

Program hyper_soc

Optimal Earth Orbit-to-Interplanetary Injection with SOCS

This document is the user's manual for a Fortran computer program called `hyper_soc` that uses the Sparse Optimal Control Software (SOCS) object code library developed by Boeing Phantom Works (www.boeing.com/phantom/socs/) to solve the single-maneuver, finite-burn Earth orbit-to-interplanetary injection trajectory optimization problem. The software attempts to maximize the final spacecraft mass. Since this simulation involves a single propulsive maneuver, this is equivalent to minimizing the required propellant mass.

The important features of this scientific simulation are as follows:

- single, continuous thrust propulsive maneuver
- valid for circular and elliptical park orbits
- fixed or variable attitude steering options
- constant propulsive thrust magnitude
- modified equinoctial equations of motion with non-spherical gravity
- user-specified final coast duration

SOCS is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in SOCS can be found in the book, *Practical Methods for Optimal Control Using Nonlinear Programming* by John. T. Betts, SIAM, 2001.

The `hyper_soc` software consists of Fortran routines that perform the following tasks:

- main program that sets algorithm control parameters and calls the SOCS transcription/optimal control subroutine
- define problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- evaluate the *right-hand-side* differential equations
- define and compute any point and path constraints
- display the optimal solution results

SOCS will use this information to *automatically* transcribe the user's problem and perform the optimization using a sparse nonlinear programming method. The software allows the user to select the type of collocation method and other important algorithm control parameters.

Typical input file

The `hyper_soc`s 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. 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 change the number of lines. 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.

All program inputs and outputs are in an Earth-centered-inertial (ECI) coordinate system. Park orbit angular elements and hyperbolic targets can be specified in either true-of-date, mean-of-date or EME2000 coordinate frame orientation. However, they must be consistent.

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.

```
*****
** departure hyperbola trajectory optimization with SOCS
** single phase, continous-thrust maneuver w/final coast
** variable attitude steering
** hyper3.in - May 30, 2006
*****
```

The first input defines the type of steering to use during the solution process.

```
type of steering
1 = fixed attitude
2 = variable attitude
-----
2
```

The next three inputs define the initial mass, thrust magnitude and specific impulse of the propulsion system, respectively. Please note the proper units for each input.

```
initial spacecraft mass (kilograms)
4000.0

thrust magnitude (newtons)
19840.0

specific impulse (seconds)
450.0
```

The next input defines the initial guess for the maneuver duration, in seconds.

```
initial guess for thrust duration (seconds)
550.0
```

The next two inputs define the lower and upper bounds for the thrust duration.

```
lower bound for thrust duration (seconds)
1.0

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

The next six inputs define the classical orbital elements of the initial park orbit.

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

semimajor axis (kilometers)
6563.34d0

orbital eccentricity (non-dimensional)
0.015

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
90.0

right ascension of the ascending node (degrees)
0.0d0

true anomaly (degrees)
145.0
```

The next three inputs define the characteristics of the departure hyperbola. These inputs are the specific orbital energy (C3), and the right ascension (RLA) and declination (DLA) of the outgoing asymptote, respectively. Please note the proper units for each coordinate.

```
*****
* LAUNCH HYPERBOLA *
*****

specific orbital energy (C3; [km/sec]**2)
8.788564d0

right ascension of outgoing asymptote (RLA; degrees)
349.68004d0

declination of outgoing asymptote (DLA; degrees)
-6.666253d0
```

This next numeric input allows the user to specify a final fixed-duration orbit coast. For a simulation without a final coast, set this value to zero.

```
final coast duration (seconds)
100.0d0
```

This integer input specifies the 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 = oblate Earth
-----
2
```

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 SOCS. 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 = SOCS-defined nodes
2 = user-defined nodes
3 = user-defined step size
-----
1
```

For options 2 or 3, this next input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
25
```

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

```
name of solution output file
hyper1.csv
```

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

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

discretization/collocation method
-----
1 = trapezoidal
2 = separated Hermite-Simpson
3 = compressed Hermite-Simpson
4 = Runge-Kutta 4-stage
-----
1
```

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

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

The next input specifies the relative error in the solution of the differential equations of orbital motion.

```
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 allowed during the simulation.

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

The next input is an integer that specifies 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 SOCS 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 SOCS optimal control algorithm is controlled with the following integer input.

```
*****  
optimal control output  
-----  
1 = none  
2 = terse  
3 = standard  
4 = interpretive  
-----  
1
```

The level of output from the SOCS differential equations algorithm is controlled with the following integer input.

```
*****  
differential equation output  
-----  
1 = none  
2 = terse  
3 = standard  
4 = interpretive  
5 = diagnostic  
-----  
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the SOCS user's manual. To ignore this special output control, input the simple character string no.

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

SOCS solution

The following is the SOCS solution for this example. This simulation used an oblate earth gravity model, variable attitude steering, and a final 100 second coast. The solution output includes the orbital characteristics at the beginning and end of the propulsive maneuver. Please see Appendix B for additional information about the contents of this data display.

```
program hyper_soc
```

```
oblate earth gravity model
```

```
variable attitude steering
```

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

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.656334000000D+04	0.150000000010D-01	0.285000000000D+02	0.195006115362D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.213742357901D+01	0.340195924218D+03	0.175202039580D+03	0.881956270344D+02
rx (km)	ry (km)	rz (km)	rmag (km)
-.646112459283D+04	0.234812665728D+03	0.258243931564D+03	0.647054541135D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.362933336654D+00	-.694302353479D+01	-.375978186858D+01	0.790400253738D+01

We can see how the software has modified the angular elements of the park orbit.

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

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.453027575649D+05	0.114402243113D+01	0.284832207251D+02	0.194946917715D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.212549933374D+01	0.211148072764D+02	0.216061724992D+03	0.000000000000D+00
rx (km)	ry (km)	rz (km)	rmag (km)
-.533674180350D+04	-.370176985041D+04	-.189971873331D+04	0.676704103493D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.504708201625D+01	-.879765413739D+01	-.487168094366D+01	0.112518900820D+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. The delta-v is computed using a cubic spline which is evaluated at each collocation point.

final mass	1764.41944167265	kilograms
propellant mass	2235.58055832735	kilograms
thrust duration	497.258076463921	seconds
delta-v	3611.91414438766	meters/second

end of final coast

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.453544515350D+05	0.114386209641D+01	0.284806735313D+02	0.194956413278D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.212090855372D+01	0.301198678490D+02	0.225076281127D+03	0.000000000000D+00
rx (km)	ry (km)	rz (km)	rmag (km)
-.479993024677D+04	-.455676840647D+04	-.237406823910D+04	0.703133477285D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.566636394191D+01	-.829462813233D+01	-.461067876204D+01	0.110529134088D+02

This final section of the program output displays the actual hyperbolic conditions computed by the software. From this display we can determine how well SOCS satisfies the user-defined targets.

outgoing hyperbola

right ascension	349.680040000000	degrees
declination	-6.66625300000017	degrees
orbital energy	8.78856400000029	(km/sec)**2

The following is a plot of the behavior of the pitch and flight path angles during the maneuver.

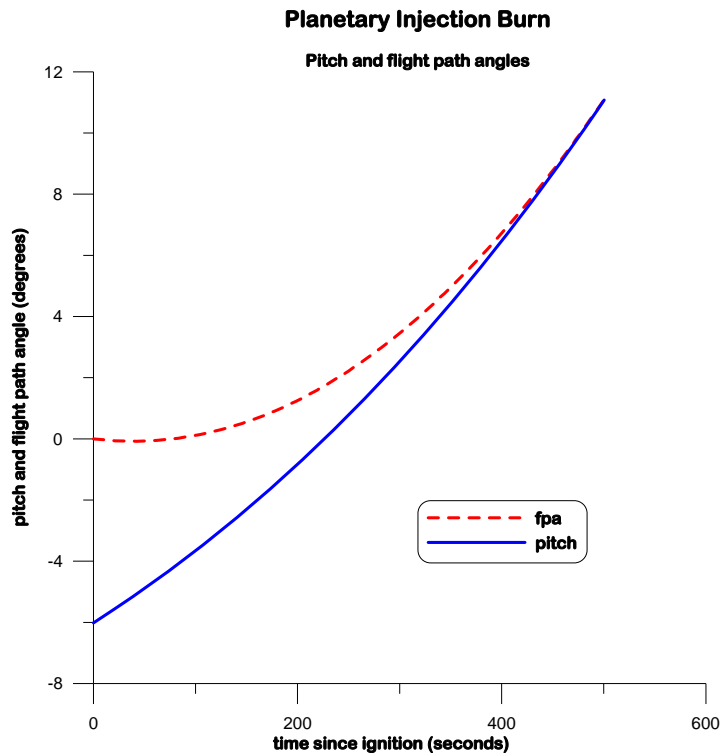


Figure 1 Pitch and flight path angles

The following series of plots illustrate the behavior of the yaw angle, several orbital elements, spacecraft mass, thrust-to-weight and other useful information.

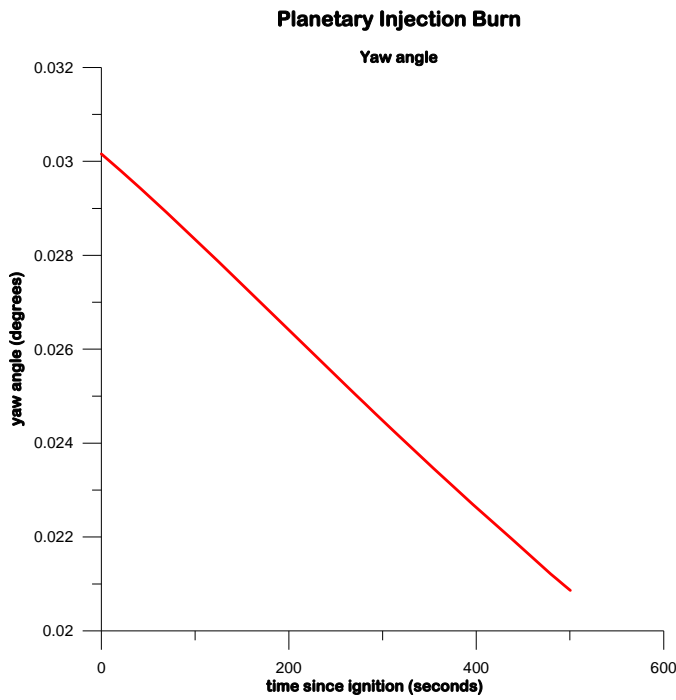


Figure 2 Yaw angle

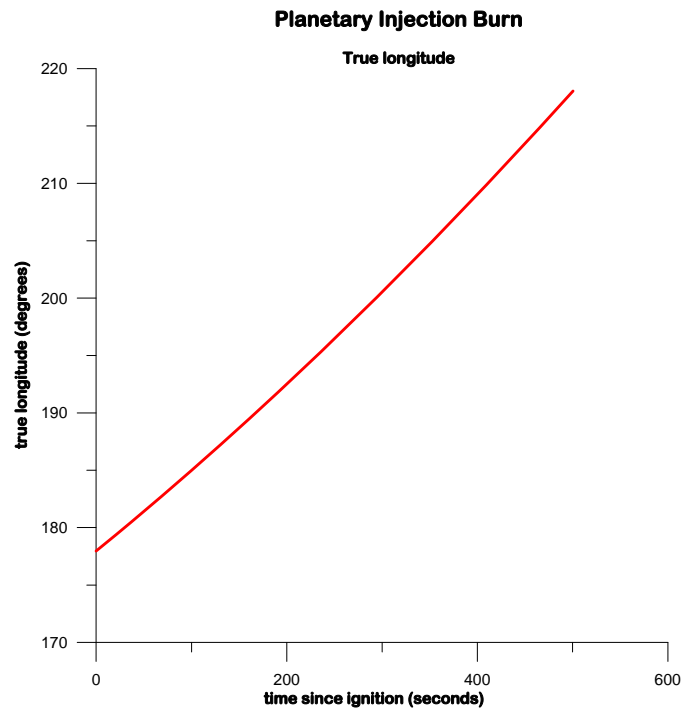


Figure 3 True longitude

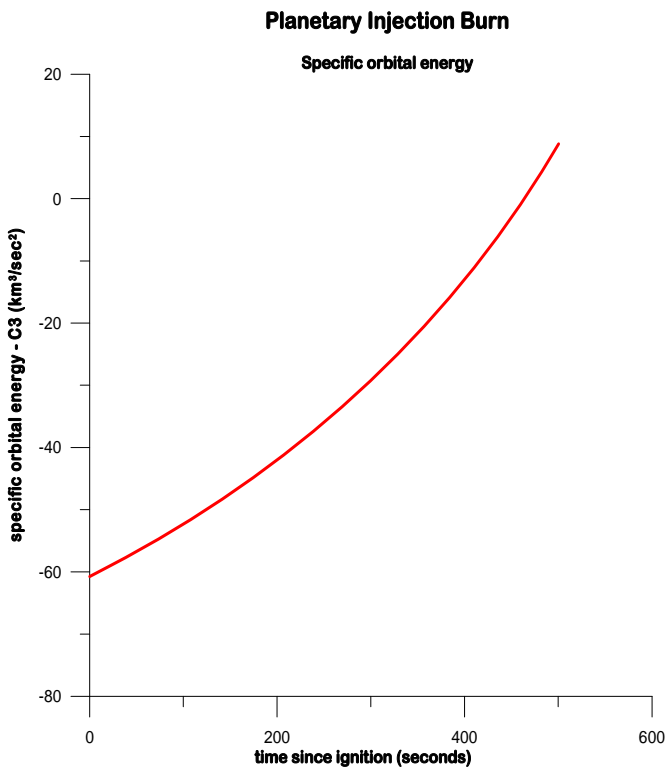


Figure 4 Specific orbital energy

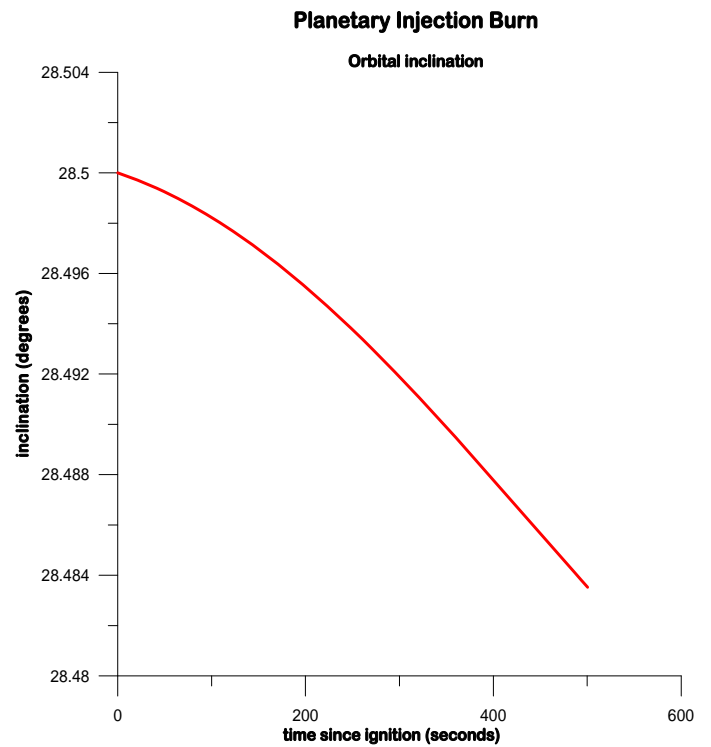


Figure 5 Orbital inclination

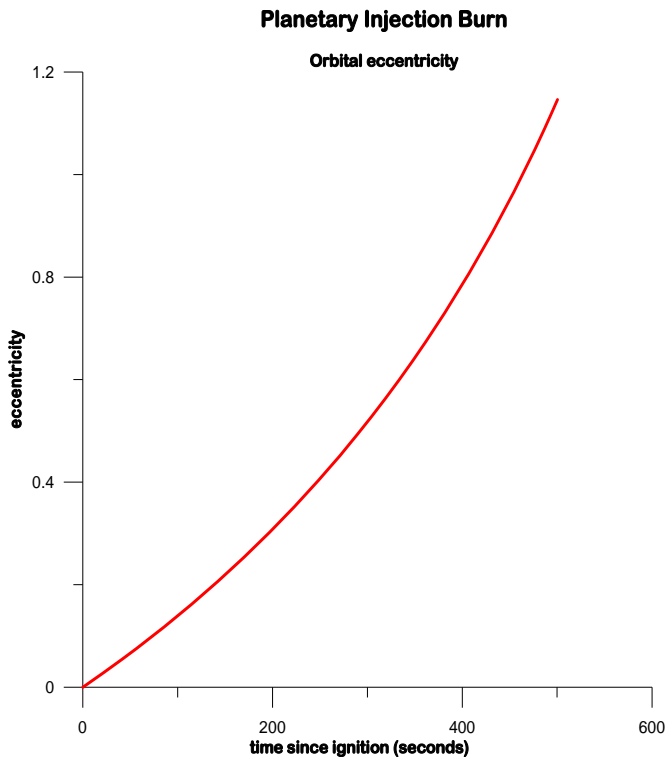


Figure 6 Orbital eccentricity

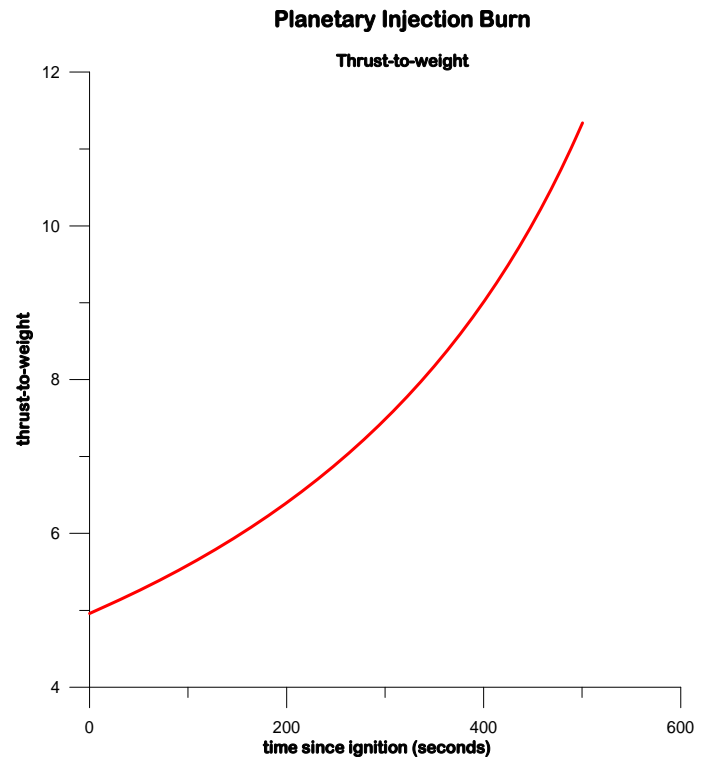


Figure 7 Thrust-to-weight

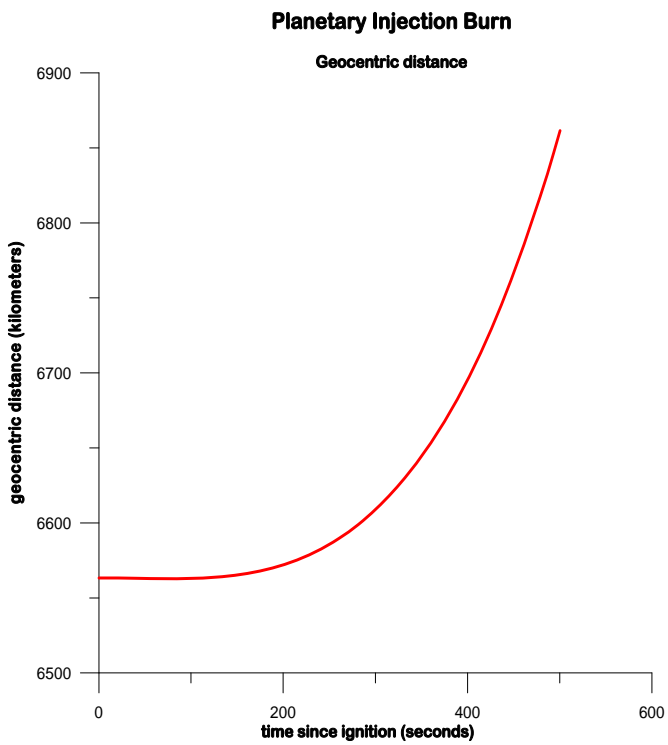


Figure 8 Geocentric distance

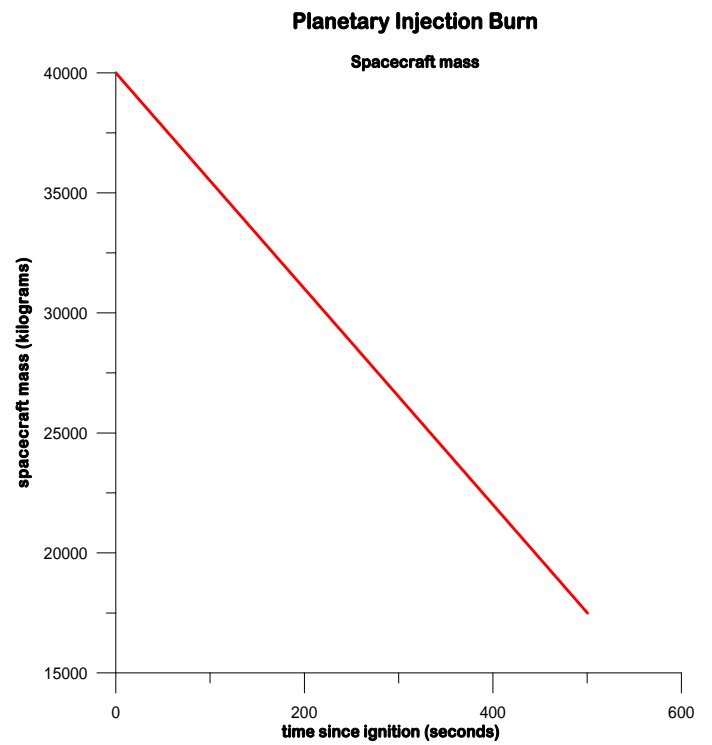
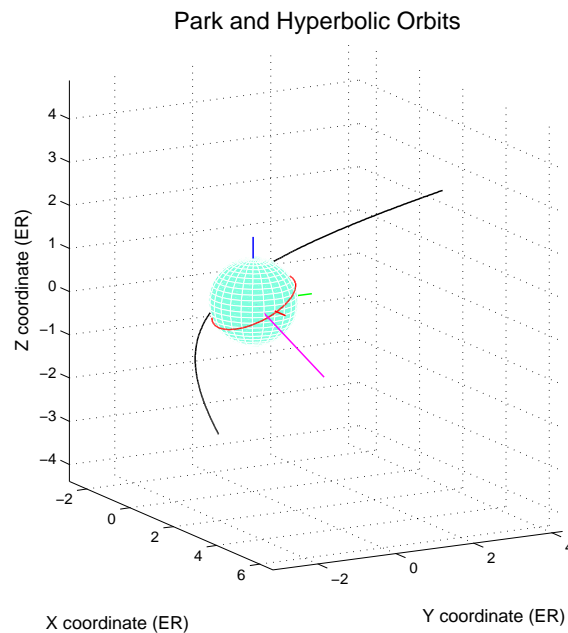


Figure 9 Spacecraft mass

The `hyper_soc`s computer program will also create an output file called `hyper_soc`s.dat. This file contains the Earth-centered inertial position and velocity vectors of the park orbit and departure hyperbola along with the right ascension and declination of the launch asymptote. The software package includes a MATLAB script called `hplot.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 hyperbolic departure.

The next plot is a typical display of the park and hyperbolic orbits for this example. This display is labeled with an Earth centered, inertial coordinate system. The x-axis of this system is red, the y-axis green and the z-axis blue. The outgoing asymptote is colored magenta, the park orbit trace is red, and the hyperbolic trajectory is black.



Impulsive Solution

For comparison purposes, the following is the solution for an impulsive transfer from a circular park orbit with the same semimajor axis and inclination as the `hyper_soc`s example. It was created using the `hyper1.exe` computer program which is available at the Orbital and Celestial Mechanics Web Site (www.cdeagle.com).

```
-----
Interplanetary Injection from a Circular Park Orbit
-----

departure hyperbola characteristics
-----

c3                8.78856400000000    km**2/sec**2
asymptote right ascension  349.680040000000    degrees
```

asymptote declination -6.66625300000000 degrees

orbital elements and state vector of park orbit at injection - opportunity #1

sma (km) eccentricity inclination (deg) argper (deg)
0.6563340000D+04 0.0000000000D+00 0.2850000000D+02 0.0000000000D+00
raan (deg) true anomaly (deg) arglat (deg) period (min)
0.1572493370D+03 0.4320292164D+02 0.4320292164D+02 0.8819562703D+02
rx (km) ry (km) rz (km) rmag (km)
-.5939043529D+04 -.1791276305D+04 0.2143950353D+04 0.6563340000D+04
vx (kps) vy (kps) vz (kps) vmag (kps)
0.2989310236D+01 -.6666955349D+01 0.2710549924D+01 0.7793032157D+01

orbital elements and state vector of hyperbola at injection - opportunity #1

sma (km) eccentricity inclination (deg) argper (deg)
-.4535445153D+05 0.1144712146D+01 0.2850000000D+02 0.4320292164D+02
raan (deg) true anomaly (deg) arglat (deg) period (min)
0.1572493370D+03 0.3600000000D+03 0.4320292164D+02 0.1666666667D+98
rx (km) ry (km) rz (km) rmag (km)
-.5939043529D+04 -.1791276305D+04 0.2143950353D+04 0.6563340000D+04
vx (kps) vy (kps) vz (kps) vmag (kps)
0.4377795746D+01 -.9763646613D+01 0.3969555846D+01 0.1141276760D+02

injection delta-v vector and magnitude - opportunity #1

x-component of delta-v 1388.48550955101 meters/second
y-component of delta-v -3096.69126449904 meters/second
z-component of delta-v 1259.00592265734 meters/second
delta-v magnitude 3619.73544767349 meters/second

orbital elements and state vector of park orbit at injection - opportunity #2

sma (km) eccentricity inclination (deg) argper (deg)
0.6563340000D+04 0.0000000000D+00 0.2850000000D+02 0.0000000000D+00
raan (deg) true anomaly (deg) arglat (deg) period (min)
0.2110742966D+01 0.1950418682D+03 0.1950418682D+03 0.8819562703D+02
rx (km) ry (km) rz (km) rmag (km)
-.6279022342D+04 -.1729369327D+04 -.8127681799D+03 0.6563340000D+04
vx (kps) vy (kps) vz (kps) vmag (kps)
0.2264713418D+01 -.6535013010D+01 -.3591104049D+01 0.7793032157D+01

orbital elements and state vector of hyperbola at injection - opportunity #2

```
-----
      sma (km)      eccentricity      inclination (deg)      argper (deg)
-.4535445153D+05    0.1144712146D+01    0.2850000000D+02    0.1950418682D+03

      raan (deg)    true anomaly (deg)    arglat (deg)    period (min)
0.2110742966D+01    0.5088887490D-13    0.1950418682D+03    0.1666666667D+98

      rx (km)      ry (km)      rz (km)      rmag (km)
-.6279022342D+04    -.1729369327D+04    -.8127681799D+03    0.6563340000D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
0.3316635606D+01    -.9570419225D+01    -.5259112901D+01    0.1141276760D+02
```

injection delta-v vector and magnitude - opportunity #2

```
-----
x-component of delta-v      1051.92218810999      meters/second
y-component of delta-v      -3035.40621511540      meters/second
z-component of delta-v      -1668.00885205830      meters/second
delta-v magnitude           3619.73544767351      meters/second
```

Fixed-attitude Solution

For this same example with fixed attitude steering, the hyper_socS solution is as follows. For this case, the initial guess for the length of the thrust duration was increased to 600 seconds.

program hyper_socS

oblate earth gravity model

fixed attitude steering

beginning of finite burn

```
      sma (km)      eccentricity      inclination (deg)      argper (deg)
0.656334000003D+04    0.150000001539D-01    0.285000000000D+02    0.195002667059D+03

      raan (deg)    true anomaly (deg)    arglat (deg)    period (min)
0.213755461876D+01    0.340321402102D+03    0.175324069161D+03    0.881956270350D+02

      rx (km)      ry (km)      rz (km)      rmag (km)
-.646174150985D+04    0.222691719607D+03    0.251687874560D+03    0.647047461384D+04

      vx (kps)      vy (kps)      vz (kps)      vmag (kps)
-.346347860598D+00    -.694361304680D+01    -.376043683440D+01    0.790408781389D+01
```

end of finite burn

```
      sma (km)      eccentricity      inclination (deg)      argper (deg)
-.453043055236D+05    0.114503053502D+01    0.284834024055D+02    0.195038281161D+03
```

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.212526008163D+01	0.210193002031D+02	0.216057581364D+03	0.000000000000D+00
rx (km)	ry (km)	rz (km)	rmag (km)
-.537289896229D+04	-.372626130241D+04	-.191229742011D+04	0.681248471380D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.503874375200D+01	-.876697473942D+01	-.485490473398D+01	0.112169017405D+02
final mass	1758.66885480063	kilograms	
propellant mass	2241.33114519937	kilograms	
thrust duration	498.537174082281	seconds	
delta-v	3626.32022652314	meters/second	
pitch angle	2.541498133679741E-002	degrees	
yaw angle	2.62081300666892	degrees	

end of final coast			

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.453544515338D+05	0.114487393139D+01	0.284809136979D+02	0.195047574880D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.212078214778D+01	0.299424940318D+02	0.224990068911D+03	0.000000000000D+00
rx (km)	ry (km)	rz (km)	rmag (km)
-.483732217996D+04	-.457852869827D+04	-.238514461925D+04	0.707471029572D+04
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.565054981215D+01	-.827080342622D+01	-.459748433218D+01	0.110214229880D+02

outgoing hyperbola			

right ascension	349.680039999832	degrees	
declination	-6.66625300011631	degrees	
orbital energy	8.78856400023638	(km/sec)**2	
final coast duration	100.000000000000	seconds	

Verification of the SOCS solution

For the variable attitude program option, the optimal control solution determined by SOCS can be verified by numerically integrating the orbital equations of motion with the SOCS-computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the SOCS unit thrust vector solution.

This part of the `hyper_soc`s computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion. The equations of motion used during the propulsive maneuver are coded in subroutine `meeeqms3.for` and for the coast period they are defined in subroutine `meeeqms2.for`.

The following is a display of the final solution computed using this *explicit* numerical integration method. These results are for the variable attitude example.

```

-----
verification of SOCS solution
-----

right ascension      349.680039219799      degrees
declination          -6.66625336537670      degrees
orbital energy       8.78856398762495      (km/sec)**2
final mass           1764.41944167270      kilograms
propellant mass      2235.58055832730      kilograms
delta-v              3611.91367947134      meters/second

```

A comparison of these results with the earlier collocation solution exhibits excellent agreement. The delta-v was computed by including the thrust acceleration in the equations of motion.

Problem setup for SOCS

This section provides additional details about the software implementation. It defines such things as point and path constraints, the performance index and the technique used to create an initial guess for the computer program.

(1) Initial orbit constraints

During the simulation, the software constrains the semimajor axis, eccentricity and inclination of the initial park orbit to the values provided by the user. The other angular orbital elements of the park orbit (argument of perigee, right ascension of the ascending node, and true anomaly) are determined by the `hyper_soc`s computer program.

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. Furthermore, the semiparameter is constrained to the initial value according to

$$p_L = p_U = p_i$$

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

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

Otherwise, for an elliptical park orbit, the single constraint

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

is enforced, where e is the user-defined park orbit eccentricity. In SOCS terminology, these constraints or boundary conditions are called *point functions*.

The lower and upper bounds for the other modified equinoctial elements are as follows:

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

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

The true longitude is completely free.

(2) Performance index – maximize final spacecraft mass

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

$$J = m_f$$

The value of the `maxmin` indicator in SOCS tells the software whether the user is minimizing or maximizing the performance index. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass. The spacecraft mass at the initial time is constrained 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:

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

(4) Point functions – final “scaled” asymptote unit vector

At the final time, the solution should satisfy the following “energy scaled” unit asymptote *targeted minus predicted* vector constraint

$$(C_3 \hat{\mathbf{s}})_T - (C_3 \hat{\mathbf{s}})_p = 0$$

where the user-defined “target” departure asymptote unit vector is given by

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

In this expression, α_∞ is the user-defined RLA and δ_∞ is the user-defined DLA. Appendix D contains a source code listing of the SOCS subroutine that enforces these constraints.

The predicted asymptote unit vector at any trajectory time can be computed from

$$\hat{\mathbf{s}}_P = \frac{1}{1 + C_3 \frac{h^2}{\mu^2}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\} = \frac{1}{1 + C_3 \frac{p}{\mu}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\}$$

where \mathbf{h} and \mathbf{e} are the predicted final angular momentum and orbital eccentricity vectors, respectively. These vectors are computed using the following equations;

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

$$\mathbf{e} = \frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} = \frac{1}{\mu} \left[\left(v^2 - \frac{\mu}{r} \right) \mathbf{r} - (\mathbf{r} \cdot \mathbf{v}) \mathbf{v} \right]$$

The predicted “twice specific” orbital energy (C3) is determined from the final position \mathbf{r} and velocity \mathbf{v} vectors according to

$$C_3 = |\mathbf{v}|^2 - \frac{2\mu}{|\mathbf{r}|}$$

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 propulsive maneuver.

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

$$p \geq 0.8p_i$$

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

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

where m_{sc_i} is the initial spacecraft mass. The elements f and g are unconstrained.

For the variable steering option, the components of the unit thrust vector are bounded as follows:

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

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

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

For the fixed attitude steering option, the pitch and yaw angles are bounded as follows:

$$-90^\circ \leq \theta \leq +90^\circ$$

$$-90^\circ \leq \psi \leq +90^\circ$$

Creating an initial guess for SOCS

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

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 SOCS 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 $\mathbf{u}_T = [0 \ 1 \ 0]^T$.

The dynamic variables and control variables at each grid point are determined by SOCS 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.

Please note that this type of steering method creates a *coplanar* initial guess. It works best when the park orbit and departure hyperbola are nearly coplanar.

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. For this application the equations are well-behaved as the spacecraft transitions from a circular or elliptical park orbit to hyperbolic flight conditions.

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

$$p = a(1 - e^2)$$

$$f = e \cos(\omega + \Omega)$$

$$g = e \sin(\omega + \Omega)$$

$$h = \tan(i/2) \cos \Omega$$

$$k = \tan(i/2) \sin \Omega$$

$$L = \Omega + \omega + \theta$$

where

p = semiparameter

a = semimajor axis

e = orbital eccentricity

i = orbital inclination

ω = argument of periapsis

Ω = right ascension of the ascending node

θ = true anomaly

L = true longitude

The relationship between classical and modified equinoctial orbital elements is:

semimajor axis

$$a = \frac{p}{1 - f^2 - g^2}$$

orbital eccentricity

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

orbital inclination

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

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

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

where

$$\alpha^2 = h^2 - k^2$$

$$s^2 = 1 + h^2 + k^2$$

$$r = \frac{p}{w}$$

$$w = 1 + f \cos L + g \sin L$$

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

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

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

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

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

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

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

where $\Delta_r, \Delta_t, \Delta_n$ are *non-two-body* perturbations in the radial, tangential and normal directions, respectively. For an interplanetary spacecraft, the radial direction is along the heliocentric 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

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

and

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

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

$$\hat{\mathbf{i}}_r = \frac{\mathbf{r}}{|\mathbf{r}|}$$

$$\hat{\mathbf{i}}_n = \frac{\mathbf{r} \times \mathbf{v}}{|\mathbf{r} \times \mathbf{v}|}$$

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

For *unperturbed* two-body motion, $\mathbf{P} = \mathbf{0}$ 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 in the modified equinoctial coordinate system can be expressed as

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

where

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

and

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

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

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

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

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

where

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

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

Finally, the zonal gravity perturbation contribution is

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

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

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

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

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

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

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. *For the variable steering option, the components of this unit vector are 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 SOCS.

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:

$$\begin{aligned} u_{T_r} &= \sin \theta \\ u_{T_t} &= \cos \theta \cos \psi \\ u_{T_n} &= \cos \theta \sin \psi \end{aligned}$$

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

$$\begin{aligned} \theta &= \sin^{-1}(u_{T_r}) \\ \psi &= \tan^{-1}(u_{T_n}, u_{T_t}) \end{aligned}$$

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

For the fixed attitude steering option, the pitch and yaw angles are problem parameters.

References and Bibliography

“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

Compiling and Running the Software

This appendix describes how to compile and run the `hyper_soc`s computer program. This software was created using version 6.4.3 of SOCS and Compaq Visual Fortran.

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

```
df hyper_soc.s.f *.for c:\socs\socs643.lib advapi32.lib
```

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

An input file created by the user can be run from the command line or a simple batch file with a statement similar to the following:

```
hyper_soc hyper1.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 hyper_soc           *
*                                         *
*   Earth orbit-to-interplanetary       *
* trajectory optimization with SOCS     *
*                                         *
*                May 30, 2006           *
*****
please input the name of the simulation definition file
```

The source code that reads the name of an input file included on the command line is

```
c   if present, use command line argument #1 for input file
    call getarg(1, inputfname$, istatus)
```

The source code that creates the file name input prompt is as follows:

```
c   clear screen

    isys = system("cls")

    if (istatus .eq. -1) then
c   *****
c   input filename not on command line
c   request name of simulation definition input file
c   *****

    print *, ' '
```

```

print *, ' '

print *, ' *****'
print *, ' *           Program hyper_socls           *'
print *, ' *                                           *'
print *, ' *   Earth orbit-to-interplanetary   *'
print *, ' * trajectory optimization with SOCS *'
print *, ' *                                           *'
print *, ' *           May 30, 2006           *'
print *, ' *****'

print *, ' '
print *, ' '

print *,
&      'please input the name of the simulation definition file'

      read (*, *) inputfname$
end if

```

If your compiler does not accept input from a command line, you will have to modify this source code for your particular Fortran compiler. You may also choose to eliminate the code that accepts a command line input file. Please note also that your compiler may have a different command to clear the screen.

APPENDIX B

Contents of the Simulation Summary and CSV Files

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

The simulation summary screen display contains the following information:

`sma (km)` = semimajor axis in kilometers
`eccentricity` = orbital eccentricity (non-dimensional)
`inclination (deg)` = orbital inclination in degrees
`argper (deg)` = argument of perigee in degrees
`raan (deg)` = right ascension of the ascending node in degrees
`true anomaly (deg)` = true anomaly in degrees
`arglat (deg)` = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.
`period (min)` = orbital period in minutes. If the orbit is hyperbolic, the period is undefined and a value of 0 is displayed.
`rx (km)` = x-component of the eci position vector in kilometers
`ry (km)` = y-component of the eci position vector in kilometers
`rz (km)` = z-component of the eci position vector in kilometers
`rmag (km)` = geocentric position magnitude in kilometers
`vx (kps)` = x-component of the eci velocity vector in kilometers/second
`vy (kps)` = y-component of the eci velocity vector in kilometers/second
`vz (kps)` = z-component of the eci velocity vector in kilometers/second
`vmag (kps)` = velocity vector scalar magnitude in kilometers/seconds

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

`time (sec)` = time since ignition in seconds
`time (min)` = 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 (m/s2)** = thrust acceleration in meters per second squared
yaw (deg) = thrust vector yaw angle in degrees
pitch (deg) = thrust vector pitch angle in degrees
perigee alt (km) = perigee altitude in kilometers
apogee alt (km) = apogee altitude in kilometers
u-radial = radial component of unit thrust vector
u-tangential = tangential component of unit thrust vector
u-normal = normal component of unit thrust vector
semi-parameter (km) = orbital semiparameter in kilometers
f equinoctial element = $\text{ecc} * \cos(\text{argper} + \text{raan})$
g equinoctial element = $\text{ecc} * \sin(\text{argper} + \text{raan})$
h equinoctial element = $\tan(i/2) * \cos(\text{raan})$
k equinoctial element = $\tan(i/2) * \sin(\text{raan})$
true longitude (deg) = orbital true longitude in degrees
rx (km) = x-component of eci position vector in kilometers
ry (km) = y-component of eci position vector in kilometers
rz (km) = z-component of eci position vector in kilometers
rmag (km) = geocentric radius magnitude in kilometers
vx (kps) = x-component of eci velocity vector in kilometers per second
vy (kps) = y-component of eci velocity vector in kilometers per second
vz (kps) = z-component of eci velocity vector in kilometers per second
vmag (kps) = scalar velocity vector in kilometers per second
fpa (deg) = flight path angle in degrees
c33 (km/sec)2** = twice specific orbital energy
shat(1) = x-component of c3-scaled unit asymptote vector
shat(2) = y-component of c3-scaled unit asymptote vector
shat(3) = z-component of c3-scaled unit asymptote vector
deltav (mps) = accumulative delta-v in meters per second

APPENDIX C

Fortran Functions and Subroutines

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

- hyper_socs.f** - SOCS main executive program
- asympt.for** - c3-scaled unit asymptote vector subroutine
- atan3.for** - four quadrant inverse tangent function
- csint.for** - cubic spline integration of tabular data subroutine
- display.for** - subroutine that displays the simulation summary
- eci2mee.for** - convert eci position and velocity vectors to modified equinoctial orbital elements subroutine
- eci2orb.for** - convert eci position and velocity vectors to classical orbital elements subroutine
- linput.for** - read and echo a line of text from an input file subroutine
- mee2coe.for** - convert modified equinoctial orbital elements to classical orbital elements
- mee2eci.for** - convert modified equinoctial orbital elements to eci position and velocity vectors subroutine
- meeeqms2.for** - modified equinoctial orbital elements equations of motion subroutine - used during the final coast computations
- meeeqms3.for** - modified equinoctial orbital elements equations of motion subroutine - used during the SOCS verification computations
- odeigs.for** - SOCS initial guess subroutine
- odeinp.for** - SOCS simulation input subroutine
- odepf.for** - SOCS point functions subroutine
- odeprt.for** - SOCS print subroutine - creates comma-separated-variable file
- oderhs.for** - SOCS subroutine that evaluates the equations of motion and any algebraic equations
- orb2eci.for** - convert classical orbital elements to eci position and velocity vector subroutine
- readfpn.for** - read and echo floating point number from input file subroutine
- readint.for** - read and echo integer from input file subroutine
- readtext.for** - read and echo text from input file subroutine

rkf78.for - Runge-Fehlberg-Kutta (RKF78) numerical integration subroutine
rkf78cn.for - evaluate RKF78 integration coefficients subroutine
utility.for - number and text manipulation functions and subroutines
uvector.for - unit vector subroutine
vcross.for - vector cross product subroutine
vdot.for - vector dot product subroutine
vecmag.for - vector scalar magnitude function
xmod.for - modulo 2 pi function

APPENDIX D

Example Fortran Subroutines

This appendix contains two Fortran 77 routines that illustrate typical programming conventions used in the `hyper_soc`s software. The first subroutine is the point function routine required by the SOCS software. The second subroutine illustrates how the c3-scaled unit asymptote vector is computed.

```
      subroutine odepf(iphase, iphend, time, y, ny, p, np,
&                    fptf, nf, iferr)
c      initial and final point functions
c      *****
      implicit double precision (a-h, o-z)
      external meeeqms2
      include 'socscopl.inc'
      dimension y(ny), fptf(nf), p(np)
      dimension yi(7), yf(7), reci(3), veci(3), shat(3)
c      initialize error indicator
      iferr = 0
c      unload modified equinoctial elements into local variables
      pmee = y(1)
      fmee = y(2)
      gmee = y(3)
      hmee = y(4)
      xkmee = y(5)
      xlmee = y(6)
      if (iphase .eq. 1 .and. iphend .eq. -1) then
c      initial orbit inclination constraint
      fptf(1) = sqrt(hmee * hmee + xkmee * xkmee)
c      initial orbit eccentricity constraint
      if (ecc1 .gt. 0.0d0) then
      fptf(2) = sqrt(fmee * fmee + gmee * gmee)
```

```

        end if

    end if

    if (iphase .eq. 1 .and. iphend .eq. +1) then
c      *****
c      constrain final c3-scaled outgoing asymptote vector
c      *****

        if (dtcoast .gt. 0.0d0) then
c          propagate orbit and enforce at end of coast

            neq = 7

c          truncation error tolerance

            tetol = 1.0d-12

c          step size guess (seconds)

            h = 10.0d0

c          initial time (seconds)

            t1 = 0.0d0

c          integration time interval (seconds)

            t2 = dtcoast

c          load initial integration vector

            do i = 1, 7
                yi(i) = y(i)
            end do

c          integrate from t1 to t2

            call rkf78 (meeeqlms2, neq, t1, t2, h, tetol, yi, yf)

c          extract modified equinoctial orbital elements

            pmee = yf(1)

            fmee = yf(2)

            gmee = yf(3)

            hmee = yf(4)

            xkmee = yf(5)

            xlmee = yf(6)

            xmass = yf(7)
        end if
    end if

```

```

c      compute eci state vector at end of coast

      call mee2eci(pmee, fmee, gmee, hmee, xkmee, xlmee, reci,
&                veci, smovrp, tani2s, cosl, sinl, wmee, radius,
&                hsmks, ssqrd)

c      radius magnitude (kilometers)

      rmag = vecmag(reci)

c      velocity magnitude (kilometers/second)

      vmag = vecmag(veci)

c      specific orbital energy (km/sec)**2

      c33 = vmag * vmag - 2.0d0 * emu / rmag

      if (c33 .lt. 0.0d0) then
c          -----
c          elliptic orbit
c          -----

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

      else
c          -----
c          hyperbolic orbit
c          -----

          call asympt(emu, reci, veci, shat)

c      compute c3-scaled outgoing asymptote vector

          do i = 1, 3
              fptf(i) = c33 * shat(i)
          end do
      end if

end if

return
end

```

```

      subroutine asympt(cbm, rsc, vsc, shat)

c      c3-scaled outgoing unit asymptote vector

c      input

c      cbm = central body gravitational constant (km**3/sec**2)
c      rsc = spacecraft position vector (kilometers)
c      vsc = spacecraft velocity vector (kilometers/second)

```

```

c      output

c      shat = c3-scaled outgoing unit asymptote vector

c      *****

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

      dimension rsc(3), vsc(3), hv(3), ev(3), hxe(3)

      dimension vxh(3), rhat(3), shat(3)

      zmu = 1.0d0 / cbmu

c      position magnitude (km)

      rmag = vecmag(rsc)

c      velocity magnitude (km/sec)

      vmag = vecmag(vsc)

c      twice specific orbital energy (km/sec)**2

      c33 = vmag * vmag - 2.0d0 * cbmu / rmag

      if (c33 .lt. 0.0d0) then
         pause 'subroutine asympt - orbit is not hyperbolic'
         return
      end if

c      angular momentum vector

      call vcross(rsc, vsc, hv)

      hmag = vecmag(hv)

      call vcross(vsc, hv, vxh)

c      compute unit position and eccentricity vectors

      do i = 1, 3
         rhat(i) = rsc(i) / rmag

         ev(i) = zmu * vxh(i) - rhat(i)
      end do

      fac1 = 1.0d0 / (1.0d0 + c33 * hmag * hmag / cbmu / cbmu)

      fac2 = sqrt(c33) / cbmu

      call vcross(hv, ev, hxe)

c      compute components of c3-scaled unit asymptote vector

      shat(1) = fac1 * (fac2 * hxe(1) - ev(1))

```

```
shat(2) = fac1 * (fac2 * hxe(2) - ev(2))  
shat(3) = fac1 * (fac2 * hxe(3) - ev(3))  
  
return  
end
```