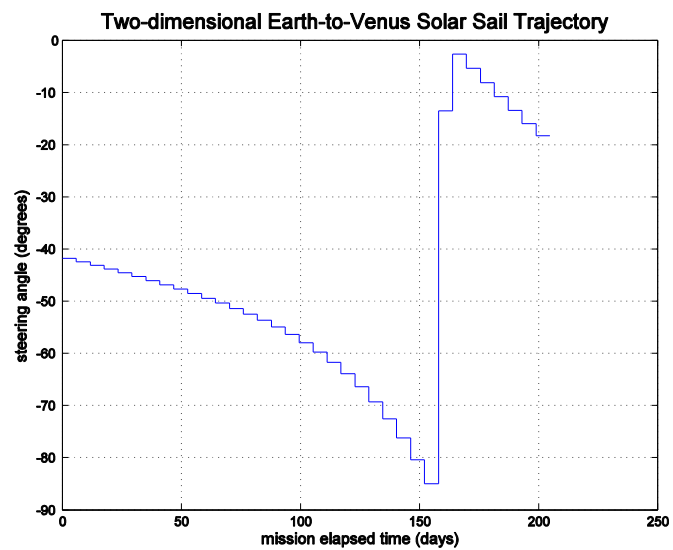
radius 1.00000000 (AU)
radial velocity 0.00000000 (AU/day)
transverse velocity 1.00000000 (AU/day)

final state vector

radius 0.72333100 (AU)
radial velocity -0.00000001 (AU/day)
transverse velocity 1.17579460 (AU/day)

total transfer time 3.52308069 (non-dimensional)

total transfer time 204.80527993 days



Earth-to-Mars with optical solar sail

This example uses 50 trajectory segments and models a typical non-ideal solar sail.

```
*****
* input data file for ss2d_opt.m
* Earth-to-Mars trajectory - non-ideal sail
*****

number of trajectory segments
50

characteristic acceleration (meters/second**2)
0.001

optical force model coefficient b1 (b1 = 0 = ideal)
0.0864

optical force model coefficient b2 (b2 = 1 = ideal)
0.8272

optical force model coefficient b3 (b3 = 0 = ideal)
-5.45e-3

target planet (1 = Venus, 2 = Mars)
2

initial guess for mission duration (days)
400

lower bound for mission duration (days)
250

upper bound for mission duration (days)
500

initial guess for steering angles (degrees)
20.0

lower bound for steering angles (degrees)
0.0

upper bound for steering angles (degrees)
90.0
```

The following is the script output for this example.

```
program ss2d_opt - Earth-to-Mars

number of segments 50

initial state vector

radius                1.00000000 (AU)

radial velocity       0.00000000 (AU/day)

transverse velocity   1.00000000 (AU/day)

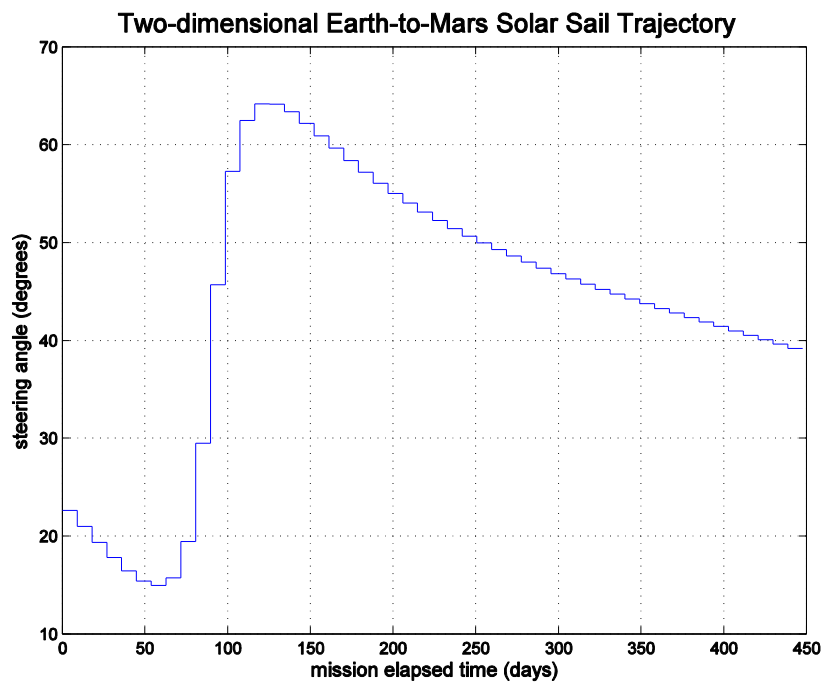
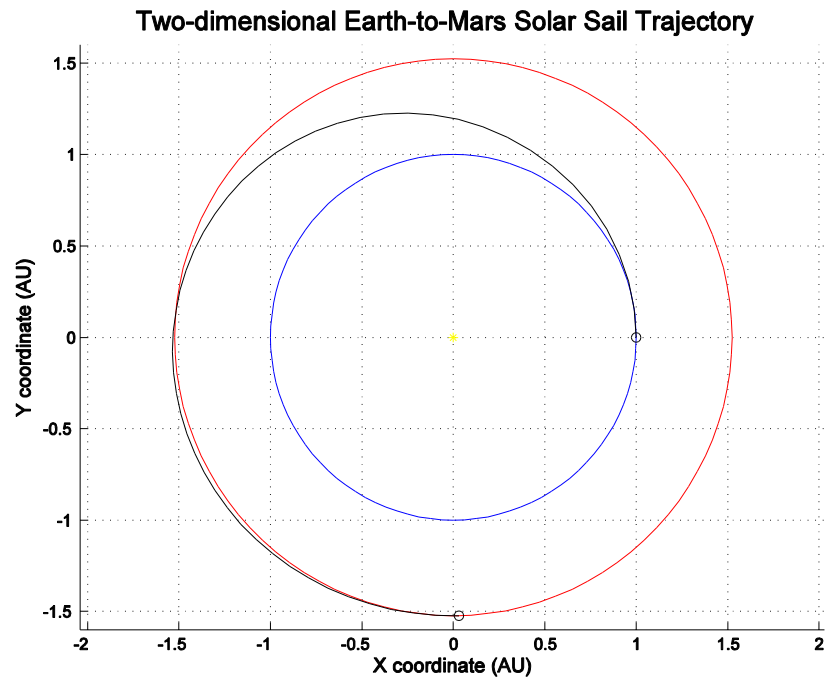
final state vector

radius                1.52367998 (AU)
```

Orbital Mechanics with MATLAB

radial velocity 0.00000000 (AU/day)
transverse velocity 0.81012702 (AU/day)

total transfer time 7.70083775 (non-dimensional)
total transfer time 447.66849522 days



Bi-elliptic Transfer Between Coplanar Circular Orbits

This document describes a MATLAB script called `bielliptic.m` that can be used to determine the characteristics of a time-free, three impulse bi-elliptic transfer between two coplanar circular orbits. The impulsive ΔV assumption means that the velocity, but not the position, of the space vehicle is changed instantaneously. This script also creates graphic displays of the three-dimensional orbits and transfer trajectory, and the evolution of the primer vector and its derivative.

For the coplanar bi-elliptic transfer, all three velocity impulses are confined to the orbital planes of the initial and final orbits. The first impulse creates an elliptical transfer orbit with a perigee altitude equal to the altitude of the initial circular orbit and an apogee altitude well beyond the altitude of the final orbit. The second impulse creates a second transfer ellipse with an apogee radius equal to that of the first transfer ellipse and a perigee radius identical to the value of the final orbit radius.

The third impulsive maneuver circularizes the second transfer orbit at perigee of the transfer orbit. The first two impulses are *prograde* which means that they are in the direction of orbital motion. The third impulse is retrograde since it must slow down the spacecraft for insertion into the final mission orbit.

For final to initial radius ratios greater than 15.58, the bi-elliptic transfer requires less total propulsive energy than the Hohmann transfer. It can also be shown that an outer (intermediate apogee altitude greater than the final orbit altitude) bi-elliptic transfer is more efficient than an inner transfer.

Interacting with the script

The following is a typical user interaction with this script. User inputs to the script are in bold font. Please note that the script will either accept a user-defined intermediate altitude or calculate the optimum value using Brent's root-finding algorithm.

```
Bi-elliptic Orbit Transfer Analysis
```

```
please input the initial altitude (kilometers)
? 300.0
```

```
please input the final altitude (kilometers)
? 5000.0
```

```
type of intermediate altitude computation
```

```
<1> optimal
```

```
<2> user-defined
```

```
selection (1 or 2)
? 2
```

```
please input the bi-elliptic altitude (kilometers)
? 10000.0
```

Orbital Mechanics with MATLAB

The following is the script output created for this example.

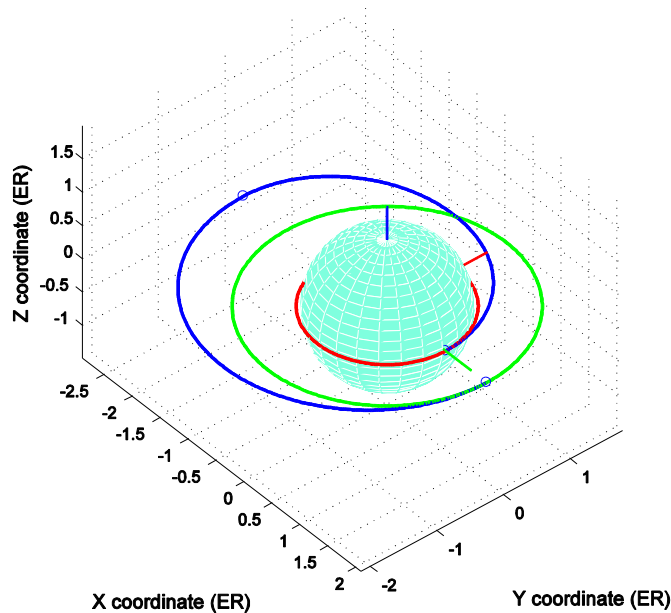
Bi-elliptic Orbit Transfer Analysis

```
-----  
  
initial orbit altitude          300.0000 kilometers  
initial orbit radius            6678.1363 kilometers  
initial orbit velocity          7725.7606 meters/second  
  
first ellipse perigee altitude   300.0000 kilometers  
first ellipse perigee radius     6678.1363 kilometers  
first ellipse apogee altitude   10000.0000 kilometers  
first ellipse apogee radius     16378.1363 kilometers  
first ellipse perigee velocity   9208.6069 meters/second  
first ellipse apogee velocity   3754.7820 meters/second  
first ellipse eccentricity       0.42070981  
  
second ellipse perigee altitude  5000.0000 kilometers  
second ellipse perigee radius    11378.1363 kilometers  
second ellipse apogee altitude  10000.0000 kilometers  
second ellipse apogee radius    16378.1363 kilometers  
second ellipse perigee velocity  6429.8373 meters/second  
second ellipse apogee velocity  4466.9042 meters/second  
second ellipse eccentricity      0.18013946  
  
final orbit altitude            5000.0000 kilometers  
final orbit radius              11378.1363 kilometers  
final orbit velocity            5918.7953 meters/second  
  
first delta-v                   1482.8463 meters/second  
second delta-v                  712.1221 meters/second  
third delta-v                   511.0420 meters/second  
total delta-v                   2706.0105 meters/second  
  
first ellipse transfer time      1.7109 hours  
                                0.0713 days  
  
second ellipse transfer time     2.2598 hours  
                                0.0942 days  
  
total transfer time              3.9707 hours  
                                0.1654 days
```

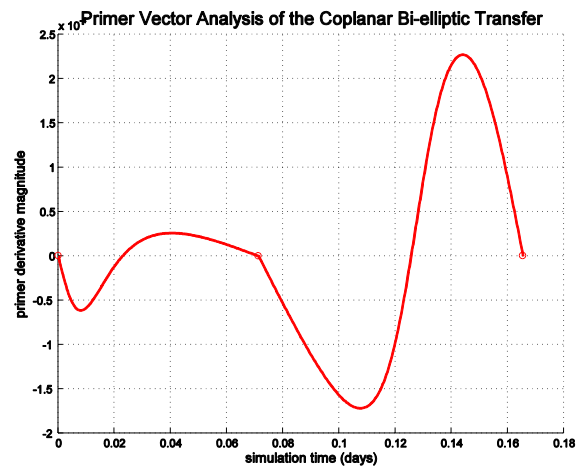
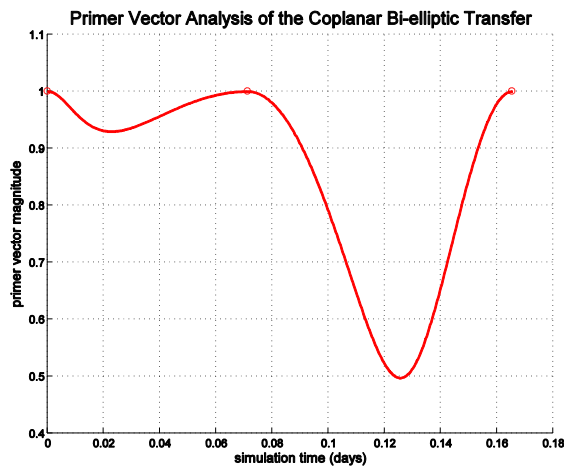
Orbital Mechanics with MATLAB

The `bielliptic` MATLAB script will also create graphic displays of the three-dimensional initial, final and transfer trajectory and the evolution of the primer vector and its derivative. The first graphic image is a three-dimensional display of the solution. In this image, the initial orbit is red, the final orbit is green, and the transfer trajectory is blue. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis. The interactive graphic features of MATLAB allow the user to rotate and zoom the display. These capabilities allow the user to interactively find the best viewpoint as well as verify basic three-dimensional geometry of the orbital transfer.

Bi-elliptic Transfer: Initial, Transfer and Final Orbits



These next two plots illustrate the evolution of the primer vector and its derivative as a function of time, in days, since the first impulse. The location of each impulse is marked with a small red circle.



The `bielliptic` MATLAB script will also create color Postscript disk files of these graphic images. Each image includes a TIFF preview and is created with MATLAB source code similar to

```
print -depsc -tiff -r300 bielliptic1.eps
```

Orbital Mechanics with MATLAB

For comparison, here are the characteristics of this mission using a two impulse Hohmann transfer.

```
Hohmann Orbit Transfer Analysis
-----

initial orbit altitude          300.0000 kilometers
initial orbit radius            6678.1363 kilometers
initial orbit inclination       0.0000 degrees
initial orbit velocity          7725.7606 meters/second

final orbit altitude            5000.0000 kilometers
final orbit radius              11378.1363 kilometers
final orbit inclination         0.0000 degrees
final orbit velocity            5918.7953 meters/second

first inclination change        0.0000 degrees
second inclination change       0.0000 degrees
total inclination change        0.0000 degrees

first delta-v                   947.4074 meters/second
second delta-v                  828.2781 meters/second
total delta-v                   1775.6855 meters/second

transfer orbit semimajor axis   9028.1363 kilometers
transfer orbit eccentricity     0.26029736
transfer orbit inclination      0.0000 degrees
transfer orbit perigee velocity 8673.1680 meters/second
transfer orbit apogee velocity  5090.5171 meters/second
transfer orbit coast time       4268.5281 seconds
                                71.1421 minutes
                                1.1857 hours
```

Since the radius ratio (r_f/r_i) for this example is less than 15.58, the Hohmann orbit transfer is more efficient, in terms of the total ΔV required, than the bi-elliptic transfer.

The following is the script output for this same example where we allow the software to compute the optimal intermediate altitude. Since the radius ratio is less than 15.58, the script finds a two impulse Hohmann transfer. For this situation, the apogee altitude of the intermediate transfer ellipses is equal to the altitude of the final circular orbit.

Orbital Mechanics with MATLAB

Bi-elliptic Orbit Transfer Analysis

initial orbit altitude	300.0000 kilometers
initial orbit radius	6678.1363 kilometers
initial orbit velocity	7725.7606 meters/second
first ellipse perigee altitude	300.0000 kilometers
first ellipse perigee radius	6678.1363 kilometers
first ellipse apogee altitude	5000.0003 kilometers
first ellipse apogee radius	11378.1366 kilometers
first ellipse perigee velocity	8673.1680 meters/second
first ellipse apogee velocity	5090.5170 meters/second
first ellipse eccentricity	0.26029737
second ellipse perigee altitude	5000.0000 kilometers
second ellipse perigee radius	11378.1363 kilometers
second ellipse apogee altitude	5000.0003 kilometers
second ellipse apogee radius	11378.1366 kilometers
second ellipse perigee velocity	5918.7953 meters/second
second ellipse apogee velocity	5918.7952 meters/second
second ellipse eccentricity	0.00000001
final orbit altitude	5000.0000 kilometers
final orbit radius	11378.1363 kilometers
final orbit velocity	5918.7953 meters/second
first delta-v	947.4074 meters/second
second delta-v	828.2781 meters/second
third delta-v	0.0000 meters/second
total delta-v	1775.6856 meters/second
first ellipse transfer time	1.1857 hours 0.0494 days
second ellipse transfer time	1.6776 hours 0.0699 days
total transfer time	2.8633 hours 0.1193 days

Orbital Mechanics with MATLAB

Here's the script output and primer plots for an orbit transfer example where we allow the software to compute the optimal apogee altitude of the two intermediate transfer ellipses. For this example, the altitude of the initial circular orbit is 300 kilometers and the altitude of the final circular orbit is 100,000 kilometers ($r_f/r_i = 15.929$).

Bi-elliptic Orbit Transfer Analysis

```
-----
initial orbit altitude          300.0000 kilometers
initial orbit radius            6678.1363 kilometers
initial orbit velocity          7725.7606 meters/second

first ellipse perigee altitude   300.0000 kilometers
first ellipse perigee radius     6678.1363 kilometers
first ellipse apogee altitude   10631435.2731 kilometers
first ellipse apogee radius     10637813.4094 kilometers
first ellipse perigee velocity   10922.4475 meters/second
first ellipse apogee velocity    6.8568 meters/second
first ellipse eccentricity       0.99874524

second ellipse perigee altitude 100000.0000 kilometers
second ellipse perigee radius    106378.1363 kilometers
second ellipse apogee altitude  10631435.2731 kilometers
second ellipse apogee radius    10637813.4094 kilometers
second ellipse perigee velocity  2723.9367 meters/second
second ellipse apogee velocity   27.2394 meters/second
second ellipse eccentricity      0.98019802

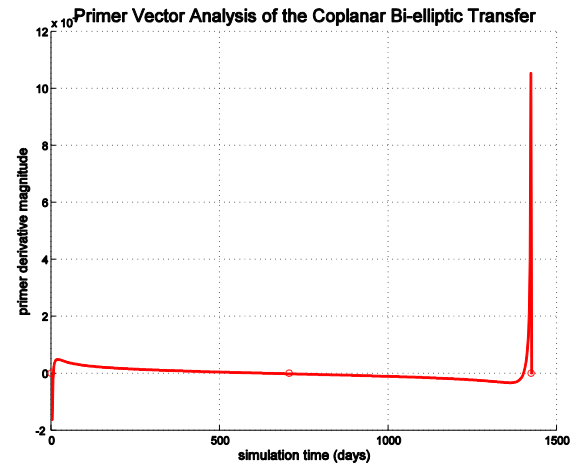
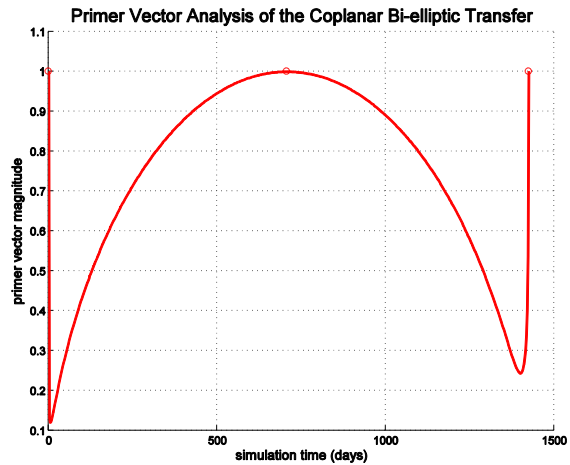
final orbit altitude            100000.0000 kilometers
final orbit radius              106378.1363 kilometers
final orbit velocity            1935.7207 meters/second

first delta-v                   3196.6869 meters/second
second delta-v                  20.3825 meters/second
third delta-v                   788.2160 meters/second
total delta-v                   4005.2855 meters/second

first ellipse transfer time      16971.5253 hours
                                707.1469 days
```

Orbital Mechanics with MATLAB

second ellipse transfer time	17210.5245 hours
	717.1052 days
total transfer time	34182.0498 hours
	1424.2521 days



Here's the Hohmann transfer solution for this example. Since the radius ratio for this example is greater than 15.58, the bi-elliptic orbit transfer is more efficient than the Hohmann transfer.

Hohmann Orbit Transfer Analysis

initial orbit altitude	300.0000 kilometers
initial orbit radius	6678.1363 kilometers
initial orbit inclination	0.0000 degrees
initial orbit velocity	7725.7606 meters/second
final orbit altitude	100000.0000 kilometers
final orbit radius	106378.1363 kilometers
final orbit inclination	0.0000 degrees
final orbit velocity	1935.7207 meters/second
first inclination change	0.0000 degrees
second inclination change	0.0000 degrees
total inclination change	0.0000 degrees
first delta-v	2872.5124 meters/second
second delta-v	1270.3893 meters/second
total delta-v	4142.9017 meters/second
transfer orbit semimajor axis	56528.1363 kilometers

Orbital Mechanics with MATLAB

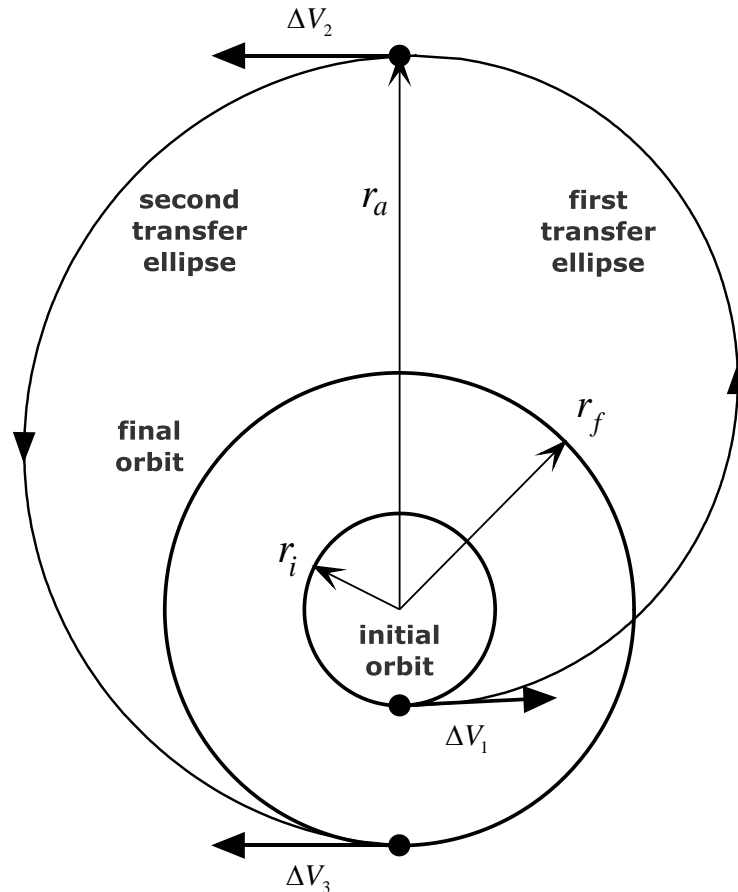
```

transfer orbit eccentricity      0.88186173
transfer orbit inclination      0.0000 degrees
transfer orbit perigee velocity 10598.2730 meters/second
transfer orbit apogee velocity  665.3314 meters/second
transfer orbit coast time      66877.1857 seconds
                                1114.6198 minutes
                                18.5770 hours

```

Technical Discussion

The following diagram (not to scale) illustrates the geometry of the coplanar bi-elliptic orbit transfer. In this figure, r_i is the geocentric radius of the initial circular orbit, r_a is the apogee radius of the two transfer ellipses, and r_f is the radius of the final circular orbit. The locations and directions of the first, second and third impulsive maneuvers are labeled $\Delta V_1, \Delta V_2$ and ΔV_3 , respectively.



The total impulsive delta-v for a bi-elliptic orbital transfer is a function of the initial, intermediate and final orbital altitudes. The relationship between geocentric radius and orbital altitude is as follows:

$$r_i = r_e + h_i \quad r_a = r_e + h_a \quad r_f = r_e + h_f$$

where r_i is the geocentric radius of the initial circular park orbit, r_a is the radius at the intermediate impulse, and r_f is the radius of the final circular mission orbit. In these equations, h_i, h_a and h_f are the corresponding altitudes and r_e is the radius of the Earth.

The magnitude of the first impulse is

$$\Delta V_1 = v_{p_1} - v_i$$

and is simply the difference between the speed on the initial circular orbit and the perigee speed of the first transfer ellipse. The scalar magnitude of the second impulse is

$$\Delta V_2 = v_{a_2} - v_{a_1}$$

which is the difference between the speed at apogee of the first transfer ellipse and the apogee speed of the second transfer ellipse.

Finally, the scalar magnitude of the final delta-v is

$$\Delta V_3 = v_f - v_{p_2}$$

which is the speed difference between the final circular orbit and the speed at perigee of the second transfer ellipse.

The orbital speeds required for these computations can be determined from

$$\begin{aligned} v_i &= \sqrt{\frac{\mu}{r_i}} & v_f &= \sqrt{\frac{\mu}{r_f}} \\ v_{p_1} &= \sqrt{\frac{2\mu}{r_i} - \frac{\mu}{a_1}} & v_{a_1} &= \sqrt{\frac{2\mu}{r_a} - \frac{\mu}{a_1}} \\ v_{p_2} &= \sqrt{\frac{2\mu}{r_f} - \frac{\mu}{a_2}} & v_{a_2} &= \sqrt{\frac{2\mu}{r_a} - \frac{\mu}{a_2}} \end{aligned}$$

In these equations, μ is the gravitational constant of the central body, and the semimajor axis of each transfer ellipse is computed from

$$a_1 = \frac{r_i + r_a}{2} \quad a_2 = \frac{r_a + r_f}{2}$$

The transfer time from the first impulse to the final impulse is equal to the sum of the half orbital periods of the two transfer ellipses according to

$$\tau = \pi \sqrt{\frac{a_1^3}{\mu}} + \pi \sqrt{\frac{a_2^3}{\mu}}$$

Software implementation

For the optimal intermediate altitude script option, the software calls the built-in bounded minimization MATLAB algorithm to solve for the intermediate apogee altitude that minimizes the total delta-v required for the mission.

The call to the algorithm is as follows:

```
[x, fx, exitflag] = fminbnd('befunc', xmin, xmax);
```

where `befunc` is the objective function for this problem. In the argument list, `xmin` and `xmax` are the lower and upper bounds for the intermediate radius, `x` is the solved-for intermediate altitude, and `fx` is the corresponding total delta-v. In this script, they are equal to the radius of the final orbit and one hundred times this radius as follows

```
xmin = rf;  
  
xmax = 100.0 * rf;
```

The following is the MATLAB source code for the objective function.

```
function fx = befunc (x)  
  
% bi-elliptic radius objective function  
  
% required by bielliptic.m  
  
% Orbital Mechanics with MATLAB  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
global mu ri rf  
  
% semimajor axes of the transfer orbits (kilometers)  
  
sma1 = (ri + x) / 2.0;  
  
sma2 = (x + rf) / 2.0;  
  
% initial orbit velocity (kilometers/second)  
  
vi = sqrt(mu / ri);  
  
% first transfer ellipse periapsis velocity (kilometers/second)  
  
vt1a = sqrt((2.0 * mu / ri) - (mu / sma1));  
  
% first transfer ellipse apoapsis velocity (kilometers/second)  
  
vt1b = sqrt((2.0 * mu / x) - (mu / sma1));  
  
% second transfer ellipse periapsis velocity (kilometers/second)  
  
vt2b = sqrt((2.0 * mu / x) - (mu / sma2));
```

Orbital Mechanics with MATLAB

```
% second transfer ellipse periapsis velocity (kilometers/second)

vt2c = sqrt((2.0 * mu / rf) - (mu / sma2));

% final orbit velocity (kilometers/second)

vf = sqrt(mu / rf);

% compute delta-v contributions (kilometers/second)

dva = abs(vt1a - vi);

dvb = abs(vt2b - vt1b);

dvc = abs(vf - vt2c);

% calculate objective function value

fx = dva + dvb + dvc;
```

The classic bi-elliptic technical paper is “The Bi-elliptical Transfer Between Co-planar Circular Orbits” by Rudolf F. Hoelker and Robert Silber which was published in *Advances in Ballistic Missiles and Space Technology*, Volume 3, Pergamon, Oxford, 1961.

An excellent document that describes impulsive orbital transfers is “Optimal Impulsive Maneuvers in Orbital Transfers” by Silvano Sgubini and Paolo Teofilatto. A PDF version of this document can be downloaded from http://naca.central.cranfield.ac.uk/dcscs/2002/E05a_sgubini.pdf.

Additional information can also be found in Chapter 6 of “Fundamentals of Astrodynamics and Applications” by David A. Vallado, Microcosm Press, 2007.

Primer Vector Analysis

This section summarizes the primer vector analysis included with this MATLAB script. The term primer vector was invented by Derek F. Lawden and represents the adjoint vector for velocity. A technical discussion about primer theory can be found in Lawden’s classic text, *Optimal Trajectories for Space Navigation*, Butterworths, London, 1963. Another excellent resource is “Primer Vector Theory and Applications”, Donald J. Jezewski, NASA TR R-454, November 1975, along with “Optimal, Multi-burn, Space Trajectories”, also by Jezewski.

As shown by Lawden, the following four necessary conditions must be satisfied in order for an impulsive orbital transfer to be *locally optimal*:

- (1) the primer vector and its first derivative are everywhere continuous
- (2) whenever a velocity impulse occurs, the primer is a unit vector aligned with the impulse and has unit magnitude ($\mathbf{p} = \hat{\mathbf{p}} = \hat{\mathbf{u}}_r$ and $\|\mathbf{p}\| = 1$)
- (3) the magnitude of the primer vector may not exceed unity on a coasting arc ($\|\mathbf{p}\| = p \leq 1$)

(4) at all interior impulses (not at the initial or final times) $\mathbf{p} \cdot \dot{\mathbf{p}} = 0$; therefore, $d\|\mathbf{p}\|/dt = 0$ at the intermediate impulses

Furthermore, the scalar magnitudes of the primer vector derivative at the initial and final impulses provide information about how to improve the nominal transfer trajectory by changing the endpoint times and/or moving the impulse times. These four cases for non-zero slopes are summarized as follows;

- If $\dot{p}_0 > 0$ and $\dot{p}_f < 0 \rightarrow$ perform an initial coast before the first impulse and add a final coast after the second impulse
- If $\dot{p}_0 > 0$ and $\dot{p}_f > 0 \rightarrow$ perform an initial coast before the first impulse and move the second impulse to a later time
- If $\dot{p}_0 < 0$ and $\dot{p}_f < 0 \rightarrow$ perform the first impulse at an earlier time and add a final coast after the second impulse
- If $\dot{p}_0 < 0$ and $\dot{p}_f > 0 \rightarrow$ perform the first impulse at an earlier time and move the second impulse to a later time

The primer vector analysis of a two impulse orbital transfer involves the following steps.

First partition the two-body state transition matrix as follows:

$$\Phi(t, t_0) = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{r}}{\partial \mathbf{v}_0} \\ \frac{\partial \mathbf{v}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{v}}{\partial \mathbf{v}_0} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix}$$

where

$$\Phi_{11} = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} \end{bmatrix} = \begin{bmatrix} \partial x / \partial x_0 & \partial x / \partial y_0 & \partial x / \partial z_0 \\ \partial y / \partial x_0 & \partial y / \partial y_0 & \partial y / \partial z_0 \\ \partial z / \partial x_0 & \partial z / \partial y_0 & \partial z / \partial z_0 \end{bmatrix}$$

and so forth.

The value of the primer vector at any time t along a two body trajectory is given by

$$\mathbf{p}(t) = \Phi_{11}(t, t_0)\mathbf{p}_0 + \Phi_{12}(t, t_0)\dot{\mathbf{p}}_0$$

and the value of the primer vector derivative is

$$\dot{\mathbf{p}}(t) = \Phi_{21}(t, t_0)\mathbf{p}_0 + \Phi_{22}(t, t_0)\dot{\mathbf{p}}_0$$

which can also be expressed as

$$\begin{Bmatrix} \mathbf{p} \\ \dot{\mathbf{p}} \end{Bmatrix} = \Phi(t, t_0) \begin{Bmatrix} \mathbf{p}_0 \\ \dot{\mathbf{p}}_0 \end{Bmatrix}$$

The primer vector boundary conditions at the initial and final impulses are as follows:

$$\mathbf{p}(t_0) = \mathbf{p}_0 = \frac{\Delta \mathbf{V}_0}{|\Delta \mathbf{V}_0|} \quad \mathbf{p}(t_f) = \mathbf{p}_f = \frac{\Delta \mathbf{V}_f}{|\Delta \mathbf{V}_f|}$$

These two conditions illustrate that at the locations of velocity impulses, the primer vector is a unit vector in the direction of the corresponding impulse.

The value of the primer vector derivative at the initial time is

$$\dot{\mathbf{p}}(t_0) = \dot{\mathbf{p}}_0 = \Phi_{12}^{-1}(t_f, t_0) \{ \mathbf{p}_f - \Phi_{11}(t_f, t_0) \mathbf{p}_0 \}$$

provided the Φ_{12} sub-matrix is non-singular.

The scalar magnitude of the derivative of the primer vector can be determined from

$$\frac{d\|\mathbf{p}\|}{dt} = \frac{d}{dt}(\mathbf{p} \cdot \mathbf{p})^{\frac{1}{2}} = \frac{\dot{\mathbf{p}} \cdot \mathbf{p}}{\|\mathbf{p}\|}$$

As noted by D. J. Jezewski, the primer vector is sometimes called the Lagrange multiplier, costate vector or perhaps an adjoint variable.

Program oneburn_ocs

Single Maneuver, Finite-Burn Trajectory Optimization

This document is the user's manual for a Fortran computer program called `oneburn_ocs` that uses the *Sparse Optimization Suite* distributed by [Applied Mathematical Analysis](#) to solve an Earth orbit transfer trajectory optimization problem. The software models the trajectory as a single, finite-burn propulsive maneuver followed by a user-defined, time-bounded final coast phase. This computer program attempts to maximize the final spacecraft mass. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass.

The important features of this scientific simulation are as follows:

- single, continuous thrust orbital maneuver
- variable inertial attitude steering
- constant propulsive thrust magnitude
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified final coast phase

The *Sparse Optimization 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 *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010 (www.siam.org).

The `oneburn_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 *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `oneburn_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:

```
oneburn_ocs leo2gto.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 oneburn_ocs           *  
*                                           *  
*   single maneuver, finite-burn          *  
*   trajectory optimization                *  
*                                           *  
*           February 20, 2012              *  
*****  
  
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.

The screen output created by the `oneburn_ocs` computer program can be re-directed to a text file with a command line similar to

```
oneburn_ocs leo2gto.in >leo2gto.txt
```

To create a DOS command window in Windows 7, select **start**, then **All Programs**, then **Accessories** and finally **Command Prompt**. The size, font and other characteristics of the screen can be controlled by the user with the `c:\` icon in the upper left corner of the window. To log into the subdirectory created during the installation of the Fortran executable and support files, type **root:** and then **cd subdirectory** from the DOS command line where **root** is the name of the root directory, usually `c:`, and **subdirectory** is the name of the subdirectory created by the user.

The DOS command line prompt looks similar to `C:\oneburn_ocs>_.`

Input File Format and Contents

The `oneburn_ocs` software is “data-driven” by a user-created text file. This text file should be simple ASCII format with no special characters.

The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in *courier* font and all explanations are in *times italic* font. This example attempts to optimize the maneuver required to transfer a spacecraft from a circular low Earth orbit (LEO) to a typical elliptical geosynchronous transfer orbit (GTO).

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.

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.

```
*****
** earth-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program oneburn_ocs
** leo2gto.in - February 20, 2012
*****
```

The first three inputs define the initial mass prior to the propulsive maneuver, and the thrust magnitude and specific impulse of the upper stage or spacecraft propulsion system.

```
initial spacecraft mass (kilograms)
10000.0

thrust magnitude (newtons)
99200.0

specific impulse (seconds)
450.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

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

The next two numeric inputs define either the user's initial guess for the delta-v magnitude or the maneuver duration, and should be consistent with the previous input.

```
initial guess for delta-v (meters/second)
2800.0

initial guess for thrust duration (seconds)
550.0
```

The next two inputs define the lower and upper bounds for the thrust duration. These inputs are required for either type of propulsive initial guess.

```
lower bound for thrust duration (seconds)
0.01

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

The next section of the input data file lets the user define the characteristics of a final coast phase that follows the propulsive maneuver. These three inputs define an initial guess for the coast duration as well as lower and upper bounds on the coast duration.

```
*****
coast maneuver
*****
```

```

initial guess for coast duration (seconds)
20.0

lower bound for coast duration (seconds)
1.0

upper bound for coast duration (seconds)
2000.0

```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```

*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
6563.14d0

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
0.0d0

true anomaly (degrees)
0.0

```

This next integer input allows the user to define the type of initial orbit constraints to use during the simulation.

```

*****
initial orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----
3

```

The next six inputs define the classical orbital elements of the final mission orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```

*****
* FINAL ORBIT *
*****

semimajor axis (kilometers)
24364.8d0

orbital eccentricity (non-dimensional)
0.73062206d0

orbital inclination (degrees)
26.3355d0

argument of perigee (degrees)
270.0d0

```

```

right ascension of the ascending node (degrees)
0.0d0

true anomaly (degrees)
0.0d0

```

This next integer input allows the user to define the type of final orbit constraints to use during the simulation.

```

*****
final orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
-----
1

```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```

*****
* type of gravity model *
-----
1 = spherical Earth
2 = oblate gravity model
-----
2

```

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 options *
*****
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 binary initial guess data file
leo2gto.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 data file (yes or no)
no

```

This next input specifies the type of solution data file to create.

```
*****
* type of comma-delimited solution data file *
*****
1 = OCS-defined nodes
2 = user-defined nodes
3 = user-defined step size
-----
2
```

For options 2 or 3, this 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
100
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
leo2gto.csv
```

The next series of program inputs are algorithm control options and parameters for the Sparse Optimization Suite. 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 input defines the relative error in the objective function.

```
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.

```
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
10000
```

The level of output from the Sparse Optimization Suite NLP algorithm is controlled with the following integer input.

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

The level of output from the Sparse Optimization Suite 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 Sparse Optimization Suite 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 Sparse Optimization Suite 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 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 oneburn_ocs software. Please consult Appendix C.

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

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

```

name of optimal control configuration file
leo2gto_config.txt

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

name of optimal control configuration file
leo2gto_config1.txt

```

Optimal control solution

The following is the optimal control solution for this example. This example used variable attitude steering during the propulsive maneuver, and a final bounded coast. The output includes the time and orbital characteristics at the beginning and end of the propulsive maneuver. This example optimizes the maneuver required to transfer from a circular low Earth orbit (LEO) to a typical elliptical geosynchronous transfer orbit (GTO). Appendix A contains a brief summary of this information.

```

program oneburn_ocs
=====

input file ==> leo2gto.in

oblate earth gravity model

-----
beginning of finite burn
-----

mission elapsed time      00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.656314000000D+04    0.303099251680D-16    0.285000000000D+02    0.000000000000D+00

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.343626272476D+03    0.284333562833D+03    0.284333562833D+03    0.881915957810D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
-.164199672837D+02    -.581964995764D+04    -.303417392626D+04    0.656314000000D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.772230207356D+01    -.501755544829D+00    0.920593794451D+00    0.779315089527D+01

-----
end of finite burn
-----

mission elapsed time      00:03:29.965

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.243647798517D+05    0.730621883714D+00    0.263354841203D+02    0.269999803084D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.125070318737D-03    0.159279650912D+02    0.285927768175D+03    0.630817068563D+03

      rx (km)      ry (km)      rz (km)      rmag (km)
0.183083308977D+04    -.574950274971D+04    -.284601509915D+04    0.667147162297D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.100245469314D+02    0.145694376803D+01    0.721178679773D+00    0.101555071272D+02

```


The following program output is the final spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v.

final mass	5280.17375040397	kilograms
propellant mass	4719.82624959603	kilograms
thrust duration	209.965300814227	seconds
delta-v	2818.25253032923	meters/second

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

This section of the numeric results summarizes the time and orbital conditions at the beginning and end of the final coast.

```

-----
beginning of coast maneuver
-----

mission elapsed time      00:03:29.965

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.243647798517D+05    0.730621883714D+00    0.263354841203D+02    0.269999803084D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.125070318734D-03    0.159279650912D+02    0.285927768175D+03    0.630817068563D+03

      rx (km)      ry (km)      rz (km)      rmag (km)
0.183083308977D+04    -.574950274971D+04    -.284601509915D+04    0.667147162297D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.100245469314D+02    0.145694376803D+01    0.721178679773D+00    0.101555071272D+02

-----
end of coast maneuver
-----

mission elapsed time      00:03:30.965

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.243648000000D+05    0.730622060000D+00    0.263355000000D+02    0.270000000000D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.360000000000D+03    0.160144835450D+02    0.286014483545D+03    0.630817851038D+03

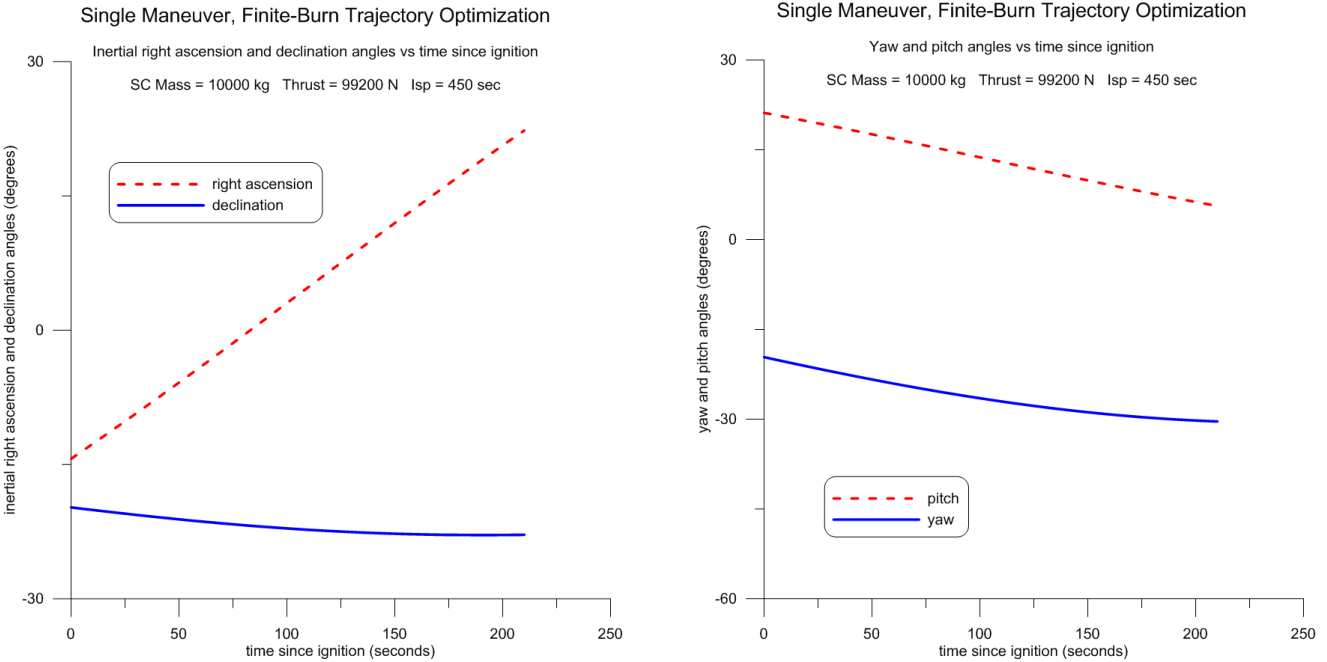
      rx (km)      ry (km)      rz (km)      rmag (km)
0.184085640569D+04    -.574804194749D+04    -.284529200486D+04    0.667266252177D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.100220828732D+02    0.146465973235D+01    0.725009431797D+00    0.101544577367D+02

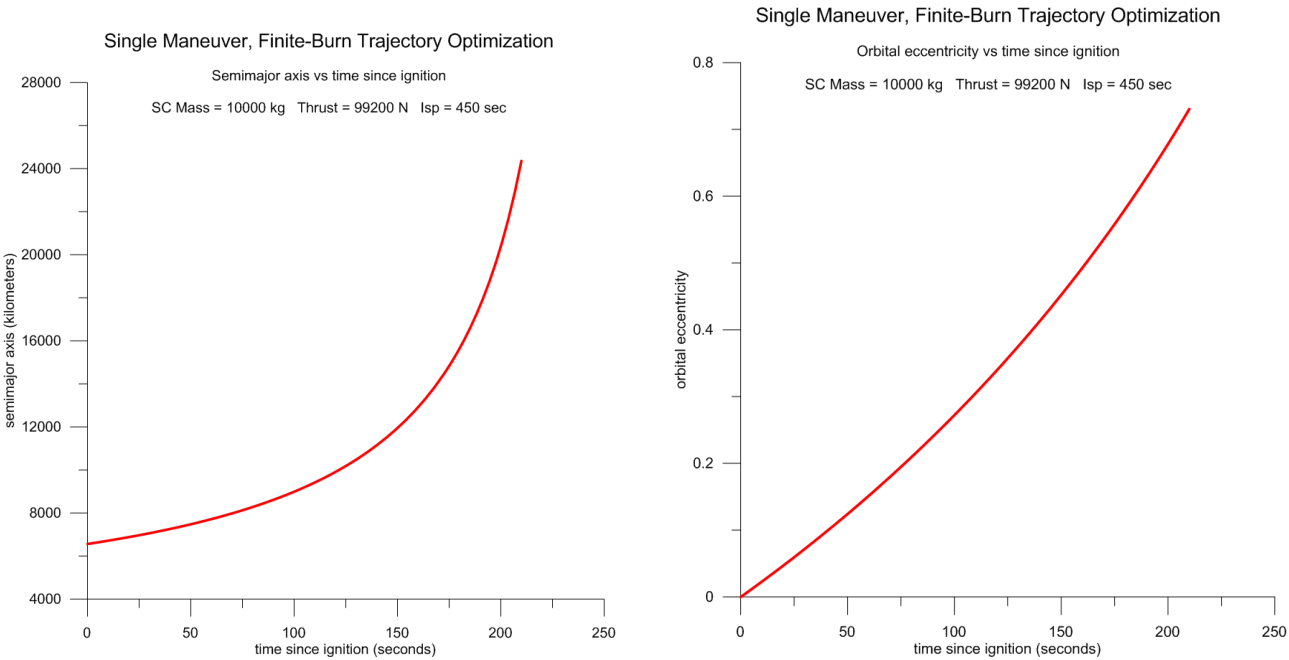
coast duration      1.00000000000151      seconds
1.666666666669177E-002      minutes

```

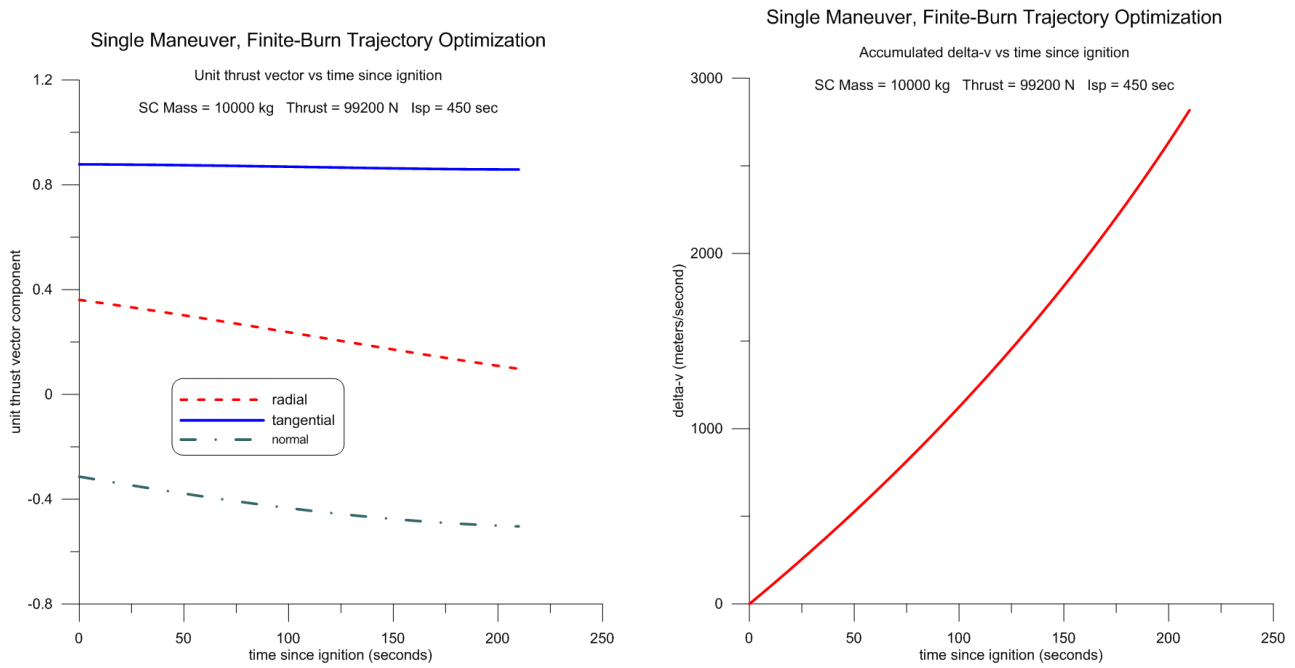
The following plots illustrate the evolution of the inertial right ascension, declination and pitch and yaw angles during this finite-burn maneuver.



The next two plots illustrate the evolution of the semimajor axis and orbital eccentricity of the transfer orbit during this finite-burn maneuver.

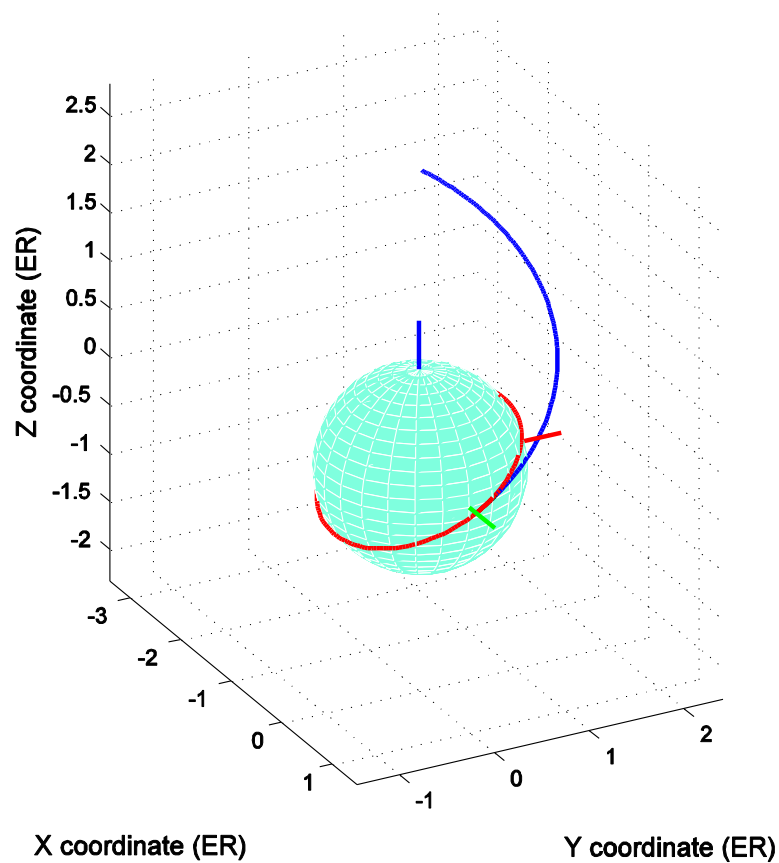


These final two plots illustrate the behavior of the radial, tangential and radial components of the unit thrust vector, and the accumulated delta-v during the propulsive maneuver.



The following is a graphics display of the initial (red trace) and final orbits (blue trace).

Initial and Final Orbits



Verification of the optimal control solution

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

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

```
=====
verification of optimal control solution
=====
```

```
-----
end of finite burn
-----
```

mission elapsed time 00:03:29.965

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.243647798517D+05	0.730621883714D+00	0.263354841203D+02	0.269999803084D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.125070318737D-03	0.159279650912D+02	0.285927768175D+03	0.630817068563D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.183083308977D+04	-.574950274971D+04	-.284601509915D+04	0.667147162297D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.100245469314D+02	0.145694376803D+01	0.721178679773D+00	0.101555071272D+02

final mass	5280.17375040377	kilograms
propellant mass	4719.82624959623	kilograms
thrust duration	209.965300814227	seconds
delta-v	2818.25213904842	meters/second

final mission orbit

mission elapsed time 00:03:30.965

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.243648000037D+05	0.730622059923D+00	0.263355000035D+02	0.270000000226D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.359999999997D+03	0.160144833255D+02	0.286014483551D+03	0.630817851180D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.184085640604D+04	-.574804194714D+04	-.284529200507D+04	0.667266252165D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.100220828707D+02	0.146465974729D+01	0.725009439586D+00	0.101544577369D+02

In addition to the user-defined solution output file, the `oneburn_ocs` program will create two additional comma-separated-variable data files named `orbits.csv` and `maneuver.csv`. The first file contains the position vectors of the initial and final orbits normalized with respect to the radius of the Earth. The second data file contains the information described in Appendix A starting at ignition and ending at burnout of the propulsive maneuver.

Creating an initial guess

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

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

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

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

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

In these equations

m_i = initial mass

m_f = final mass

m_p = propellant mass

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

I_{sp} = specific impulse

ΔV = impulsive velocity increment

F = thrust

g = acceleration of gravity

The software requires an initial guess for the thrust duration. The user should also provide lower and upper bounds for the total thrust duration. All of these inputs should be in seconds. If the *Sparse Optimization Suite* cannot find a feasible solution, try increasing the guess for thrust duration.

The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting, 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. It works best when the initial and final orbits are nearly coplanar.

The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory

optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the program to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

Binary restart data files can also be used to initialize a `oneburn_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 may be necessary to make successive small changes in the mission definition and run multiple simulations to eventually reach the final desired solution.

Problem setup

This part of the user's manual provides details about the software implementation within `oneburn_ocs`. It defines such things as point and path constraints (boundary conditions), bounds on the dynamic variables, and the performance index or objective function.

(1) Point functions – initial orbit constraints

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

- 1) constrain semimajor axis, eccentricity and inclination
- 2) constrain all initial orbital elements
- 3) option 2 with unconstrained true longitude

For option 1, the initial orbit inclination is constrained by enforcing

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

where i is the initial orbit inclination.

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

$$f = 0 \quad \text{and} \quad g = 0$$

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

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

is enforced, where e is the initial orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the initial modified equinoctial orbital elements as follows:

$$p_L = p_U = p_i$$

$$f_L = f_U = f_i$$

$$g_L = g_U = g_i$$

$$h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the initial true longitude unbounded.

In optimal control terminology, these derived 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 or termination of the propulsive maneuver. This is simply

$$J = m_f$$

The value of the `maxmin` indicator in the Sparse Optimization Suite algorithm tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

(3) Path constraint – unit thrust vector scalar magnitude

For a *variable steering* trajectory, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows:

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

(4) Point functions – final mission orbit constraints

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

- 1) constrain semimajor axis, eccentricity and inclination
- 2) constrain all final orbital elements
- 3) option 2 with unconstrained true longitude

For option 1, the final orbit inclination is constrained by enforcing

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

where i is the mission orbit inclination.

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

$$f = 0 \quad \text{and} \quad g = 0$$

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

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

is enforced, where e is the park orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the user-defined final modified equinoctial orbital elements as follows:

$$p_L = p_U = p_i$$

$$f_L = f_U = f_i$$

$$g_L = g_U = g_i$$

$$h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the final true longitude unbounded.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05m_{sc_i} \leq m_{sc} \leq 1.05m_{sc_i}$$

$$100p_f \leq p \leq 0.8p_i$$

$$-1 \leq f \leq +1$$

$$-1 \leq g \leq +1$$

$$-1 \leq h \leq +1$$

$$-1 \leq k \leq +1$$

where m_{sc_i} is the initial spacecraft mass.

Finally, the three components of the unit thrust vector are constrained as follows

$$-1.1 \leq u_r \leq +1.1$$

$$-1.1 \leq u_t \leq +1.1$$

$$-1.1 \leq u_n \leq +1.1$$

Technical Discussion

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 summarized as follows:

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

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

position vector

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

velocity vector

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

where

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

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

$$\begin{aligned} \dot{p} &= \frac{dp}{dt} = \frac{2p}{w} \sqrt{\frac{p}{\mu}} \Delta_t \\ \dot{f} &= \frac{df}{dt} = \sqrt{\frac{p}{\mu}} \left[\Delta_r \sin L + [(w+1) \cos L + f] \frac{\Delta_t}{w} - (h \sin L - k \cos L) \frac{g \Delta_n}{w} \right] \\ \dot{g} &= \frac{dg}{dt} = \sqrt{\frac{p}{\mu}} \left[-\Delta_r \cos L + [(w+1) \sin L + g] \frac{\Delta_t}{w} + (h \sin L - k \cos L) \frac{f \Delta_n}{w} \right] \\ \dot{h} &= \frac{dh}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \cos L \\ \dot{k} &= \frac{dk}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \sin L \end{aligned}$$

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

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

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

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

where

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

and

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

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

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

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

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

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:

$$\begin{aligned}\Delta_{J_{2r}} &= -\frac{3\mu J_2 R_e^2}{2r^4} \left[1 - \frac{12(h \sin L - k \cos L)^2}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2t}} &= -\frac{12\mu J_2 R_e^2}{r^4} \left[\frac{(h \sin L - k \cos L)(h \cos L + k \sin L)}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2n}} &= -\frac{6\mu J_2 R_e^2}{r^4} \left[\frac{(1 - h^2 - k^2)(h \sin L - k \cos L)}{(1 + h^2 + k^2)^2} \right]\end{aligned}$$

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

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

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

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 and \dot{m} is the propellant flow rate.

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

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

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

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the “local horizontal” and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four quadrant operation.

The `oneburn_ocs` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. The following section summarizes the inertial-to/from-modified equinoctial coordinate transformations and the calculation of the inertial unit thrust vector in terms of right ascension and declination angles.

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

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

where

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

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 `oneburn_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

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

where the calculation for right ascension is a four quadrant inverse tangent operation.

Algorithm Resources

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“Survey of Numerical Methods for Trajectory Optimization”, John T. Betts, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998.

“Optimal Interplanetary Orbit Transfers by Direct Transcription”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

“Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, 20-22 August 1990.

“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.

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.

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

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

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 the CSV data files produced by the `oneburn_ocs` software.

The simulation summary screen display contains the following information:

mission elapsed time = simulation time since beginning of maneuver (hh:mm:ss.sss)
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 spacecraft's position vector in kilometers
ry (km) = y-component of the spacecraft's position vector in kilometers
rz (km) = z-component of the spacecraft's position vector in kilometers
rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers
vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second
vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second
vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second
vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second
final mass = final spacecraft mass in kilograms
propellant mass = expended propellant mass in kilograms
thrust duration = maneuver duration in seconds
delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

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

time (sec) = simulation time since ignition in seconds

time (min) = simulation time since ignition in minutes
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
period (min) = orbital period in minutes
mass (kg) = spacecraft mass in kilograms
thracc (mps/s) = thrust acceleration in meters/second**2
perigee altitude = perigee altitude in kilometers
apogee altitude = apogee altitude in kilometers
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 eci unit thrust vector
ut-eci-y = y-component of eci unit thrust vector
ut-eci-z = z-component of eci unit thrust vector
semi-parameter = orbital semiparameter in kilometers
f equinoctial element = modified equinoctial orbital element
g equinoctial element = modified equinoctial orbital element
h equinoctial element = modified equinoctial orbital element
k equinoctial element = modified equinoctial orbital element
true longitude = true longitude in degrees
rx (km) = x-component of the spacecraft's position vector in kilometers
ry (km) = y-component of the spacecraft's position vector in kilometers
rz (km) = z-component of the spacecraft's position vector in kilometers
rmag (km) = magnitude of spacecraft's position vector in kilometers
vx (km) = x-component of the spacecraft's velocity vector in kilometers/second
vy (km) = y-component of the spacecraft's velocity vector in kilometers/second
vz (km) = z-component of the spacecraft's velocity vector in kilometers/second
vmag (km) = magnitude of spacecraft's velocity vector in kilometers/second
rasc (deg) = inertial right ascension of the unit thrust vector in degrees
decl (deg) = inertial declination of the unit thrust vector in degrees

yaw (deg) = out-of-plane yaw angle of the unit thrust vector in degrees
pitch (deg) = in-plane pitch angle of the unit thrust vector in degrees
fpa (deg) = inertial flight path angle in degrees
deltav (mps) = accumulative delta-v in meters per second

The `orbits.csv` file contains the following information:

time (seconds) = simulation time since ignition in seconds
rp1-x (er) = x-component of the initial orbit position vector in earth radii
rp1-y (er) = y-component of the initial orbit position vector in earth radii
rp1-z (er) = z-component of the initial orbit position vector in earth radii
rp2-x (er) = x-component of the final orbit position vector in earth radii
rp2-y (er) = y-component of the final orbit position vector in earth radii
rp2-z (er) = z-component of the final orbit position vector in earth radii

APPENDIX B

Example LEO-to-LEO Orbit Transfer

This appendix illustrates the orbit transfer for a non-coplanar LEO-to-LEO example. For this example, all the orbital elements except true longitude for both the initial and final orbits are fixed.

The main portion of the simulation definition file for this example is as follows:

```
*****
** earth-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program oneburn_ocs
** leo2leo.in - February 21, 2012
*****

initial spacecraft mass (kilograms)
10000.0

thrust magnitude (newtons)
99200.0

specific impulse (seconds)
450.0

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

initial guess for delta-v (meters/second)
5800.0d0

initial guess for thrust duration (seconds)
4550.0

lower bound for thrust duration (seconds)
0.01

upper bound for thrust duration (seconds)
10000.0

*****
coast maneuver
*****

initial guess for coast duration (seconds)
10.0

lower bound for coast duration (seconds)
1.0

upper bound for coast duration (seconds)
200.0

*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
6563.14
```

```

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
20.0d0

true anomaly (degrees)
0.0d0

*****
initial orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----
3

*****
* FINAL ORBIT *
*****

semimajor axis (kilometers)
6728.14d0

orbital eccentricity (non-dimensional)
0.0d0

orbital inclination (degrees)
51.6d0

argument of perigee (degrees)
0.0d0

right ascension of the ascending node (degrees)
20.0d0

true anomaly (degrees)
0.0d0

*****
final orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
-----
3

*****
* type of gravity model *
-----
1 = spherical Earth
2 = oblate gravity model
-----
2

*****
* initial guess options *
*****

```

```

1 = numerical integration
2 = binary data file
-----
1

name of binary initial guess data file
leo2leo.rsbin

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

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

*****
* type of comma-delimited solution data file *
*****
1 = OCS-defined nodes
2 = user-defined nodes
3 = user-defined step size
-----
1

number of user-defined nodes or print step size in solution data file
100

name of solution output file
leo2leo.csv

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

discretization/collocation method
-----
1 = trapezoidal
2 = separated Hermite-Simpson
3 = compressed Hermite-Simpson
-----
1

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

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

maximum number of mesh refinement iterations
20

maximum number of function evaluations
500000

maximum number of algorithm iterations
10000

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

```

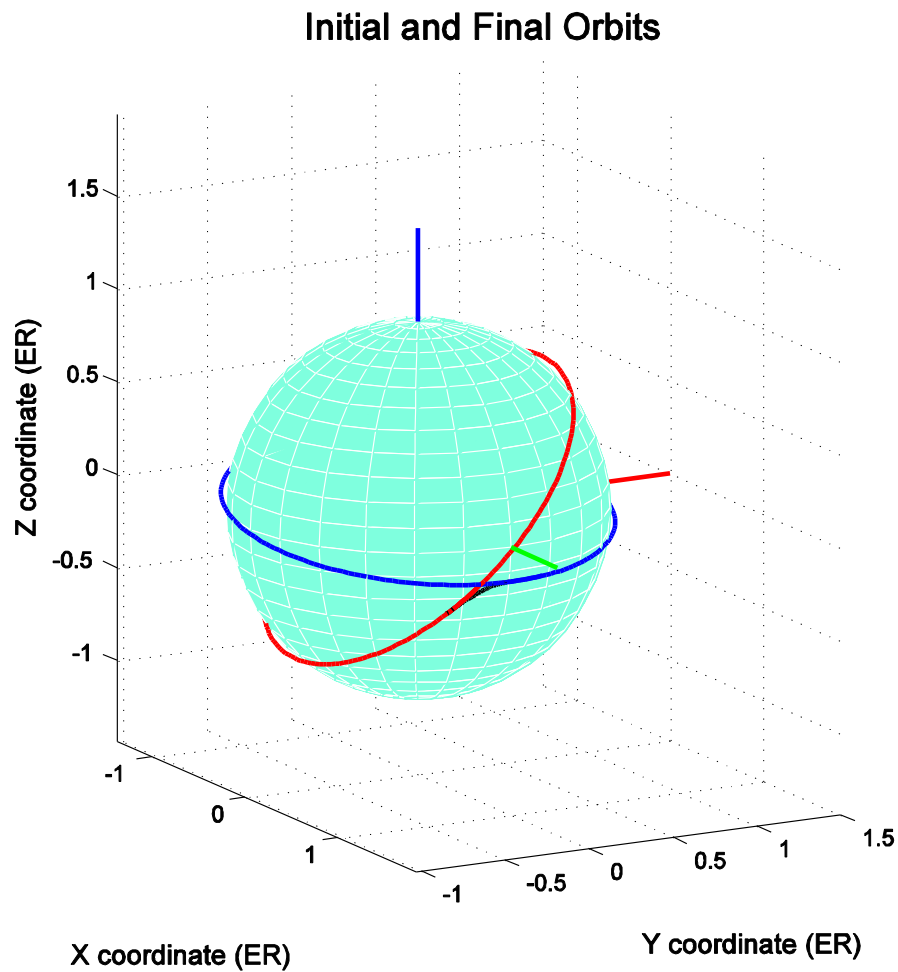
```

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

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

```

The following is the graphics display for this example. The initial orbit trace is blue, the final orbit is red and the transfer maneuver trace is black.



Here are the numerical results created by the oneburn_ocs program for this example.

```

program oneburn_ocs
=====

input file ==> leo2leo.in

oblate earth gravity model

-----
beginning of finite burn
-----

mission elapsed time      00:00:00.000


      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.656314000000D+04    0.231260711765D-15    0.285000000000D+02    0.000000000000D+00

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.200000000000D+02    0.349632240730D+03    0.349632240730D+03    0.881915957810D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
0.642165842152D+04    0.123266976185D+04    -.563591195034D+03    0.656314000000D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-.986248713166D+00    0.681033027729D+01    0.365785673126D+01    0.779315089527D+01

-----
end of finite burn
-----

mission elapsed time      00:04:19.701


      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.672939043691D+04    0.346714306486D-03    0.516042165183D+02    0.315662746264D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.200014039317D+02    0.513704955945D+02    0.703324185873D+01    0.915636904293D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
0.609955227473D+04    0.276472241466D+04    0.645646678312D+03    0.672793338370D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-.250687876646D+01    0.413739153123D+01    0.598798676484D+01    0.769795007313D+01


final mass      4162.16070871686      kilograms
propellant mass      5837.83929128314      kilograms
thrust duration      259.701018232233      seconds
delta-v      3868.21300678244      meters/second

-----
beginning of coast maneuver
-----

mission elapsed time      00:04:19.701


      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.672939043691D+04    0.346714306485D-03    0.516042165183D+02    0.315662746264D+03

```

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.200014039317D+02	0.513704955945D+02	0.703324185873D+01	0.915636904293D+02
rx (km)	ry (km)	rz (km)	rmag (km)
0.609955227473D+04	0.276472241466D+04	0.645646678312D+03	0.672793338370D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.250687876646D+01	0.413739153123D+01	0.598798676484D+01	0.769795007313D+01

end of coast maneuver

mission elapsed time 00:07:39.701

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.672814000000D+04	0.351002915346D-15	0.516000000000D+02	0.000000000000D+00
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.200000000000D+02	0.201443655965D+02	0.201443655965D+02	0.915381704433D+02
rx (km)	ry (km)	rz (km)	rmag (km)
0.544337714471D+04	0.351284568192D+04	0.181588224703D+04	0.672814000000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.402604621147D+01	0.331121293169D+01	0.566309181054D+01	0.769699863782D+01

coast duration	200.000000000023	seconds
	3.3333333333372	minutes

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

end of finite burn

mission elapsed time 00:04:19.701

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.672939043691D+04	0.346714306486D-03	0.516042165183D+02	0.315662746264D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.200014039317D+02	0.513704955945D+02	0.703324185873D+01	0.915636904293D+02
rx (km)	ry (km)	rz (km)	rmag (km)
0.609955227473D+04	0.276472241466D+04	0.645646678312D+03	0.672793338370D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.250687876646D+01	0.413739153123D+01	0.598798676484D+01	0.769795007313D+01

final mass	4162.16070871692	kilograms
propellant mass	5837.83929128308	kilograms
thrust duration	259.701018232233	seconds
delta-v	3868.21189633888	meters/second

final mission orbit

mission elapsed time 00:07:39.701

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.672814024373D+04	0.364896455298D-07	0.515999998019D+02	0.490431457999D+01
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.199999999785D+02	0.152400513000D+02	0.201443658800D+02	0.915381754174D+02
rx (km)	ry (km)	rz (km)	rmag (km)
0.544337713202D+04	0.351284570367D+04	0.181588226839D+04	0.672814000686D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.402604625682D+01	0.331121302687D+01	0.566309190148D+01	0.769699876939D+01

APPENDIX C

Typical Sparse Optimization Suite Configuration File

The `oneburn_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 Sparse Optimization Suite user's manual. Please note that the `oneburn_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 `oneburn_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
TOLMSA=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.rsb
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:PTHTOL=0.100000000000000D+02
INSNLP:RHO1WR=0.100000000000000D+03
INSNLP:IMAXMU=10
INSNLP:MUCALC=3
INSNLP:MXQPIT=1

Program twoburn_ocs

Two Maneuver, Finite-Burn Trajectory Optimization

This document is the user's manual for a Fortran computer program called `twoburn_ocs` that uses the *Sparse Optimization Suite* distributed by [Applied Mathematical Analysis](#) to solve the classic orbit transfer trajectory optimization problem. The software models the trajectory as a three phase mission in the sequence burn-coast-burn. The two burns are simulated as constant-thrust, finite-burn propulsive maneuvers. This computer program attempts to maximize the spacecraft mass at the end of the final propulsive maneuver.

The important features of this scientific simulation are as follows:

- two finite-burn, continuous thrust orbital maneuvers
- variable attitude steering
- constant propulsive thrust magnitude and specific impulse
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified initial and final orbit constraints

The *Sparse Optimization 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 *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010 (www.siam.org).

The `twoburn_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 *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `twoburn_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:

```
twoburn_ocs leo2geo.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 twoburn_ocs      *
*                               *
*      two-maneuver, finite-burn *
*      trajectory optimization  *
*                               *
*      April 15, 2012          *
*****

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.

The screen output created by the `twoburn_ocs` computer program can be re-directed to a text file with a command line similar to

```
twoburn_ocs leo2geo.in >leo2geo.txt
```

To create a DOS command window in Windows 7, select **start**, then **All Programs**, then **Accessories** and finally **Command Prompt**. The size, font and other characteristics of the screen can be controlled by the user with the **c:** icon in the upper left corner of the window. To log into the subdirectory created during the installation of the Fortran executable and support files, type **root:** and then **cd subdirectory** from the DOS command line where **root** is the name of the root directory, usually **c:**, and **subdirectory** is the name of the subdirectory created by the user.

The DOS command line prompt looks similar to **C:\twoburn_ocs>_**.

Input file format and contents

The `twoburn_ocs` software is “data-driven” by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in *times italic* font. This example attempts to optimize the maneuvers required to transfer a spacecraft from a near circular low Earth orbit (LEO) to a typical geosynchronous Earth orbit (GEO).

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.

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.

```
*****
** two maneuver, finite-burn earth-orbit
** trajectory optimization
** program twoburn_ocs
** leo2geo.in - April 15, 2012
*****
```

The first input is the initial mass of the entire spacecraft in kilograms.

```
initial spacecraft mass (kilograms)
15000.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

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

The next four inputs define the thrust magnitude and the specific impulse of the upper stage or spacecraft propulsion system, and the user's initial guess for either the delta-v or thrust duration for the first maneuver.

```
-----
first propulsive maneuver
-----
thrust magnitude (newtons)
25000.0

specific impulse (seconds)
400.0

initial guess for delta-v (meters/second)
2480.0

initial guess for thrust duration (seconds)
170.0
```

The next four inputs define the thrust magnitude and the specific impulse of the upper stage or spacecraft propulsion system, and the user's initial guess for either the delta-v or thrust duration for the second maneuver.

```
-----
second propulsive maneuver
-----

thrust magnitude (newtons)
5000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
1790.0

initial guess for thrust duration (seconds)
170.0
```

The next three inputs define the user's initial guess for the duration of the coast phase along with a lower and upper bound for the coast duration.

```

-----
coast phase
-----

initial guess for coast duration (minutes)
315.0

lower bound for coast duration (minutes)
200.0

upper bound for coast duration (minutes)
400.0

```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```

*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
6563.14

orbital eccentricity (non-dimensional)
0.015

orbital inclination (degrees)
28.5

argument of perigee (degrees)
120.0

right ascension of the ascending node (degrees)
100.0

true anomaly (degrees)
0.0

```

This next integer input allows the user to define the type of initial orbit constraints to use during the simulation. Please see the “Problem setup” section later in this document for information about this program option.

```

*****
initial orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----
1

```

The next six inputs define the classical orbital elements of the final mission orbit. These elements are also defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```

*****
* FINAL ORBIT *
*****

semimajor axis (kilometers)
42166.263

```

```

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
2.5

argument of perigee (degrees)
300.0

right ascension of the ascending node (degrees)
120.0

true anomaly (degrees)
0.0

```

This next integer input allows the user to define the type of final orbit constraints to use during the simulation. Please see the “Problem setup” section later in this document for information about this program option.

```

*****
final orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
-----
3

```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```

*****
* type of gravity model *
-----
1 = spherical Earth
2 = oblate gravity model
-----
2

```

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 options *
*****
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 binary initial guess data file
leo2geo.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 data file (yes or no)
no
```

This next input specifies the type of solution data file to create.

```
*****
* 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 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
25
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
leo2geo.csv
```

The next series of program inputs are algorithm control options and parameters for the Sparse Optimization Suite. 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 input defines the relative error in the objective function.

```
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.

```
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
10000
```

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

```
maximum number of algorithm iterations
10000
```

The level of output from the Sparse Optimization Suite NLP algorithm is controlled with the following integer input.

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

The level of output from the Sparse Optimization Suite 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 Sparse Optimization Suite 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 Sparse Optimization Suite 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 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 rendezvous_ocs software. Please consult Appendix C.

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

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

name of optimal control configuration file
leo2geo_config.txt

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

name of optimal control configuration file
leo2geo_config1.txt
```

Optimal control solution

The following is the twoburn_ocs solution for this example. The output includes the time and orbital characteristics at the beginning and end of each mission phase.

```
program twoburn_ocs
=====

input file ==> leo2geo.in

numerical integration initial guess

oblate earth gravity model

-----
beginning of first propulsive maneuver
-----

mission elapsed time      00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.656314000000D+04    0.150000000004D-01    0.285000000000D+02    0.179196327908D+03

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.119947466863D+03    0.318037646103D+03    0.137233974011D+03    0.881916022527D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
-0.977069953510D+03    -0.606099670550D+04    0.210248702383D+04    0.648928335362D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.708880451413D+01    -0.202549730354D+01    -0.278600998827D+01    0.788134762721D+01

-----
end of first propulsive maneuver
-----

mission elapsed time      00:18:38.659
```

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.244711724047D+05	0.725948454612D+00	0.264721389995D+02	0.179895113440D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.119828797416D+03	0.407077273961D+02	0.220602840836D+03	0.634953461869D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.659297946572D+04	-.275407175855D+04	-.216599500646D+04	0.746617866480D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.487228379563D+01	0.718704821378D+01	-.388504453231D+01	0.951243304473D+01

The following program output is the spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v for the first maneuver.

spacecraft mass	7870.53129355029	kilograms
propellant mass	7129.46870644971	kilograms
thrust duration	1118.65926864158	seconds
	18.6443211440263	minutes
delta-v	2529.82034928729	meters/second

This section of the numeric results summarizes the time and orbital conditions at the beginning and end of the transfer orbit coast.

beginning of coast phase

mission elapsed time 00:18:38.659

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.244711724047D+05	0.725948454612D+00	0.264721389995D+02	0.179895113440D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.119828797416D+03	0.407077273961D+02	0.220602840836D+03	0.634953461869D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.659297946572D+04	-.275407175855D+04	-.216599500646D+04	0.746617866480D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.487228379563D+01	0.718704821379D+01	-.388504453231D+01	0.951243304473D+01

end of coast phase

mission elapsed time 05:07:30.284

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.244242477038D+05	0.725339679047D+00	0.264690756581D+02	0.179961581757D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.119758219971D+03	0.177464376454D+03	0.357425958210D+03	0.633128004987D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.193739413380D+05	0.372905472704D+05	-.841352842717D+03	0.420314452916D+05

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.137738985018D+01	-.487203868378D+00	0.715777169774D+00	0.162693188718D+01

coast duration	17331.6249596538	seconds
	288.860415994230	minutes
	4.81434026657050	hours

beginning of second propulsive maneuver

mission elapsed time 05:07:30.284

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.244242477038D+05	0.725339679047D+00	0.264690756581D+02	0.179961581757D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.119758219971D+03	0.177464376454D+03	0.357425958210D+03	0.633128004987D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.193739413380D+05	0.372905472704D+05	-.841352842717D+03	0.420314452916D+05

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.137738985018D+01	-.487203868378D+00	0.715777169774D+00	0.162693188718D+01

end of second propulsive maneuver

mission elapsed time 05:43:11.886

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.421662630000D+05	0.157018823694D-15	0.250000000000D+01	0.000000000000D+00

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.120000000000D+03	0.383296455222D+01	0.383296455222D+01	0.143617512421D+04

rx (km)	ry (km)	rz (km)	rmag (km)
-.234747395985D+05	0.350273495880D+05	0.122951225622D+03	0.421662630000D+05

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.255141829470D+01	-.171038725638D+01	0.133811476571D+00	0.307458377550D+01

The following program output is the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v for the second maneuver.

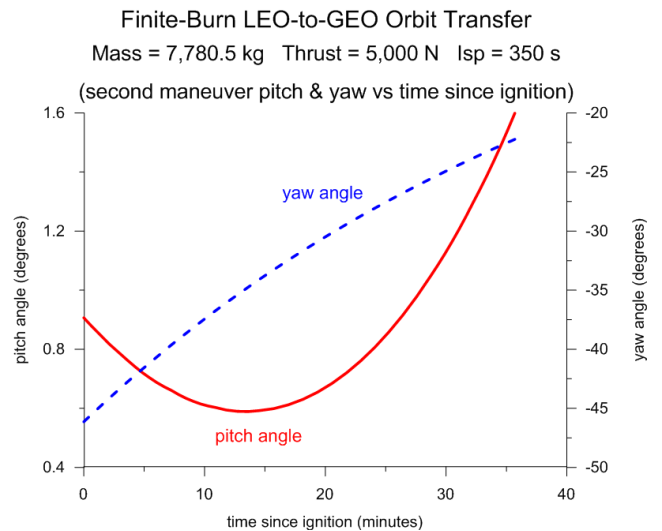
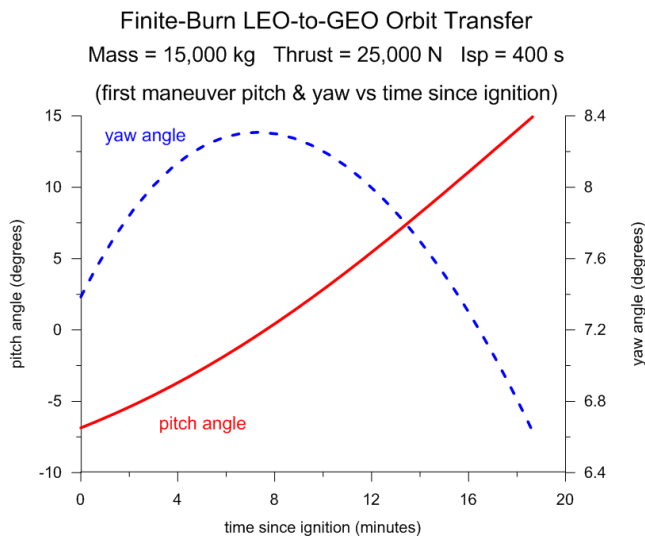
propellant mass	3119.75087432484	kilograms
thrust duration	2141.60134383684	seconds
	35.6933557306139	minutes
delta-v	1732.69627276496	meters/second

After the simulation is complete, the software will display a simulation summary similar to the following;

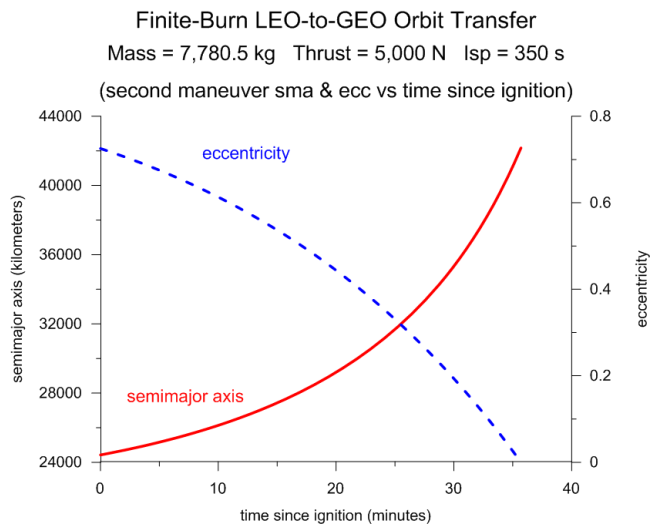
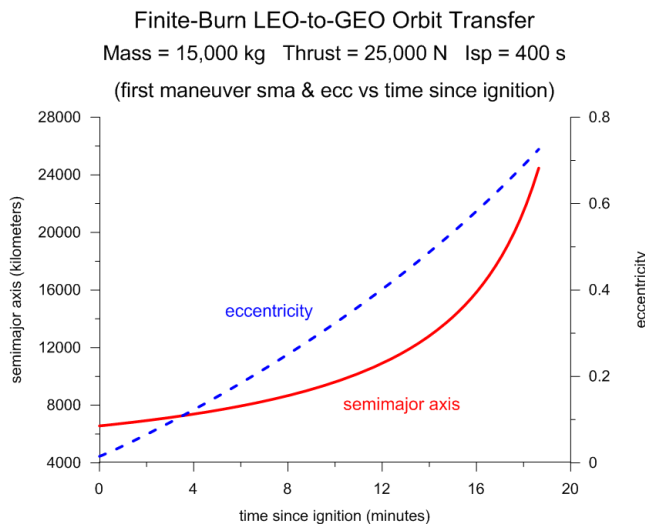
SIMULATION SUMMARY

initial spacecraft mass	15000.0000000000	kilograms
total propellant mass	10249.2195807745	kilograms
final spacecraft mass	4750.78041922545	kilograms
total delta-v	4262.51662205225	meters/second
total thrust duration	3260.26061247841	seconds
	54.3376768746402	minutes

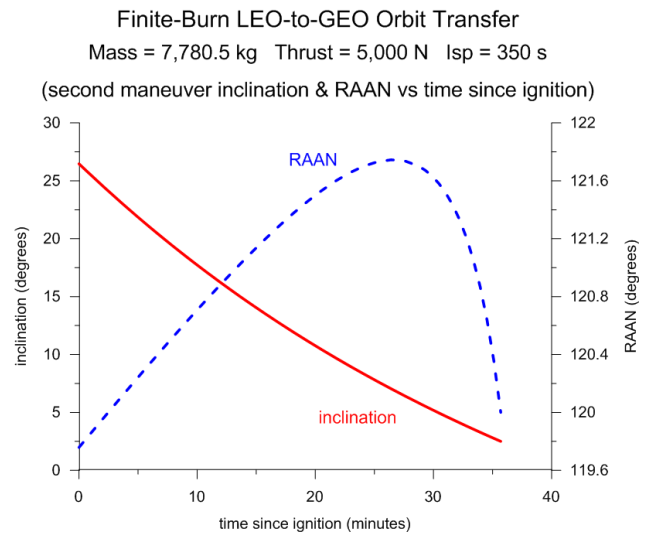
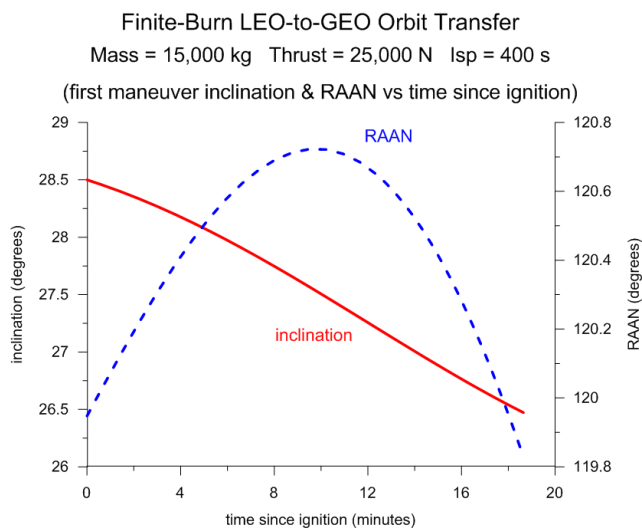
The following two plots illustrate the evolution of the pitch and yaw steering angles during the first and second finite-burn maneuver.



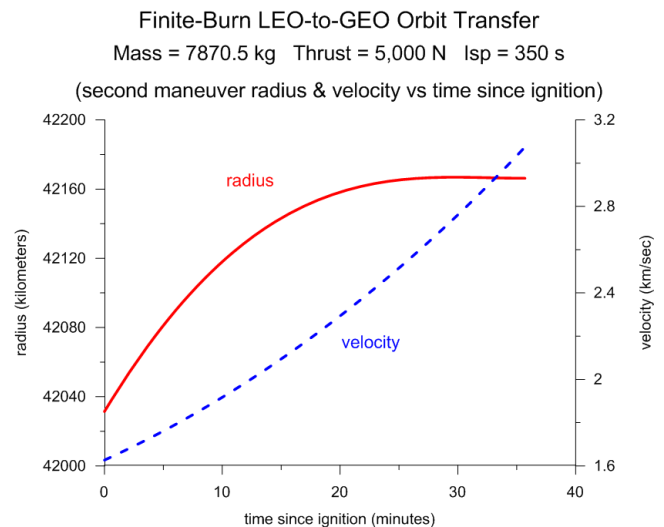
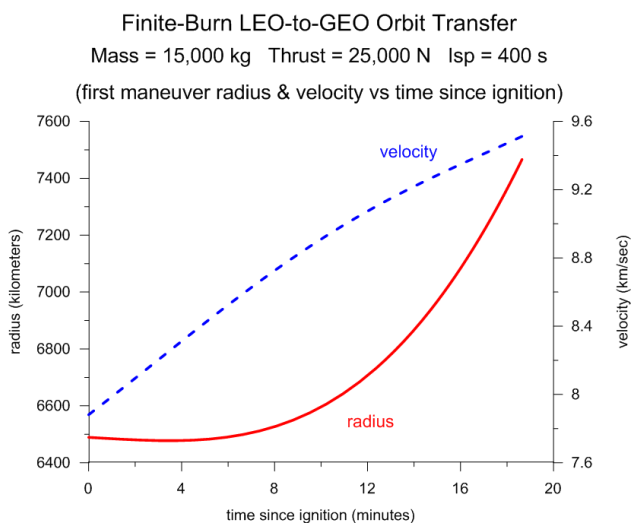
The next pair of plots illustrate the behavior of the semimajor axis and orbital eccentricity during the first and second maneuvers.



The next pair of plots illustrate the behavior of the orbital inclination and right ascension of the ascending node (RAAN) during the first and second maneuvers.



The following two plots illustrate the evolution of the geocentric radius and velocity during the each finite-burn maneuver.

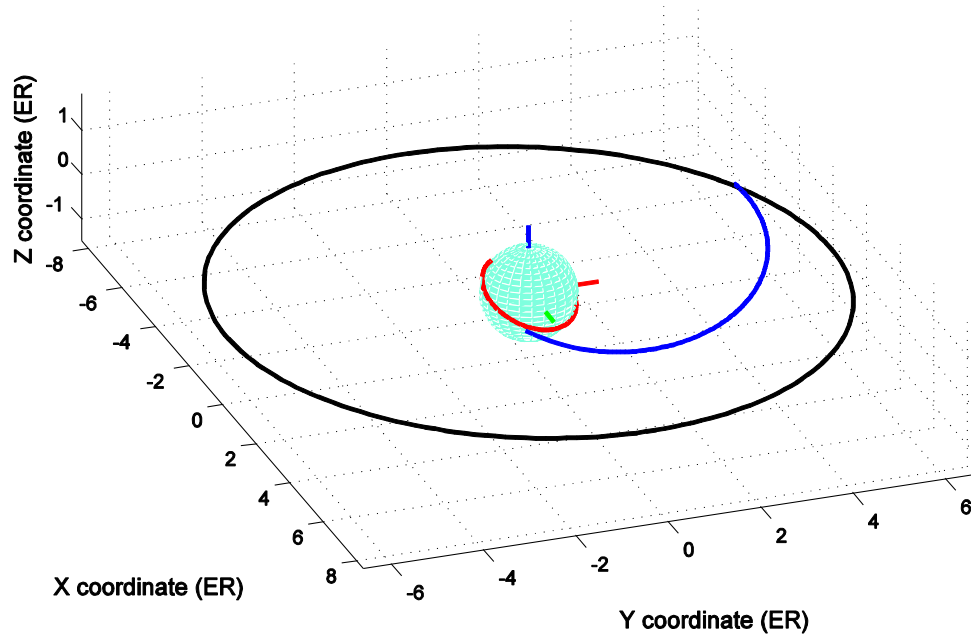


All of these plots were created using the Grapher scientific plotting program (www.goldensoftware.com) and the contents of the simulation summary data file described in Appendix A.

The `twoburn_ocs` computer program will also create three output files named `orbit1.csv`, `orbit2.csv` and `orbit3.csv`. This file contains the Earth-centered inertial position vectors of the park, transfer and final mission orbit. The `twoburn_ocs` software package includes a MATLAB script called `oplot.m` that can be used to create trajectory graphic displays using these 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 three-dimensional geometry of the orbital transfer.

The following is the graphics display for this example. The initial orbit trace is red, the transfer orbit is blue and the final mission orbit is black. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis.

Initial, Transfer and Final Orbits



Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* software can be verified by numerically integrating the orbital equations of motion with the optimal control solution and the initial park orbit conditions determined by the software. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution.

This part of the `twoburn_ocs` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

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

```
=====
verification of optimal control solution
=====
```

```
-----
final mission orbit
-----
```

```
mission elapsed time      05:43:11.886
```

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.421662635019D+05	0.160175963159D-07	0.250000121139D+01	0.127337790391D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.119999993671D+03	0.351099192462D+03	0.383297150142D+01	0.143617514985D+04

rx (km)	ry (km)	rz (km)	rmag (km)
-.234747398791D+05	0.350273491999D+05	0.122951507257D+03	0.421662628346D+05
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.255141829452D+01	-.171038730613D+01	0.133811541581D+00	0.307458380586D+01
right ascension	123.829327888839	degrees	
declination	0.167067507520765	degrees	
final spacecraft mass	4750.78041919988	kilograms	
first delta-v	2529.82005790686	meters/second	
second delta-v	1732.69612356106	meters/second	
total delta-v	4262.51618146792	meters/second	

Creating an initial guess

The software allows the user to input either a delta-v or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. An estimate of the thrust duration can be determined from the following expression:

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

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

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

In these equations

m_i = initial mass

m_f = final mass

m_p = propellant mass

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

I_{sp} = specific impulse

ΔV = impulsive velocity increment

F = thrust

g = acceleration of gravity

For the thrust duration initial guess option, the software requires an initial guess for the thrust duration for each propulsive maneuver. All of these inputs should be in seconds. If the `twoburn_ocs` computer program cannot find a feasible solution, try increasing the guess for thrust duration.

The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting, 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.

The dynamic variables and control variables at each grid point are determined by the *Sparse Optimization Suite* by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 4`. These program options create an initial guess from the numerical integration of the equations programmed in the `oderhs` subroutine. The number and location of the initial collocation nodes are determined from the variable step-size numerical integration.

Problem setup

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

(1) Point functions – initial orbit constraints

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

- 1) constrain semimajor axis, eccentricity and inclination
- 2) constrain all initial orbital elements
- 3) option 2 with unconstrained true longitude

For option 1, the initial orbit inclination is constrained by enforcing

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

where i is the park orbit inclination.

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

$$f = 0 \text{ and } g = 0$$

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

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

is enforced, where e is the user-defined park orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the initial modified equinoctial orbital elements as follows:

$$p_L = p_U = p_i$$

$$f_L = f_U = f_i$$

$$g_L = g_U = g_i$$

$$h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the initial true longitude unbounded.

In optimal control terminology, these derived 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 or termination of the propulsive maneuver. This is simply

$$J = m_f$$

The value of the `maxmin` indicator in the *Sparse Optimization Suite* algorithm tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

(3) Path constraint – unit thrust vector scalar magnitude

For the *variable steering* program option, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows:

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

(4) Point functions – final mission orbit constraints

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

- 4) constrain semimajor axis, eccentricity and inclination
- 5) constrain all final orbital elements
- 6) option 2 with unconstrained true longitude

For option 1, the final orbit inclination is constrained by enforcing

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

where i is the mission orbit inclination.

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

$$f = 0 \quad \text{and} \quad g = 0$$

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

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

is enforced, where e is the user-defined mission orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the user-defined final modified equinoctial orbital elements as follows:

$$p_L = p_U = p_i$$

$$f_L = f_U = f_i$$

$$g_L = g_U = g_i$$

$$h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the final true longitude unbounded.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05m_{sc_i} \leq m_{sc} \leq 1.05m_{sc_i}$$

$$100p_f \leq p \leq 0.8p_i$$

$$-1 \leq f \leq +1$$

$$-1 \leq g \leq +1$$

$$-1 \leq h \leq +1$$

$$-1 \leq k \leq +1$$

where m_{sc_i} is the initial spacecraft mass.

Finally, the three components of the unit thrust vector are constrained as follows:

$$-1.1 \leq u_r \leq +1.1$$

$$-1.1 \leq u_t \leq +1.1$$

$$-1.1 \leq u_n \leq +1.1$$

Technical Discussion

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 summarized as follows:

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

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

position vector

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

velocity vector

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

where

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

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

$$\begin{aligned} \dot{p} &= \frac{dp}{dt} = \frac{2p}{w} \sqrt{\frac{p}{\mu}} \Delta_t \\ \dot{f} &= \frac{df}{dt} = \sqrt{\frac{p}{\mu}} \left[\Delta_r \sin L + [(w+1) \cos L + f] \frac{\Delta_t}{w} - (h \sin L - k \cos L) \frac{g \Delta_n}{w} \right] \\ \dot{g} &= \frac{dg}{dt} = \sqrt{\frac{p}{\mu}} \left[-\Delta_r \cos L + [(w+1) \sin L + g] \frac{\Delta_t}{w} + (h \sin L - k \cos L) \frac{f \Delta_n}{w} \right] \\ \dot{h} &= \frac{dh}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \cos L \\ \dot{k} &= \frac{dk}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \sin L \end{aligned}$$

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

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

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

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

where

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

and

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

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

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

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

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

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:

$$\begin{aligned}\Delta_{J_{2r}} &= -\frac{3\mu J_2 R_e^2}{2r^4} \left[1 - \frac{12(h \sin L - k \cos L)^2}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2t}} &= -\frac{12\mu J_2 R_e^2}{r^4} \left[\frac{(h \sin L - k \cos L)(h \cos L + k \sin L)}{(1 + h^2 + k^2)^2} \right] \\ \Delta_{J_{2n}} &= -\frac{6\mu J_2 R_e^2}{r^4} \left[\frac{(1 - h^2 - k^2)(h \sin L - k \cos L)}{(1 + h^2 + k^2)^2} \right]\end{aligned}$$

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

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

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

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 and \dot{m} is the propellant flow rate.

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

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

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

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the “local horizontal” and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four quadrant operation.

The `twoburn_ocs` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. The following section summarizes the inertial-to/from-modified equinoctial coordinate transformations and the calculation of the inertial unit thrust vector in terms of right ascension and declination angles.

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

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

where

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

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 `twoburn_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

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

where the calculation for right ascension is a four quadrant inverse tangent operation.

Algorithm Resources

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“Optimal Interplanetary Orbit Transfers by Direct Transcription”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

“Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers”, John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, 20-22 August 1990.

“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.

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 the CSV data files produced by the `twoburn_ocs` software.

The simulation summary screen display contains the following information:

mission elapsed time = simulation time since the beginning of the simulation

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 spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

spacecraft mass = current spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v is determined using a cubic spline integration of the thrust acceleration data at each collocation node.

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

time (sec) = mission elapsed time in seconds

time (min) = mission elapsed time in minutes
semimajor axis (km) = semimajor axis in kilometers
eccentricity = orbital eccentricity (non-dimensional)
inclination (deg) = orbital inclination in degrees
argument of perigee = argument of perigee in degrees
raan (deg) = right ascension of the ascending node in degrees
true anomaly (deg) = true anomaly in degrees
period (min) = orbital period in minutes
mass (kg) = spacecraft mass in kilograms
thracc (mps/s) = thrust acceleration in meters/second**2
yaw (deg) = thrust vector yaw angle in degrees
pitch (deg) = thrust vector pitch angle in degrees
rasc (deg) = inertial right ascension in degrees
decl (deg) = inertial declination in degrees
perigee altitude = perigee altitude in kilometers
apogee altitude = apogee altitude in kilometers
ut-radial = radial component of unit thrust vector
ut-tangential = tangential component of unit thrust vector
ut-normal = normal component of unit thrust vector
semi-parameter = orbital semiparameter in kilometers
f equinoctial element = modified equinoctial orbital element
g equinoctial element = modified equinoctial orbital element
h equinoctial element = modified equinoctial orbital element
k equinoctial element = modified equinoctial orbital element
true longitude = true longitude in degrees
rx (km) = x-component of the spacecraft's position vector in kilometers
ry (km) = y-component of the spacecraft's position vector in kilometers
rz (km) = z-component of the spacecraft's position vector in kilometers
fpa (deg) = flight path angle in degrees
rmag (km) = geocentric radius in kilometers
vmag (kps) = velocity in kilometers per second
deltav1 (mps) = first maneuver accumulative delta-v in meters per second
deltav2 (mps) = second maneuver accumulative delta-v in meters per second
dvacc (mps) = total accumulative delta-v in meters per second

APPENDIX B

Example LEO-to-ISS Orbit Transfer

This appendix illustrates the `twoburn_ocs` solution and trajectory graphics for a variable attitude, medium-thrust LEO-to-ISS (International Space Station) orbit transfer. For this example, the impulsive delta-v's were determined using the `hohmann.exe` computer program which can be found at **The Celestial and Orbital Mechanics Website** (www.cdeagle.com). The `hohmann.exe` software computes a Hohmann transfer solution for this problem.

The following is the first part of the input data file for this example.

```
*****
** two maneuver, finite-burn earth-orbit
** trajectory optimization
** program twoburn_ocs
** leo2iss.in - April 16, 2012
*****

initial spacecraft mass (kilograms)
8000.0

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

-----

first propulsive maneuver
-----

thrust magnitude (newtons)
5000.0

specific impulse (seconds)
300.0

initial guess for delta-v (meters/second)
180.0

initial guess for thrust duration (seconds)
170.0

-----

second propulsive maneuver
-----

thrust magnitude (newtons)
10000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
2905.0

initial guess for thrust duration (seconds)
80.0
```

```

-----
coast phase
-----

initial guess for coast duration (minutes)
45.0

lower bound for coast duration (minutes)
20.0

upper bound for coast duration (minutes)
60.0

*****
* INITIAL ORBIT *
*****

semimajor axis (kilometers)
6563.14

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
100.0

true anomaly (degrees)
0.0

*****
initial orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----
1

*****
* FINAL ORBIT *
*****

semimajor axis (kilometers)
6728.14

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
51.6

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
100.0

true anomaly (degrees)
0.0

```

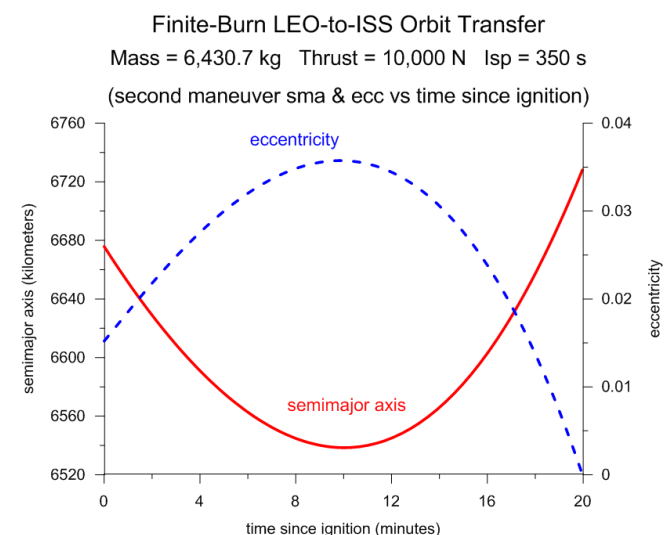
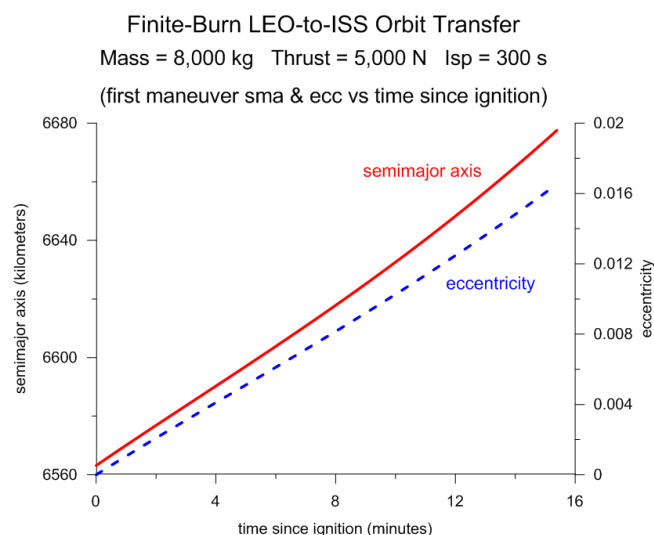
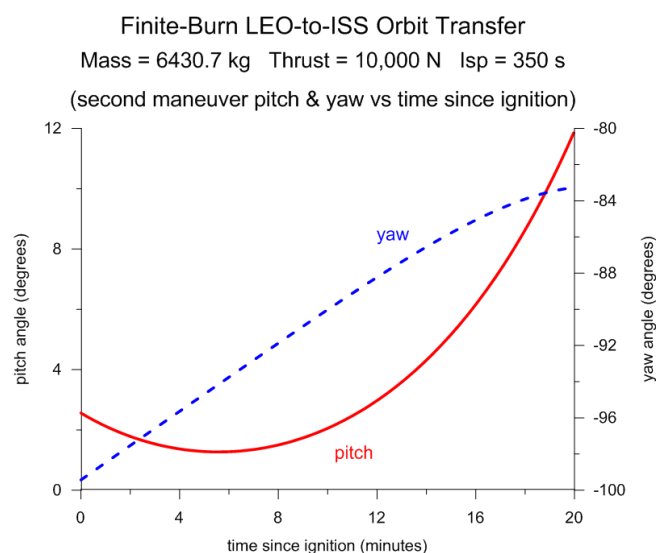
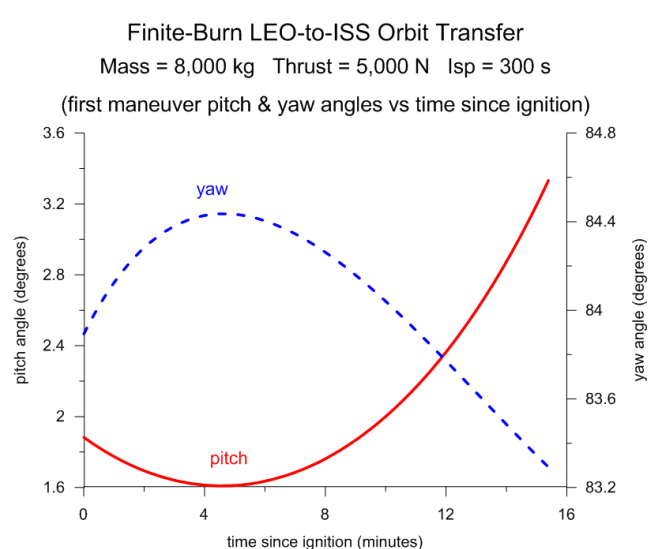
```

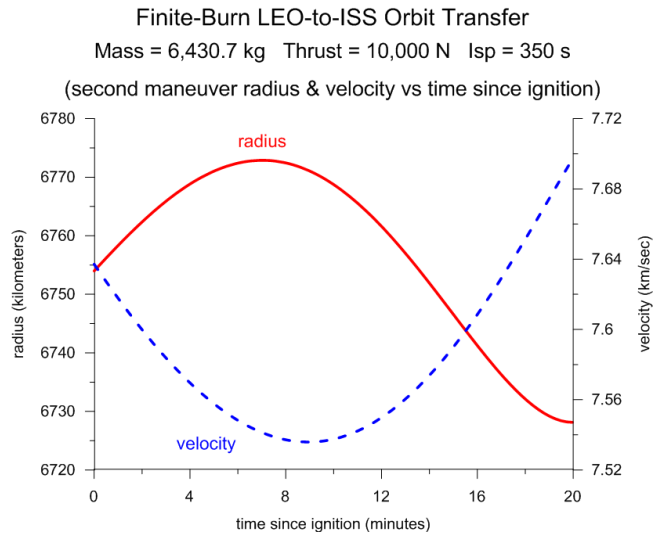
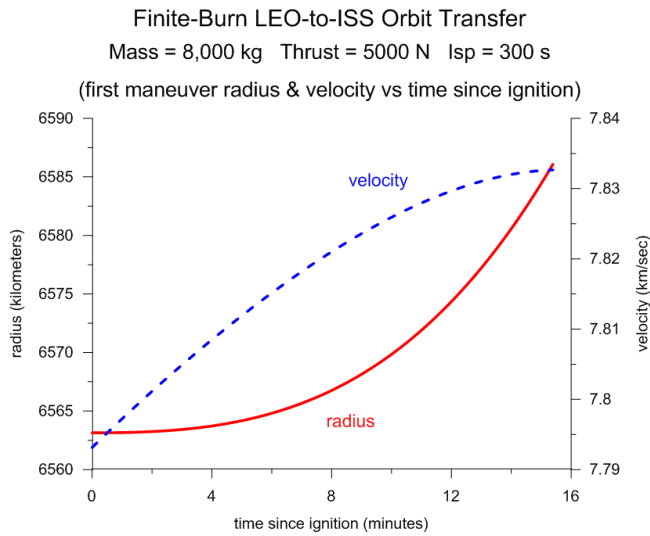
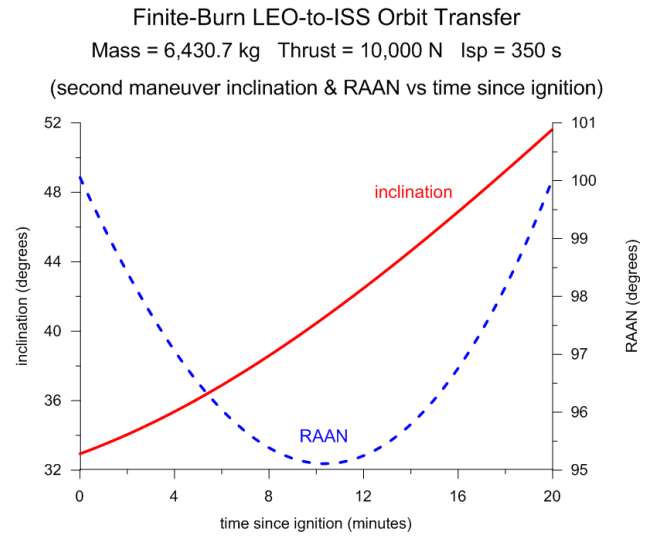
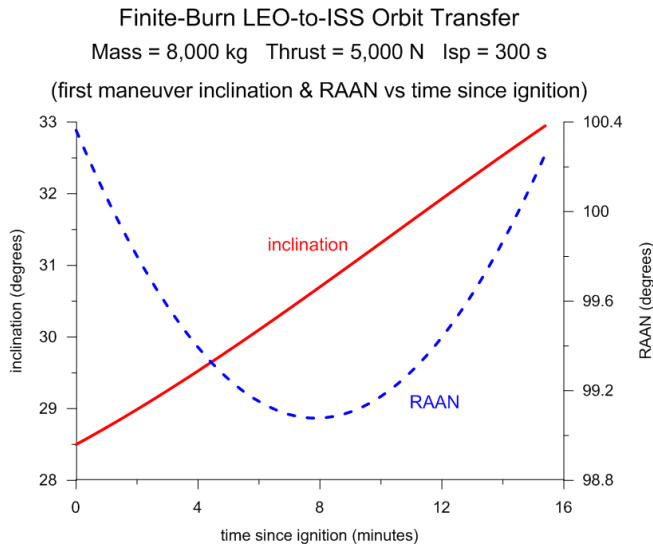
*****
final orbit constraint options
*****
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
-----
3

*****
* type of gravity model *
-----
1 = spherical Earth
2 = oblate gravity model
-----
2

```

The following are plots of the trajectory characteristics for this example.





Here's the main program output and verification for this example.

```

program twoburn_ocs
=====

input file ==> leo2iss.in

numerical integration initial guess

oblate earth gravity model

-----
beginning of first propulsive maneuver
-----

mission elapsed time      00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.656314000000D+04    0.110612553947D-15    0.285000000000D+02    0.000000000000D+00

```

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.100363294121D+03	0.326736552541D+03	0.326736552541D+03	0.881916022527D+02
rx (km)	ry (km)	rz (km)	rmag (km)
0.212477171977D+04	0.596738793323D+04	-.171768246564D+04	0.656314000000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.640214595143D+01	0.317457253705D+01	0.310930995346D+01	0.779315032339D+01

end of first propulsive maneuver

mission elapsed time 00:15:23.359

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.667754855184D+04	0.165328254596D-01	0.329474801412D+02	0.355233433521D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.100255885100D+03	0.345888894857D+02	0.298223230068D+02	0.905076548286D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.372197737672D+04	0.513323822355D+04	0.178135829385D+04	0.658608287403D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.495861875658D+01	-.479160585935D+01	0.371527030388D+01	0.783266366668D+01

spacecraft mass	6430.72714842286	kilograms
propellant mass	1569.27285157714	kilograms
thrust duration	923.358576595192	seconds
	15.3893096099199	minutes
delta-v	642.396161808662	meters/second

beginning of coast phase

mission elapsed time 00:15:23.359

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.667754855184D+04	0.165328254596D-01	0.329474801412D+02	0.355233433521D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.100255885100D+03	0.345888894857D+02	0.298223230068D+02	0.905076548286D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.372197737672D+04	0.513323822355D+04	0.178135829385D+04	0.658608287403D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.495861875657D+01	-.479160585935D+01	0.371527030388D+01	0.783266366668D+01

end of coast phase

mission elapsed time 00:42:01.551

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.667584555215D+04	0.152066565657D-01	0.329380649128D+02	0.355274673206D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.100057403600D+03	0.140887797298D+03	0.136162470504D+03	0.904730332878D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.301488941672D+04	-.548245175517D+04	0.254353639062D+04	0.675399236106D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.544260814774D+01	-.446092094692D+01	-.296721491969D+01	0.763715676965D+01
coast duration	1598.19246759981	seconds	
	26.6365411266635	minutes	
	0.443942352111059	hours	

beginning of second propulsive maneuver

mission elapsed time 00:42:01.551

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.667584555215D+04	0.152066565657D-01	0.329380649128D+02	0.355274673206D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.100057403600D+03	0.140887797298D+03	0.136162470504D+03	0.904730332878D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.301488941672D+04	-.548245175517D+04	0.254353639062D+04	0.675399236106D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.544260814774D+01	-.446092094692D+01	-.296721491969D+01	0.763715676965D+01

end of second propulsive maneuver

mission elapsed time 01:02:59.702

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.672814000000D+04	0.135806524537D-15	0.516000000000D+02	0.000000000000D+00
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.100000000000D+03	0.213965850265D+03	0.213965850265D+03	0.915381771606D+02
rx (km)	ry (km)	rz (km)	rmag (km)
0.326840206482D+04	-.508989721860D+04	-.294590599076D+04	0.672814000000D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.315821923111D+01	0.492352099988D+01	-.500283635428D+01	0.769699807300D+01

propellant mass	3490.78368184216	kilograms
thrust duration	1198.15128277386	seconds
	19.9691880462309	minutes
delta-v	2686.47371008881	meters/second

SIMULATION SUMMARY

```

initial spacecraft mass      8000.000000000000      kilograms
total propellant mass       5060.05653341929        kilograms
final spacecraft mass       2939.94346658071        kilograms

total delta-v              3328.86987189747        meters/second
total thrust duration      2121.50985936905        seconds
                           35.3584976561508        minutes

```

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

----- final mission orbit -----

```
mission elapsed time      01:02:59.702
```

```

      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.672813998828D+04  0.600936604302D-08  0.516000000807D+02  0.204587101726D+02

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.999999996116D+02  0.193507139109D+03  0.213965849282D+03  0.915381769214D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
0.326840199228D+04  -.508989733617D+04  -.294590593112D+04  0.672814002759D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.315821930073D+01  0.492352086106D+01  -.500283638807D+01  0.769699803473D+01

```

```

right ascension      302.705946066975      degrees
declination          -25.9666865187013      degrees

```

```

final spacecraft mass      2939.94346658047        kilograms
first delta-v            642.396149630716        meters/second
second delta-v           2686.47335229412        meters/second
total delta-v            3328.86950192484        meters/second

```

APPENDIX C

Typical Sparse Optimization Suite Configuration File

The `twoburn_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 *Sparse Optimization Suite* user's manual. Please note that the `twoburn_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 `twoburn_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
TOLMSA=0.1490116119384766D-07
TOLMR=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.rsb
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:PTHOTOL=0.100000000000000D+02
INSNLP:RHO1WR=0.100000000000000D+03
INSNLP:IMAXMU=10
INSNLP:MUCALC=3
INSNLP:MXQPIT=1

Relative Motion Between Two Earth Satellites

This document describes a MATLAB script that can be used to design and analyze relative motion trajectories between two Earth satellites in circular orbits. The algorithms in this section are based on the techniques described in the classic paper, “Terminal Guidance System for Satellite Rendezvous”, by W. H. Clohessy and R. S. Wiltshire, *Journal of the Aerospace Sciences*, Vol. 27, 1960.

In the following discussion the passive satellite is called the *target* and the active or maneuvering satellite is called the *chaser*. The state vector of the chaser satellite is defined with respect to a local vertical-local horizontal (LVLH) coordinate system centered at the target satellite.

The relationship between the chaser vehicle state vector $x, y, z, \dot{x}, \dot{y}$ and \dot{z} at any time t to the initial state vector $x_0, y_0, z_0, \dot{x}_0, \dot{y}_0$ and \dot{z}_0 at time t_0 is given by the following state transition matrix for *unperturbed* relative motion:

$$\begin{Bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{bmatrix} 1 & -6(\omega t - \sin \omega t) & 0 & -3t + \frac{4}{\omega} \sin \omega t & -\frac{2}{\omega}(1 - \cos \omega t) & 0 \\ 0 & 4 - 3 \cos \omega t & 0 & \frac{2}{\omega}(1 + \cos \omega t) & \frac{1}{\omega} \sin \omega t & 0 \\ 0 & 0 & \cos \omega t & 0 & 0 & \frac{1}{\omega} \sin \omega t \\ 0 & 6\omega(1 - \cos \omega t) & 0 & -3 + 4 \cos \omega t & 2 \sin \omega t & 0 \\ 0 & 3\omega \sin \omega t & 0 & -2 \sin \omega t & \cos \omega t & 0 \\ 0 & 0 & -\omega \sin \omega t & 0 & 0 & \cos \omega t \end{bmatrix} \begin{Bmatrix} x_0 \\ y_0 \\ z_0 \\ \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \end{Bmatrix}$$

where $\omega = \sqrt{\mu / (r / r_e)^3}$ is the orbital rate of the target's circular orbit with r equal to the radius of the target's orbit and r_e equal to the radius of the Earth. In this equation, μ is the gravitational constant of the Earth.

The x, y and z position components of the chaser satellite as a function of time are given by the following three expressions:

$$x(t) = \left[x_0 - \frac{2\dot{y}_0}{\omega} \right] + [-3\dot{x}_0 - 6\omega y_0]t + \left[2\frac{\dot{y}_0}{\omega} \right] \cos \omega t + \left[4\frac{\dot{x}_0}{\omega} + 6y_0 \right] \sin \omega t$$

$$y(t) = \left[2\frac{\dot{x}_0}{\omega} + 4y_0 \right] + \left[-2\frac{\dot{x}_0}{\omega} - 3y_0 \right] \cos \omega t + \left[\frac{\dot{y}_0}{\omega} \right] \sin \omega t$$

$$z(t) = z_0 \cos \omega t + \frac{\dot{z}_0}{\omega} \sin \omega t$$

grmotion.m – relative motion of two Earth satellites in circular orbits

This MATLAB script calculates and graphically displays the relative Keplerian motion between two satellites in circular Earth orbits. This script provides the following user options:

- user input of initial conditions
- calculate and display synchronous orbit
- calculate and display two impulse rendezvous orbit

This MATLAB script displays two-dimensional motion in the x-y or orbit plane.

User input of initial conditions

This program option allows the user to input the initials conditions of the chaser vehicle relative to the target. The user can graphically display the relative trajectory for either one orbital period or a user-defined duration.

The following is a typical user interaction with this script and this option.

```
graphics display of relative motion

relative motion menu

<1> user input of initial conditions
<2> calculate and display synchronous orbit
<3> calculate and display rendezvous orbit
selection (1, 2 or 3)
? 1

please input the altitude of the target satellite (kilometers)
? 350

please input the initial x-position of the chaser satellite (kilometers)
? 10

please input the initial y-position of the chaser satellite (kilometers)
? 10

please input delta-vx of the chaser satellite (meters/second)
? -3

please input delta-vy of the chaser satellite (meters/second)
? 5
```


Orbital Mechanics with MATLAB

```
user-defined orbit

target altitude      350.0000 kilometers
chaser x distance    10.0000 kilometers
chaser y distance    10.0000 kilometers

vx prior to maneuver -17.1600 meters/second
vy prior to maneuver  0.0255 meters/second

maneuver delta-vx    -3.0000 meters/second
maneuver delta-vy     5.0000 meters/second
total delta-v        5.8310 meters/second
```

```
the orbital period is  91.5382 minutes
```

```
simulation time menu
```

```
<1> user input of simulation time
```

```
<2> simulate for one orbital period
```

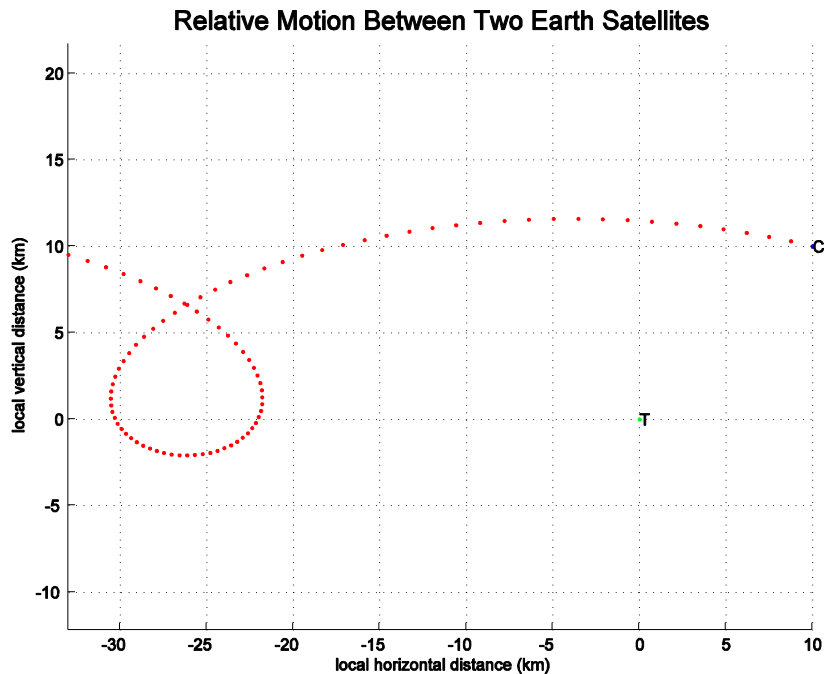
```
selection (1 or 2)
```

```
? 2
```

```
please input the plot step size (minutes)
```

```
? 1
```

The following is the companion graphics display for this example. The orbital motion of the target satellite is to the right. The target satellite is labeled with the letter T and the chaser satellite is labeled with the letter C. The chaser trajectory is displayed at the time interval input by the user.



The relative motion trajectory of the chaser spacecraft is a “drifting” ellipse with its center located at (c, d) where

$$c = x_0 - 2 \frac{\dot{y}_0}{\omega} - (3\dot{x}_0 + 6\omega y_0)t$$

$$d = \frac{2\dot{x}_0}{\omega} + 4y_0$$

The semimajor axis of this dynamic ellipse is given by

$$a = 2 \left\{ \left(-2 \frac{\dot{x}_0}{\omega} - 3y_0 \right)^2 + \left(\frac{\dot{y}_0}{\omega} \right)^2 \right\}$$

and the semiminor axis is equal to $a/2$.

Synchronous orbit

The initial velocity components required for a chaser vehicle to be synchronous or “co-orbital” with a target vehicle located in a user-defined circular orbit are given by

$$\dot{x}_{0_s} = -2\omega y_0$$

$$\dot{y}_{0_s} = 0$$

The initial velocity components for any initial x_0 and y_0 position components are given by the following two expressions:

$$\dot{x}_0 = -\frac{3}{2}\omega y_0$$

$$\dot{y}_0 = \frac{\frac{3}{2}\omega x_0 y_0}{\frac{r_{eq}}{r} + y_0}$$

Therefore, the components of the initial velocity increment for a synchronous orbit are given by

$$\Delta V_x = \dot{x}_{0_s} - \dot{x}_0$$

$$\Delta V_y = \dot{y}_{0_s} - \dot{y}_0$$

The relative motion trajectory is an ellipse with its center located at $(c,0)$ where

$$c = x_0 - 2 \frac{\dot{y}_0}{\omega}$$

The semimajor axis of this ellipse is given by

$$a = 2 \sqrt{y_0^2 + \left[\frac{\dot{y}_0}{\omega} \right]^2}$$

and the semiminor axis is equal to $a/2$.

The following is a typical user interaction and data output for this program option.

```
graphics display of relative motion
```

```
relative motion menu
```

```
<1> user input of initial conditions
```

```
<2> calculate and display synchronous orbit
```

```
<3> calculate and display rendezvous orbit
```

```
selection (1, 2 or 3)
```

```
? 2
```

```
please input the altitude of the target satellite (kilometers)
```

```
? 350
```

```
please input the initial x-position of the chaser satellite (kilometers)
```

```
? 10
```

```
please input the initial y-position of the chaser satellite (kilometers)
```

```
? 10
```

```
synchronous orbit
```

target altitude	350.0000	kilometers
chaser x distance	10.0000	kilometers
chaser y distance	10.0000	kilometers
vx prior to maneuver	-17.1600	meters/second
vy prior to maneuver	0.0255	meters/second
maneuver delta-vx	-5.7200	meters/second
maneuver delta-vy	-0.0255	meters/second
total delta-v	5.7201	meters/second

Orbital Mechanics with MATLAB

the orbital period is 91.5382 minutes

simulation time menu

<1> user input of simulation time

<2> simulate for one orbital period

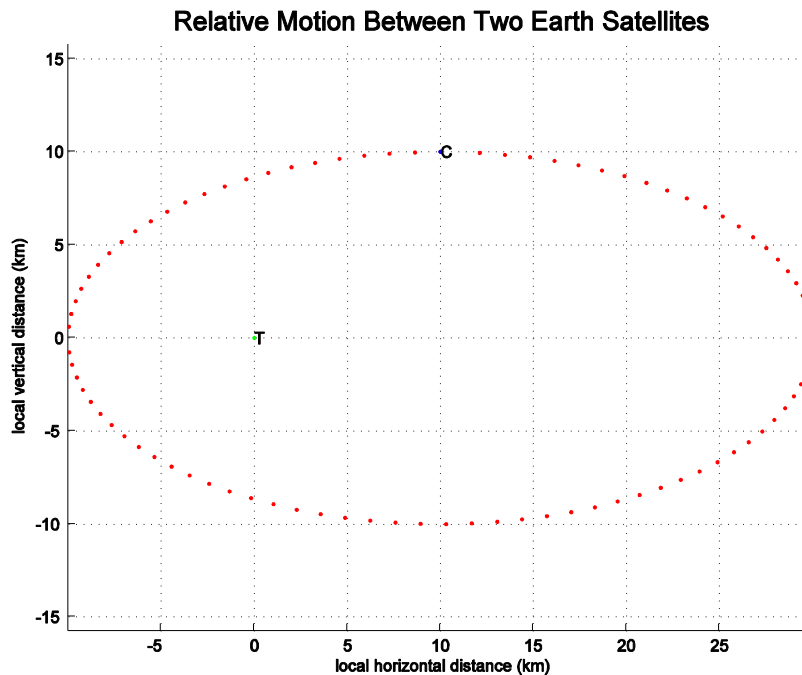
selection (1 or 2)

? 2

please input the plot step size (minutes)

? 1

The following is the companion graphics display of a synchronous orbit. The orbital motion of the target satellite is to the right. The target satellite is labeled with the letter T and the chaser satellite is labeled with the letter C. The chaser trajectory is displayed at the time interval input by the user.



Two impulse rendezvous orbit

Orbital rendezvous is the process of bringing a chaser vehicle from some initial location to a final location with zero relative velocity in a specified transfer time. This type of orbit transfer involves an initial maneuver that starts the transfer and a second maneuver that stops the chaser spacecraft at the final location. This program option of the `grmotion` script calculates the

magnitude and direction of these two *impulsive* maneuvers and graphically displays the transfer trajectory.

The initial velocity components required for a rendezvous orbit are given by

$$\dot{x}_{0_{ipi}} = \frac{14y_0(1 - \cos \omega t_r) - (6y_0\omega t_r - x_0)\sin \omega t_r}{t_r \left[3\sin \omega t_r - \frac{8}{\omega t_r}(1 - \cos \omega t_r) \right]}$$

$$\dot{y}_{0_{ipi}} = \frac{-y_0(3\omega t_r \cos \omega t_r - 4\sin \omega t_r) - 2x_0(1 - \cos \omega t_r)}{t_r \left[3\sin \omega t_r - \frac{8}{\omega t_r}(1 - \cos \omega t_r) \right]}$$

where t_r is the transfer time.

The components of the *terminal phase initiation* ΔV required to start the orbital rendezvous are determined from the following equations

$$\Delta V_x = \dot{x}_{0_{ipi}} - \dot{x}_0$$

$$\Delta V_y = \dot{y}_{0_{ipi}} - \dot{y}_0$$

where \dot{x}_0 and \dot{y}_0 are the x and y velocity components of the chaser vehicle prior to this impulsive maneuver.

The components of the ΔV required to brake the vehicle at the target are given by

$$\dot{x}_b(t_r) = (-3\dot{x}_{ipi} - 6\omega y_0) + (-2\dot{y}_{ipi})\sin \omega t_r + (4\dot{x}_{ipi} + 6\omega y_0)\cos \omega t_r$$

$$\dot{y}_b(t_r) = (2\dot{x}_{ipi} + 3y_0\omega)\sin \omega t_r + (\dot{y}_{ipi})\cos \omega t_r$$

The following is a typical user interaction with this MATLAB script and this program option.

```
relative motion menu

<1> user input of initial conditions

<2> calculate and display synchronous orbit

<3> calculate and display rendezvous orbit

selection (1, 2 or 3)

? 3

please input the altitude of the target satellite (kilometers)
? 300
```

Orbital Mechanics with MATLAB

please input the initial x-position of the chaser satellite (kilometers)
? 50

please input the initial y-position of the chaser satellite (kilometers)
? -100

please input the rendezvous time (minutes)
? 120

rendezvous orbit

target altitude	300.0000	kilometers
chaser x distance	50.0000	kilometers
chaser y distance	-100.0000	kilometers

time to rendezvous	120.0000	minutes
--------------------	----------	---------

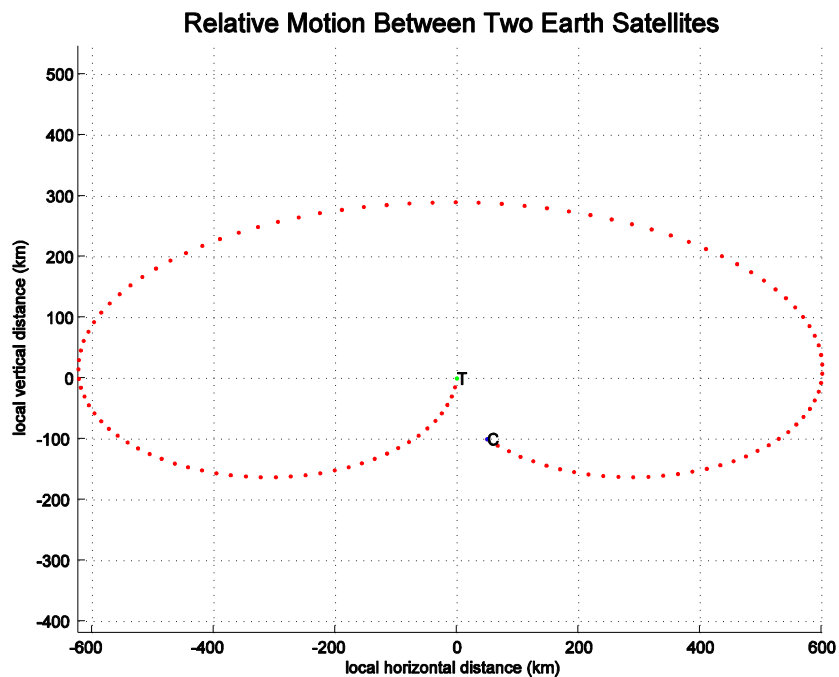
vx prior to tpi	173.5309	meters/second
vy prior to tpi	-1.3190	meters/second

tpi delta-vx	94.6752	meters/second
tpi delta-vy	-179.0340	meters/second
tpi delta-v	202.5255	meters/second

braking delta-vx	36.8316	meters/second
braking delta-vy	250.9074	meters/second
braking delta-v	253.5964	meters/second

total delta-v	456.1219	meters/second
---------------	----------	---------------

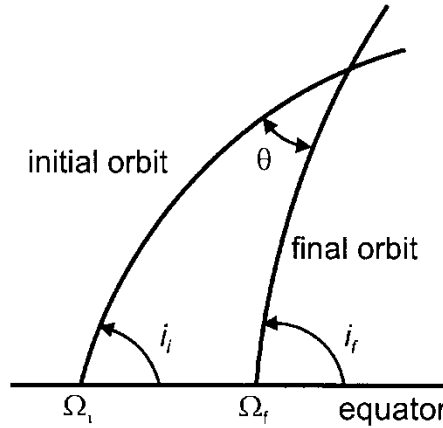
please input the plot step size (minutes)
? 1



Circular Orbit Plane Change

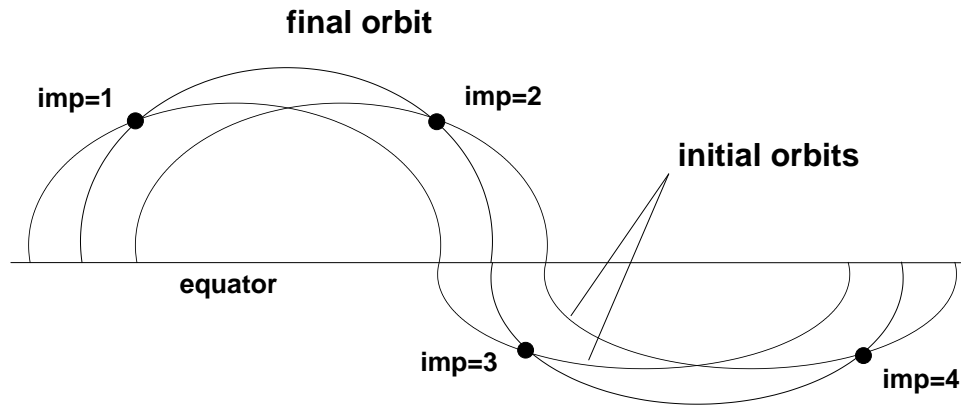
This document presents the geometry and equations associated with the single impulse maneuver that modifies the inclination and/or right ascension of the ascending node (RAAN) of circular orbits. It also describes a MATLAB script that solves this classic astrodynamic problem.

The following diagram illustrates the geometry of this type of orbital maneuver.



In this picture the orbital inclinations of the initial and final orbits are i_i and i_f , respectively. The RAAN of the initial orbit is Ω_i and Ω_f is the RAAN of the final orbit. The right ascension of the ascending node of an orbit is measured from the inertial x-axis along the equator in the direction of the Earth's rotation. From spherical trigonometry relationships, θ is the angle between the two orbit planes.

The next diagram illustrates the possible points of intersection. From this ground track schematic we can see that there are two pairs of orbit intersections on both the initial and final orbits which depend on the relative RAAN between these two orbits.



The total plane change angle due to the modification of inclination and RAAN can be expressed as

$$\theta = \cos^{-1} \left[\sin i_i \sin i_f \cos(\Omega_f - \Omega_i) + \cos i_i \cos i_f \right]$$

We can define an index imp that depends on the *sign* of the RAAN change $\Delta\Omega = \Omega_f - \Omega_i$ as follows

If $\Delta\Omega > 0$ then $imp = 1$ and 3

or

If $\Delta\Omega < 0$ then $imp = 2$ and 4.

It is convenient to define the location of impulses by their argument of latitude. The argument of latitude is the angle from the ascending node, measured along the orbital plane, to the point of interest. The argument of latitude is equal to the sum of the argument of perigee and true anomaly. Since for circular orbits there is no argument of perigee, the argument of latitude and true anomaly are identical.

The two possible arguments of latitude on the initial orbit depend on the values of imp as follows

$$u_i = \text{integer}(imp/2)\pi - (-1)^{imp} u$$

where u is the impulse argument of latitude on the initial orbit given by

$$u = \cos^{-1} \left(\frac{\cos i_i \sin i_f \cos \Delta\Omega}{\sin i_i \sin \theta} \right)$$

We can determine the argument of latitude of an impulse on the final orbit by forming the unit position vectors from the ascending node to the impulse. The argument of latitude of the first opportunity on the final orbit is given by

$$u = \cos^{-1} (\mathbf{U}_1 \bullet \mathbf{U}_2)$$

where \mathbf{U}_1 is the unit position vector of the impulse on the initial orbit and \mathbf{U}_2 is the unit position vector to the ascending node of the final orbit. The argument of latitude of the second impulse opportunity on the final orbit is equal to 180 degrees plus this value.

The maneuver ΔV vector is given by the vector difference between the velocity vectors of the initial and final orbits as follows

$$\Delta \mathbf{V} = \mathbf{V}_f - \mathbf{V}_i$$

These velocity vectors are evaluated at the points of orbital intersection. The scalar magnitude of the ΔV is determined from the components of this vector according to

$$\Delta V = \sqrt{\Delta V_x^2 + \Delta V_y^2 + \Delta V_z^2}$$

For the case where there is no RAAN change, the two impulse locations occur at the common ascending and descending nodes of both the initial and final orbits. The arguments of latitude of these two orbital points are 0 and 180 degrees, respectively.

The required ΔV can also be determined using vector manipulation. Unit vectors normal to each orbit plane can be defined as follows

$$\mathbf{n}_i = \begin{bmatrix} \sin i_i \sin \Omega_i \\ -\sin i_i \cos \Omega_i \\ \cos i_i \end{bmatrix} \quad \mathbf{n}_f = \begin{bmatrix} \sin i_f \sin \Omega_f \\ -\sin i_f \cos \Omega_f \\ \cos i_f \end{bmatrix}$$

A unit vector along the intersection of the initial and final orbit planes is given by

$$\mathbf{m} = \frac{\mathbf{n}_f \times \mathbf{n}_i}{|\mathbf{n}_f \times \mathbf{n}_i|}$$

The velocity vector prior to the maneuver is calculated from

$$\mathbf{V}_i = \frac{\mathbf{n}_i \times \mathbf{m}}{|\mathbf{n}_i \times \mathbf{m}|} V_{lc}$$

The velocity vector after the maneuver is given by

$$\mathbf{V}_f = \frac{\mathbf{n}_f \times \mathbf{m}}{|\mathbf{n}_f \times \mathbf{m}|} V_{lc}$$

where $V_{lc} = \sqrt{\mu/r}$ is the local circular velocity at the maneuver altitude. In this equation, μ is the gravitational constant of the Earth and r is the geocentric radius of the circular orbit.

Finally, the maneuver $\Delta \mathbf{V}$ vector is determined from

$$\Delta \mathbf{V} = \mathbf{V}_f - \mathbf{V}_i$$

The equations described here have been implemented in an interactive MATLAB script called `maneuver1.m`. This script will prompt you for the altitude, inclination and RAAN of both the initial and final orbits. A typical user interaction with this script along with the output is as follows.

```

program maneuver1

< one impulse transfer between circular orbits >

initial orbit

please input the circular orbit altitude (kilometers)
(altitude > 0)
? 300

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

```

```

final orbit

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 30

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 120

solution # 1

initial orbit true anomaly      90.1081 degrees
final orbit true anomaly      72.6145 degrees

delta-V required                1326.0778 meters/second

pitch angle                    0.0000 degrees
yaw angle                      94.9233 degrees

solution # 2

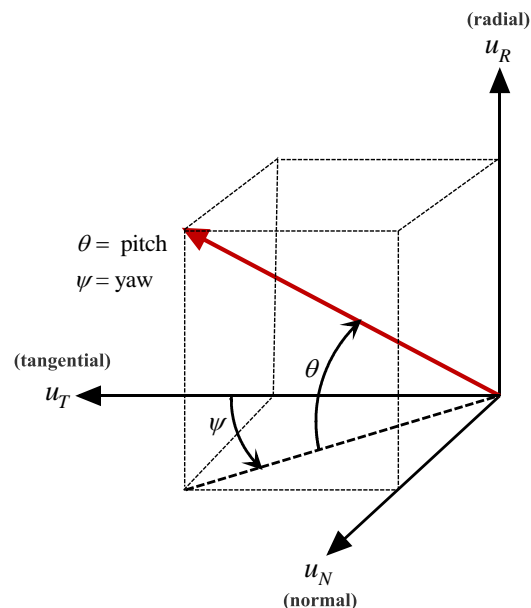
initial orbit true anomaly      270.1081 degrees
final orbit true anomaly      252.6145 degrees

delta-V required                1326.0778 meters/second

pitch angle                    -0.0000 degrees
yaw angle                      -94.9233 degrees

```

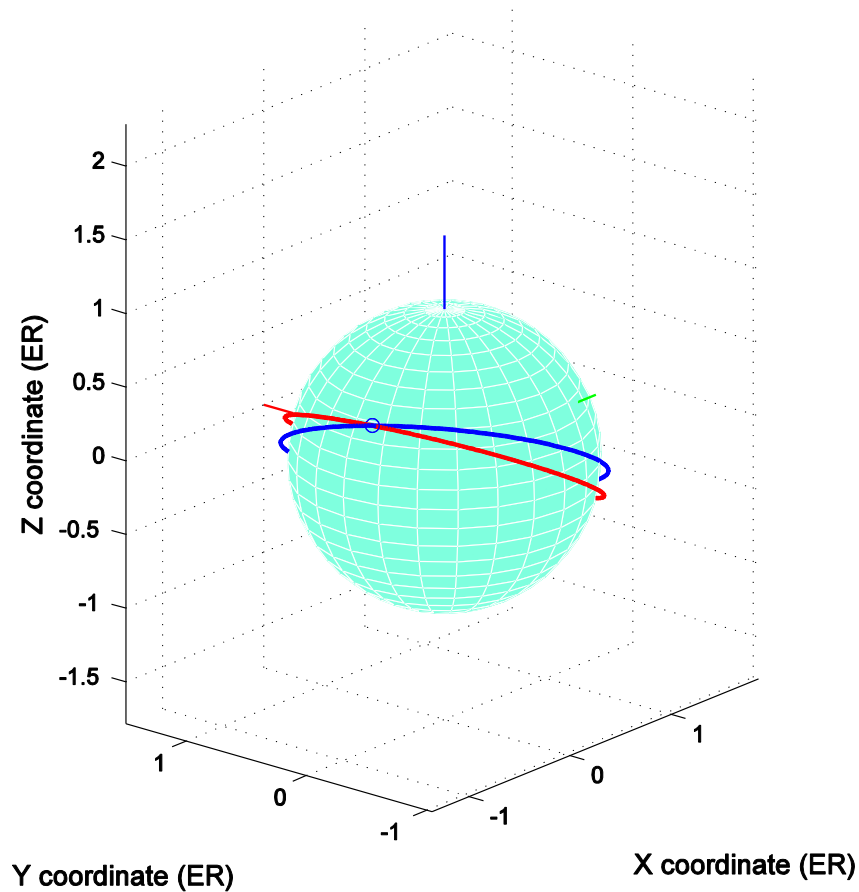
The pitch and yaw angles for each impulsive maneuver are computed and displayed in the local-vertical-local horizontal (LVLH; also called the radial-tangential-normal RTN) coordinate system. The following diagram illustrates the geometry of the pitch and yaw angles in this system. In this figure, the radial direction is along the geocentric radius vector directed away from the Earth, the tangential direction is tangent to the orbit in the direction of the orbital motion, and the normal direction is along the angular momentum vector of the orbit. The pitch angle is positive above the local horizontal plane formed by the tangential and normal directions, and the yaw angle is positive in the direction of the angular momentum vector which is perpendicular to the orbit plane.



The `maneuver1` script will also create a graphics display of the initial and final orbits for each solution. The following is one of the graphic images for this example. The initial orbit trace is red and the final mission orbit is blue. The dimensions are Earth radii (ER) and the plot is labeled with an Earth-centered-inertial (ECI) coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis. The impulse location is marked with a small blue circle.

Trajectory image files are saved to disk in both encapsulated, color Postscript format and MATLAB `fig` format with a file name indicating the solution number. For the first solution, the file names are `plane_change1.eps` and `plane_change1.fig`. The interactive features of MATLAB graphics allow the user to manipulate the `fig` version of the trajectory display. These capabilities allow the user to interactively find the best viewpoint as well as verify basic three-dimensional geometry of the orbital maneuver.

Circular Orbit Plane Change - Solution #1



Targeting with Modified Equinoctial Orbital Elements

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 *direct* modified equinoctial equations exhibit no singularity for zero eccentricity and orbital inclinations equal to 0 and 90 degrees. However, please note that two of the components are singular for an orbital inclination of 180 degrees.

The modified equinoctial elements are defined in terms of the classical orbital elements as follows:

$$p = a(1 - e^2)$$

$$f = e \cos(\omega + \Omega)$$

$$g = e \sin(\omega + \Omega)$$

$$h = \tan(i/2) \cos \Omega$$

$$k = \tan(i/2) \sin \Omega$$

$$L = \theta + \omega + \Omega$$

where

p = semiparameter

a = semimajor axis

e = orbital eccentricity

i = orbital inclination

ω = argument of perigee

Ω = right ascension of the ascending node

θ = true anomaly

L = true longitude

The classical orbital elements can be recovered from the modified equinoctial orbital elements with

semimajor axis

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

orbital eccentricity

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

orbital inclination

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

argument of periapsis

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

$$\sin \omega = \frac{g h - f k}{e \tan(i/2)}$$

$$\cos \omega = \frac{f h + g k}{e \tan(i/2)}$$

right ascension of the ascending node

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

$$\sin \Omega = \frac{k}{\tan(i/2)}$$

$$\cos \Omega = \frac{h}{\tan(i/2)}$$

true anomaly

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

$$\sin \theta = \frac{1}{e}(f \sin L - g \cos L)$$

$$\cos \theta = \frac{1}{e}(f \cos L + g \sin L)$$

In these expressions, an inverse tangent expression of the form $\theta = \tan^{-1}(a, b)$ denotes a four quadrant evaluation where $a = \sin \theta$ and $b = \cos \theta$.

Constraint formulations that enforce both the sine and cosine of a desired orbital element should be used whenever possible. This approach involves a combination of equality and inequality constraints and ensures that the “targeted” orbital element is in the correct quadrant.

To illustrate this technique, here are several examples for different values of argument of perigee and the corresponding mission constraints:

$$0^\circ < \omega < 90^\circ \rightarrow \begin{cases} \sin \omega > 0 \rightarrow g h - f k > 0 \\ f h + g k = e \tan(i/2) \cos \omega \end{cases}$$

$$\omega = 270^\circ \rightarrow \begin{cases} \sin \omega \leq 0 \rightarrow gh - fk \leq 0 \\ \cos \omega = 0 \rightarrow fh + gk = 0 \end{cases}$$

$$\omega = 178^\circ \rightarrow \begin{cases} gh - fk = e \tan(i/2) \sin \omega \\ \cos \omega \leq 0 \rightarrow fh + gk \leq 0 \end{cases}$$

The following is a *sign* table of the sine and cosine for each quadrant.

quadrant	sine	cosine
1	+	+
2	+	−
3	−	−
4	−	+

orbital eccentricity constraint

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

For a circular orbit, $f = g = 0$.

orbital inclination constraint

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

For an equatorial orbit, $h = k = 0$.

argument of perigee constraints

$$gh - fk = e \sin \omega \tan(i/2) \rightarrow \sin \omega = \frac{gh - fk}{e \tan(i/2)}$$

$$fh + gk = e \cos \omega \tan(i/2) \rightarrow \cos \omega = \frac{fh + gk}{e \tan(i/2)}$$

right ascension of the ascending node constraints

$$k = \tan(i/2) \sin \Omega \rightarrow \sin \Omega = \frac{k}{\tan(i/2)}$$

$$h = \tan(i/2) \cos \Omega \rightarrow \cos \Omega = \frac{h}{\tan(i/2)}$$

true anomaly constraints

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

In general,

$$\sin \theta = \frac{1}{e}(f \sin L - g \cos L)$$

$$\cos \theta = \frac{1}{e}(f \cos L + g \sin L)$$

For a circular orbit,

$$\sin \theta = \sin L \cos \Omega - \cos L \sin \Omega$$

$$\cos \theta = \cos L \cos \Omega + \sin L \sin \Omega$$

For a circular, equatorial orbit,

$$\theta = L, \sin \theta = \sin L \text{ and } \cos \theta = \cos L.$$

Targeting Example

For a user-defined semimajor axis, eccentricity and inclination, the set of modified equinoctial constraints are as follows:

$$p = \tilde{p}$$

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

$$\sqrt{h^2 + k^2} = \tan(\tilde{i}/2)$$

where the tilde indicates the value of the user-defined classical orbital element.

References

“On the Equinoctial Orbital Elements”, R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

“A Set of Modified Equinoctial Orbital Elements”, M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

“Equinoctial Orbit Elements: Application to Optimal Transfer Problems”, Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, August 20-22, 1990.