

Program flyby_soc

Interplanetary Gravity-Assist Trajectory Optimization with SOCS

This document is the user's manual for a Fortran computer program called `flyby_soc` that uses the Sparse Optimal Control Software (SOCS) object code library developed by Boeing (www.boeing.com/phantom/socs/) to solve the classic one impulse flyby and two-impulse rendezvous, gravity-assist interplanetary trajectory optimization problems. The software attempts to minimize the launch delta-v, the arrival delta-v or the total mission delta-v. The type of trajectory optimization is specified by the user.

The important features of this scientific simulation are as follows:

- one-impulse, *patched*, *n-body* interplanetary destination flyby trajectory modeling
- two-impulse, *patched*, *n-body* interplanetary *rendezvous* trajectory modeling
- heliocentric, inertial cartesian equations of motion with point-mass planetary perturbations
- elliptical, non-coplanar planetary, asteroid and comet orbits
- user-selected Meeus, SLP96 or DE405 planetary ephemeris model

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 `flyby_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 definition and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- evaluate the *right-hand-side* differential equations
- define and compute any point and path constraints
- display the optimal solution results

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

Typical input file

The `flyby_socs` computer program is “data-driven” by a simple text file created by the user. The contents of this file include such things as initial guesses, the discretization method, and algorithm control parameters. The following is a typical input file used by this computer program. This example is an Earth-to-Mars trajectory with a gravity-assist from Venus. The performance index for this example is the total delta-v for the mission. Please note for a flyby trajectory input file (trajectory type = 1), the only optimization option is 1 = minimize launch delta-v.

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

```
*****
** gravity-assist interplanetary trajectory optimization
** patched-conic heliocentric motion
** Earth-to-Venus-to-Mars - evm.in
** July 20, 2006
*****
```

The first program input is an integer that defines the type of delta-v optimization. Please note that option 4, no optimization simply solves the orbital two-point boundary value problem subject to the user-defined flyby altitude constraint.

```
*****
* simulation type *
*****
1 = minimize launch delta-v
2 = minimize arrival delta-v
3 = minimize total delta-v
4 = no optimization
-----
3
```

*The next input is an integer that specifies the type of planetary ephemeris model to use during the simulation. **Important note:** the Meeus algorithm (option 1) does not include an ephemeris for Pluto.*

```
ephemeris type (1 = Meeus, 2 = SLP96, 3 = DE405)
3
```

The next input is an integer that tells the simulation what type of trajectory to model.

```
trajectory type (1 = flyby at destination, 2 = rendezvous)
2
```

The next three inputs are the user's initial guess for the launch calendar date. Be sure to include all four digits of the calendar year.

```
launch calendar date initial guess (month, day, year)
2,1,2009
```

The software allows the user to specify an initial guess for the launch, flyby and arrival calendar dates, and lower and upper bounds on the actual launch, flyby and arrival calendar dates found during the optimization process. For any guess for launch time t_L and user-defined launch time lower and upper bounds Δt_l and Δt_u , the launch time t is constrained as follows:

$$t_L - \Delta t_l \leq t \leq t_L + \Delta t_u$$

Likewise, for any guess for arrival time t_A and user-defined arrival time bounds Δt_l and Δt_u , the arrival time t is constrained as follows:

$$t_A - \Delta t_l \leq t \leq t_A + \Delta t_u$$

For fixed launch, flyby or arrival times, the lower and upper bounds are set to 0.

The next two inputs are the lower and upper bounds for the launch calendar date search interval. These values should be input in days.

```
launch date search boundary (days)
-10, +10
```

The next three inputs are the user's initial guess for the flyby calendar date. Be sure to include all four digits of the calendar year.

```
flyby calendar date initial guess (month, day, year)
6,1,2009
```

The next two inputs are the lower and upper bounds for the flyby calendar date search interval. These values should be input in days.

```
flyby search boundary (days)
-60, +60
```

The next three inputs are the user's initial guess for the arrival calendar date. Be sure to include all four digits of the calendar year.

```
arrival calendar date initial guess (month, day, year)
1,17,2010
```

The next two inputs are the lower and upper bounds for the arrival calendar date search interval. These values should be input in days.

```
arrival search boundary (days)
-30, +30
```

The next two program inputs are integers that specify the launch and flyby planets.

```

*****
* launch planet *
*****
1 = Mercury
2 = Venus
3 = Earth
4 = Mars
5 = Jupiter
6 = Saturn
7 = Uranus
8 = Neptune
9 = Pluto
-----
3

```

```

*****
* flyby planet *
*****
1 = Mercury
2 = Venus
3 = Earth
4 = Mars
5 = Jupiter
6 = Saturn
7 = Uranus
8 = Neptune
9 = Pluto
-----
2

```

This next input defines the altitude constraint during the gravity assist. This altitude is with respect to a spherical flyby planet.

```

*****
flyby altitude (kilometers)
*****
500.0

```

This integer specifies the arrival planet or asteroid/comet.

```

*****
* arrival celestial body *
*****
1 = Mercury
2 = Venus
3 = Earth
4 = Mars
5 = Jupiter
6 = Saturn
7 = Uranus
8 = Neptune
9 = Pluto
0 = asteroid/comet
-----
4

```

The next series of inputs include the name and classical orbital elements of a comet or asteroid (arrival celestial body = 0). Please note that the angular orbital elements must be specified with respect to a heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system.

```
*****
* asteroid/comet orbital elements *
* (heliocentric, ecliptic J2000) *
*****

asteroid/comet name
Tempel 1

calendar date of perihelion passage (month, day, year)
7, 5.3153, 2005

perihelion distance (au)
1.506167

orbital eccentricity (nd)
0.517491

orbital inclination (degrees)
10.5301

argument of perihelion (degrees)
178.8390

longitude of the ascending node (degrees)
68.9734
```

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 (option 2) or time step size of the data file (option 3).

```
*****
* 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 (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
0.1
```

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
evm.csv
```

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

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

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

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

```
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 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. Please note that option 4 will create lots of information.

```
*****
optimal control output
-----
```

```

1 = none
2 = terse
3 = standard
4 = interpretive
-----
1

```

The level of output from the SOCS differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```

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

```

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

```

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

```

Optimal control solution and trajectory plots

The `flyby_soc`s computer program will create a comma-separated-variable (csv) solution file and a screen display of important trajectory characteristics. The csv file contains the state vector and classical orbital elements of the spacecraft's trajectory. The `flyby_soc`s software will also create a comma-separated-variable file called `planets.csv`. This file contains the heliocentric, ecliptic state vectors of all three celestial bodies. Please see Appendix B for additional details about the information contained on the screen display and in the disk files.

The following is the screen display created by the simulation for this example.

```

program flyby_soc
input file ==> evm.in
DE405 ephemeris
*****
RENDEZVOUS TRAJECTORY
*****
-----
minimize total delta-v
-----

```

LAUNCH CONDITIONS

=====

calendar date 01/26/2009
 ephemeris time 22:53:16
 julian date 2454858.453662539832294

 launch delta-vx 3.598604063989506 km/sec
 launch delta-vy -1.079092619457513E-001 km/sec
 launch delta-vz -4.086591173791428 km/sec

 launch delta-v 5.446266890254780 km/sec

launch hyperbola characteristics
 (Earth mean equator and equinox of J2000)

right ascension 22.986888957453274 degrees
 declination -44.131818516780072 degrees

 launch energy 29.661823039885469 (km/sec)**2

heliocentric orbital elements and state vector of the departure planet
 (Earth mean ecliptic and equinox of J2000)

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1000894403D+01	0.1748959502D-01	0.1517944624D-02	0.1876686433D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.2765660358D+03	0.2277127737D+02	0.2104399206D+03	0.3657470381D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.8866577256D+08	0.1176380074D+09	-.1977248281D+04	0.1473102848D+09
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.2428303599D+02	-.1805013136D+02	-.6937957233D-03	0.3025678568D+02

heliocentric orbital elements and state vector of the spacecraft
 (Earth mean ecliptic and equinox of J2000)

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.8633493747D+00	0.1583748475D+00	0.8469948443D+01	0.3365013379D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.3070007922D+03	0.2035038834D+03	0.1800052213D+03	0.2930072019D+03
rx (km)	ry (km)	rz (km)	rmag (km)
-.8866577257D+08	0.1176380074D+09	-.1977248283D+04	0.1473102848D+09
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.2068443193D+02	-.1815804063D+02	-.4087284970D+01	0.2782563678D+02

FLYBY CONDITIONS
=====

calendar date 06/02/2009
ephemeris time 08:20:50
julian date 2454984.847803846001625

incoming v-infinity 7.461883786409045 km/sec
outgoing v-infinity 7.461883787737950 km/sec
flyby altitude 500.000000098842065 kilometers
maximum turn angle 58.793127753683557 degrees
actual turn angle 56.203294165918123 degrees
true anomaly at infinity 118.101647082959076 degrees

classical orbital elements at periapsis
(planet-centered, Earth ecliptic and equinox of J2000)

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.5834416183D+04	0.2122974398D+01	0.7257850500D+02	0.3896203468D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.2372824876D+03	0.0000000000D+00	0.3896203468D+02	0.0000000000D+00

heliocentric delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)

flyby delta-vx 1.840897052865073 km/sec
flyby delta-vy 5.314099383416037 km/sec
flyby delta-vz -4.217514536698364 km/sec

flyby delta-v 7.029650281723269 km/sec
maximum delta-v 7.326580266411598 km/sec

heliocentric orbital elements of the spacecraft after flyby
(Earth mean ecliptic and equinox of J2000)

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1182404940D+01	0.3907552140D+00	0.2468728910D+01	0.3205434933D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.3483863519D+03	0.3443052036D+03	0.3048486969D+03	0.4696212933D+03

ARRIVAL CONDITIONS
=====

calendar date 01/14/2010
ephemeris time 04:01:28

julian date 2455210.667686440516263

heliocentric ecliptic delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)

arrival delta-vx 3.466486450913191 km/sec
arrival delta-vy 2.288998258409003 km/sec
arrival delta-vz -7.550962814391415E-001 km/sec

arrival delta-v 4.222109867780254 km/sec

arrival energy 17.826211735607398 km**2/sec**2

heliocentric orbital elements and state vector of the spacecraft
(Earth mean ecliptic and equinox of J2000)

sma (au) eccentricity inclination (deg) argper (deg)
0.1523705059D+01 0.9333969369D-01 0.1848903678D+01 0.2865565886D+03

raan (deg) true anomaly (deg) arglat (deg) period (days)
0.4952541956D+02 0.1465545457D+03 0.7311113434D+02 0.6869890122D+03

rx (km) ry (km) rz (km) rmag (km)
-.1320602259D+09 0.2062726145D+09 0.7565010995D+07 0.2450418825D+09

vx (kps) vy (kps) vz (kps) vmag (kps)
-.1948833059D+02 -.1100488982D+02 0.2479563425D+00 0.2238222758D+02

heliocentric orbital elements and state vector of the destination body
(Earth mean ecliptic and equinox of J2000)

sma (au) eccentricity inclination (deg) argper (deg)
0.1523705056D+01 0.9333969473D-01 0.1848903678D+01 0.2865565882D+03

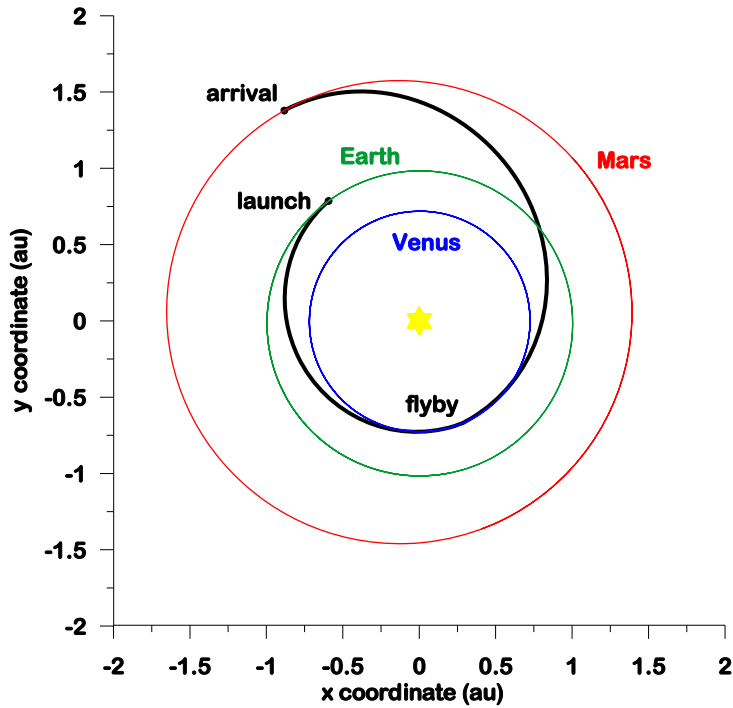
raan (deg) true anomaly (deg) arglat (deg) period (days)
0.4952541956D+02 0.1465545461D+03 0.7311113434D+02 0.6869890105D+03

rx (km) ry (km) rz (km) rmag (km)
-.1320602259D+09 0.2062726144D+09 0.7565010992D+07 0.2450418824D+09

vx (kps) vy (kps) vz (kps) vmag (kps)
-.1948833058D+02 -.1100488981D+02 0.2479563425D+00 0.2238222757D+02

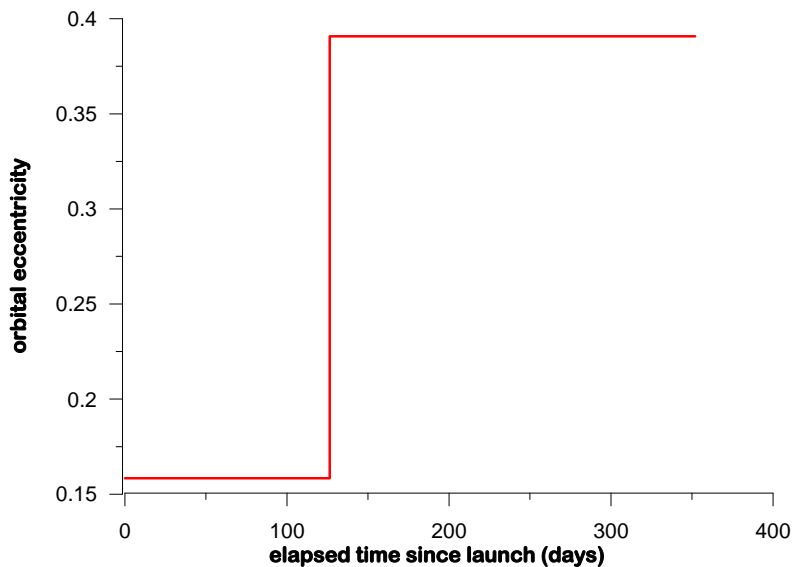
The following is a graphics display of the transfer trajectory for this example. It is a view of the trajectory and planetary orbits from the north pole of the ecliptic looking down on the ecliptic plane.

Gravity-Assist Trajectory Analysis Earth-Venus-Mars Minimum Total Delta-V

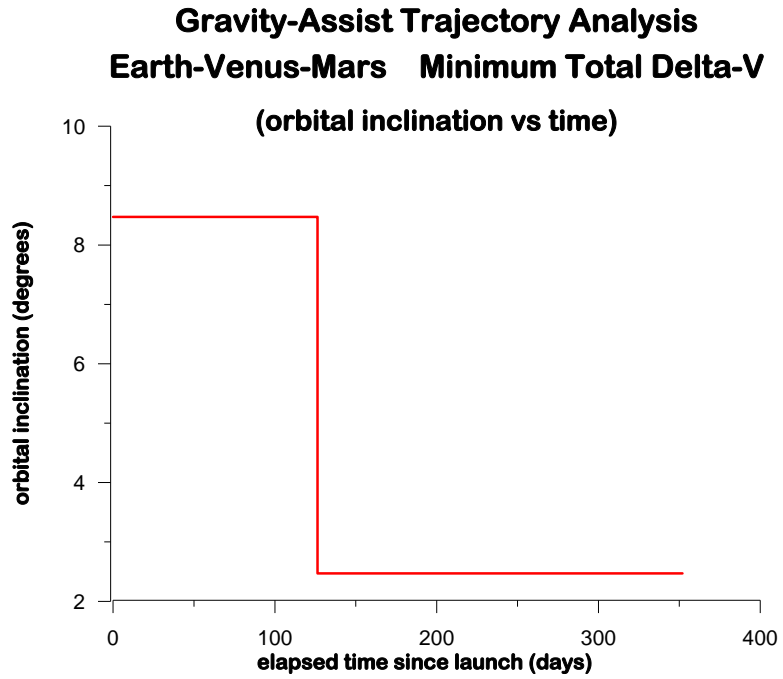


Several trajectory characteristics for this mission are shown in the following plots. These plots illustrate the effect of the gravity assist on the spacecraft's heliocentric classical orbital elements.

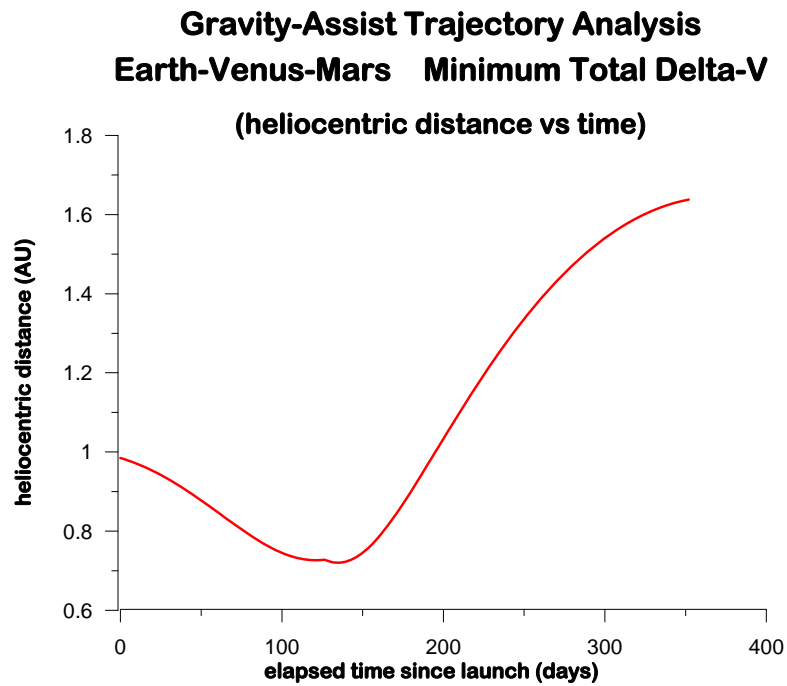
Gravity-Assist Trajectory Analysis Earth-Venus-Mars Minimum Total Delta-V (orbital eccentricity vs time)



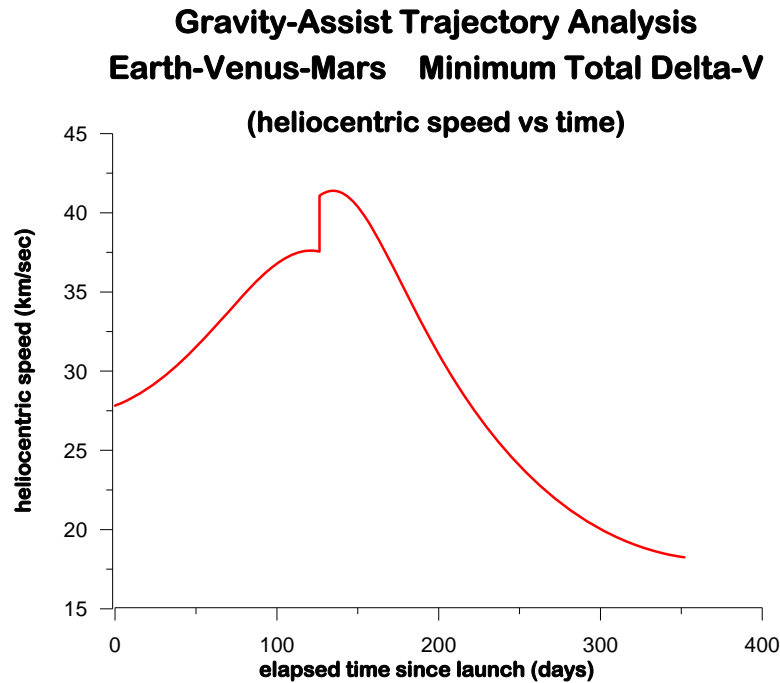
This plot illustrates the instantaneous change in orbital inclination as a result of the flyby.



This next plot illustrates the continuity of the heliocentric distance of the spacecraft.



This final plot illustrates the change in heliocentric speed produced by the flyby.



Problem setup for SOCS

This section provides additional details about the software implementation. For good scaling the time unit used in all internal calculations is days, position is expressed in astronomical units, and the velocity and delta-v unit is astronomical units per day.

This classic trajectory optimization problem is formulated using two mission phases. The first phase is the interplanetary trajectory from launch to the gravity assist at the flyby planet and the second phase is the transfer trajectory from the gravity assist to arrival at the destination planet.

(1) Launch, flyby and arrival time bounds

The software allows the user to specify an initial guess for the launch calendar date and the durations of the first and second legs of the interplanetary transfer trajectory. The user can also provide lower and upper bounds on the actual launch, flyby and arrival dates found during the optimization process.

For any guess for launch time t_L and user-defined launch time lower and upper bounds Δt_l and Δt_u , the launch time t is constrained as follows:

$$t_L - \Delta t_l \leq t \leq t_L + \Delta t_u$$

Likewise, for any guess for flyby time t_{FB} and user-defined bounds Δt_l and Δt_u , the flyby time t is constrained as follows:

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

Finally, for any guess for arrival time t_A and user-defined arrival time bounds Δt_l and Δt_u , the arrival time t is constrained as follows:

$$t_A - \Delta t_l \leq t \leq t_A + \Delta t_u$$

For fixed launch, flyby and/or arrival times, the lower and upper bounds are set to 0.

(2) Performance index – minimize delta-v

The objective function or performance index J for this simulation is one of three possible delta-v's. For this classic trajectory optimization problem, this index is simply

$$J = \Delta V$$

where ΔV is either the launch delta-v, arrival delta-v or the total delta-v. The value of the `maxmin` indicator in SOCS tells the software whether the user is minimizing or maximizing the performance index. The type of delta-v optimization is selected by the user and the performance index is enforced at the end of phase 2.

(3) Point functions – position and velocity vector “matching” at launch

For any launch time t_L the optimal solution for a rendezvous trajectory must satisfy the following state vector boundary conditions (equality constraints) at launch:

$$\begin{aligned} \mathbf{r}_{s/c}(t_L) - \mathbf{r}_p(t_L) &= 0 \\ \mathbf{v}_{s/c}(t_L) - \{\mathbf{v}_p(t_L) + \Delta \mathbf{v}(t_L)\} &= 0 \end{aligned}$$

where $\mathbf{r}_{s/c}$ and $\mathbf{v}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the launch time t_L , \mathbf{r}_p and \mathbf{v}_p are the heliocentric, inertial position and velocity vectors of the departure planet at the launch time, and $\Delta \mathbf{v}$ is the *impulsive* delta-v vector required at launch. These point constraints are enforced at the beginning of the first phase.

(4) Point functions – v-infinity and flyby altitude “matching” at flyby

Point functions are required in order to satisfy the v-infinity and altitude “match” at the flyby. These two scalar equality constraints are given by

$$\begin{aligned} |\mathbf{v}_\infty^-| - |\mathbf{v}_\infty^+| &= 0 \\ h_{fb} - h_t &= 0 \end{aligned}$$

where h_{fb} is the actual flyby altitude and h_t is the “targeted” or user-defined flyby altitude. The components of the incoming v-infinity velocity vector \mathbf{v}_∞^- are “saved” in parameters at the end of phase 1 and the outgoing v-infinity velocity vector \mathbf{v}_∞^+ and flyby altitude are computed at the beginning of phase 2. These constraints are enforced at the beginning of phase 2 using point functions that consist of the scalar difference between the v-infinity vectors and altitude.

(5) Linkage conditions – time and position vector across the flyby

In order for the interplanetary transfer trajectory to be contiguous in time and heliocentric position, the time and heliocentric position vector of the spacecraft must be “linked” across the flyby boundary. This is accomplished using the `linkst` subroutine that is part of the SOCS software suite. This linkage is enforced at the beginning of phase 2.

(6) Point functions – position and velocity vector “matching” at arrival

For any arrival time t_A the optimal solution for a rendezvous trajectory must satisfy the following state vector boundary conditions at arrival:

$$\begin{aligned} \mathbf{r}_{s/c}(t_A) - \mathbf{r}_p(t_A) &= 0 \\ \mathbf{v}_{s/c}(t_A) - \{\mathbf{v}_p(t_A) + \Delta\mathbf{v}(t_A)\} &= 0 \end{aligned}$$

where $\mathbf{r}_{s/c}$ and $\mathbf{v}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the arrival time t_A , and \mathbf{r}_p and \mathbf{v}_p are the heliocentric, inertial position and velocity vectors of the destination planet at the arrival time, and $\Delta\mathbf{v}$ is the *impulsive* delta-v vector required at arrival. This system of launch and arrival state vector equality constraints ensures a rendezvous mission. These point constraints are enforced at the end of phase 2. For a destination flyby mission, the velocity vector point functions at arrival are not enforced.

The components of the launch and arrival impulsive delta-v’s are the optimization parameters used by SOCS to solve this trajectory problem.

Initial guess

An initial guess for SOCS is created by solving the Lambert two-point boundary-value problem (TPBVP) for each leg of the interplanetary transfer trajectory using the launch calendar date and transfer time initial guesses provided by the user. The computational steps required to create the initial guess for the spacecraft state, and the launch and arrival delta-v parameters are as follows:

- (1) compute the state vector of the departure planet at the departure date initial guess

- (2) compute the state vector of the flyby planet at the flyby date initial guess
- (3) compute the state vector of the arrival planet at the arrival date initial guess
- (4) solve Lambert's problem for the departure-to-flyby leg and determine the initial velocity vector of the this leg
- (5) compute the launch delta-v vector from the launch planet's velocity vector and the initial velocity vector of the first leg of the transfer trajectory
- (6) solve Lambert's problem for the second heliocentric leg and determine the initial and final velocity vectors of the second leg
- (7) compute the arrival delta-v vector from the arrival planet's velocity vector and the final velocity vector of the second leg of the transfer trajectory
- (8) estimate the flyby incoming v-infinity vector from the flyby planet's velocity vector and the final velocity vector of the first leg of the transfer trajectory

The heliocentric state vector during each leg of the interplanetary cruise part of the mission is computed using Sheppard's two-body orbit propagation algorithm. The derivation of this algorithm is described in "Universal Keplerian State Transition Matrix", *Celestial Mechanics*, **35** (1985) 129-144. This numerical method uses a combination of universal variables and continued fractions to propagate two-body orbits.

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) = 4` within the `odeinp` subroutine. These program options create an initial guess from the numerical integration of the equations programmed in the `oderhs` subroutine. The `INIT(1) = 6` program option tells SOCS to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 4` program option tells SOCS to use the Adams predictor-corrector, variable step size numerical method to solve the initial value problem.

Technical discussion

The ΔV 's required at launch and arrival are simply the differences between the velocity on the transfer trajectory determined by the solution of Lambert's problem and the heliocentric velocities of the two planets. If we treat each planet as a point mass and assume *impulsive* maneuvers, the *planet-centered* magnitude and direction of the required maneuvers are given by the two vector equations:

$$\Delta \mathbf{V}_L = \mathbf{V}_{T_L} - \mathbf{V}_{P_L}$$

$$\Delta \mathbf{V}_A = \mathbf{V}_{T_A} - \mathbf{V}_{P_A}$$

where

\mathbf{V}_{T_L} = heliocentric velocity vector of the transfer trajectory at launch

\mathbf{V}_{T_A} = heliocentric velocity vector of the transfer trajectory at arrival

\mathbf{V}_{P_L} = heliocentric velocity vector of the launch planet

\mathbf{V}_{P_A} = heliocentric velocity vector of the arrival planet

The scalar magnitude of each maneuver is also called the “hyperbolic excess velocity” or V_∞ at launch and arrival. The hyperbolic excess velocity is the speed of the spacecraft relative to each planet at an *infinite* distance from the planet. Furthermore, the *energy* or C_3 at launch or arrival is equal to V_∞^2 for the respective maneuver. C_3 is also equal to twice the orbital energy per unit mass (the specific orbital energy).

The orientation of the departure hyperbola is specified in terms of the right ascension and declination of the outgoing asymptote. These coordinates can be calculated using the components of the V_∞ velocity vector.

The right ascension of the asymptote is determined from

$$\alpha = \tan^{-1}(\Delta V_y, \Delta V_z)$$

and the geocentric declination of the asymptote is given by

$$\delta = 90^\circ - \cos^{-1}(\Delta \hat{V}_z)$$

where $\Delta \hat{V}_z$ is z-component of the unit ΔV vector. The right ascension is computed using a four quadrant inverse tangent function.

In this computer program the heliocentric planetary coordinates and therefore the ΔV vectors are computed in the Earth mean ecliptic and equinox of J2000 coordinate system. In order to determine the orientation of the departure and arrival hyperbolas, these ΔV vectors must be transformed to the equatorial frame.

The required transformation is given by

$$\Delta \mathbf{V}_{eq} = \begin{bmatrix} 1 & -0.000000479966 & 0 \\ 0.000000440360 & 0.917482137087 & 0.397776982902 \\ -0.000000190919 & -0.397776982902 & 0.917482137087 \end{bmatrix} \Delta \mathbf{V}_{ec}$$

where $\Delta \mathbf{V}_{ec}$ is the delta-velocity vector in the ecliptic frame, and $\Delta \mathbf{V}_{eq}$ is the delta-velocity vector in the equatorial frame.

Equations of motion

The general vector equation for *point-mass* perturbations such as the Moon or planets is given by

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

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

For improved numerical behavior, use is made of Battin's $F(q)$ function given by

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

where

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

The third-body acceleration can now be expressed as

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

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

$$y_1 = r_x \quad y_2 = r_y \quad y_3 = r_z$$

$$y_4 = v_x \quad y_5 = v_y \quad y_6 = v_z$$

where r_x, r_y, r_z and v_x, v_y, v_z are the heliocentric position and velocity vector components of the spacecraft, the first-order system of differential equations of orbital motion is given by

$$\dot{y}_1 = v_x$$

$$\dot{y}_2 = v_y$$

$$\dot{y}_3 = v_z$$

$$\dot{y}_4 = -\mu_s \frac{r_x}{r^3} + a_x$$

$$\dot{y}_5 = -\mu_s \frac{r_y}{r^3} + a_y$$

$$\dot{y}_6 = -\mu_s \frac{r_z}{r^3} + a_z$$

In these equations, μ_s is the gravitational constant of the sun, and a_x , a_y and a_z are the x , y and z gravitational contributions of the planets. In these equations, $r = \sqrt{r_x^2 + r_y^2 + r_z^2}$ is the heliocentric radius magnitude of the spacecraft. All planetary perturbations except those due to the launch, flyby and arrival planets are included in the equations of motion. The equations described here are coded in the right-hand-side subroutine required by SOCS.

Gravity-assist flight mechanics

During a planetary flyby, the vector relationships between the incoming v-infinity vector \mathbf{v}_∞^- , the outgoing v-infinity vector \mathbf{v}_∞^+ and the two legs of the heliocentric transfer orbit that “patch” at the center of the flyby planet are as follows:

$$\mathbf{v}_\infty^- = \mathbf{v}_{fb} - \mathbf{v}_{to1}$$

$$\mathbf{v}_\infty^+ = \mathbf{v}_{to2} - \mathbf{v}_{fb}$$

where

\mathbf{v}_{fb} = heliocentric velocity vector of the flyby planet at the flyby date

\mathbf{v}_{to1} = heliocentric velocity vector of the first transfer orbit at the flyby date

\mathbf{v}_{to2} = heliocentric velocity vector of the second transfer orbit at the flyby date

The turn angle of the planet-centered trajectory during the flyby is determined from

$$\delta = \frac{\pi}{2} - \frac{1}{2} \sin^{-1} \left(\frac{|\mathbf{v}_\infty^- \times \mathbf{v}_\infty^+|}{|\mathbf{v}_\infty^-| |\mathbf{v}_\infty^+|} \right) = 2 \sin^{-1} \left(\frac{1}{1 + r_p v_\infty^2 / \mu} \right)$$

where r_p is the periapsis radius of the flyby hyperbola, v_∞ is the magnitude of the incoming v-infinity vector and μ is the gravitational constant of the flyby planet.

The maximum turn angle possible during a gravity assist flyby occurs when the spacecraft just gazes the planet’s surface. It is given by

$$\delta_{\max} = 2 \sin^{-1} \left(\frac{1}{1 + r_e v_\infty^2 / \mu} \right)$$

where r_e is the radius of the flyby planet.

The semimajor axis and orbital eccentricity of the flyby hyperbola are given by

$$a = -\mu / |\mathbf{v}_\infty^-|^2 = -\mu / |\mathbf{v}_\infty^+|^2$$

$$e = -1 / \cos \theta_\infty = 1 - \frac{r_p}{a} = 1 + \frac{r_p v_\infty^2}{\mu}$$

where θ_∞ is the true anomaly at infinity which is determined from the following expression:

$$\theta_\infty = \frac{\pi}{2} - \frac{1}{2} \sin^{-1} \left(\frac{|\mathbf{v}_\infty^- \times \mathbf{v}_\infty^+|}{|\mathbf{v}_\infty^-| |\mathbf{v}_\infty^+|} \right)$$

The periapsis radius of the flyby hyperbola relative to a *spherical* planet is determined from the expression $r_p = a(1 - e)$, and the flyby altitude is $h = r - r_e$ where r is the planet-centered radius at the flyby.

The heliocentric speed gained during the flyby and the heliocentric delta-v vector caused by the close encounter can be determined from the following two equations:

$$\Delta v = 2v_\infty / e$$

$$\Delta \mathbf{v} = \mathbf{v}_h^- - \mathbf{v}_h^+$$

In the second equation, \mathbf{v}_h^- is the heliocentric velocity vector of the spacecraft prior to the flyby and \mathbf{v}_h^+ is the heliocentric velocity vector after the flyby. The maximum heliocentric delta-v available from a gravity assist corresponds to a flyby at the planet's surface and is given by

$$\Delta v = \sqrt{\frac{\mu}{r_e}}$$

This value is also equal to the "local circular velocity" at the flyby planet's surface.

Planetary ephemeris

The software models the planetary coordinates using either a Meeus algorithm, the DE405 model from JPL or the SLP96 algorithm from the Bureau of Longitudes. The planetary ephemerides used in this software provide coordinates relative to the Earth mean ecliptic and equinox of J2000.

The Meeus ephemeris option is based on the algorithm described in chapter 30 of *Astronomical Algorithms* by Jean Meeus. Each orbital element is represented by a cubic polynomial of the form

$$a_0 + a_1 T + a_2 T^2 + a_3 T^3$$

where the fundamental time argument T is given by

$$T = \frac{JD - 2451545}{36525}$$

In this expression JD is the Julian date.

Please note the Meeus ephemeris does not compute coordinates for Pluto.

Comet and asteroid ephemeris

When the destination celestial body is not a planet, the classical orbital elements of an asteroid or comet relative to the ecliptic and equinox of J2000 coordinate system must be provided by the user. These elements can be obtained from the JPL Small-Body Database Browser (<http://ssd.jpl.nasa.gov/sbdb.cgi>), the MPC database at Harvard (<http://cfa-www.harvard.edu>) or the Bureau of Longitudes in Paris (<http://www.bdl.fr>).

These orbital elements consist of the following items:

- calendar date of perihelion passage
- perihelion distance (AU)
- orbital eccentricity (non-dimensional)
- orbital inclination (degrees)
- argument of perihelion (degrees)
- longitude of ascending node (degrees)

The software determines the mean anomaly of the asteroid or comet at any simulation time using the following equation:

$$M = \sqrt{\frac{\mu_s}{a^3}} t_{pp} = \sqrt{\frac{\mu_s}{a^3}} (JD - JD_{pp})$$

where μ_s is the gravitational constant of the sun, a is the semimajor axis of the celestial body, and t_{pp} is the time since perihelion passage. In the second form of this equation, JD is the current Julian date and JD_{pp} is the Julian date of perihelion passage.

The semimajor axis is determined from the perihelion distance r_p and orbital eccentricity e according to

$$a = \frac{r_p}{(1 - e)}$$

This solution of Kepler's equation in this computer program is based on a numerical solution devised by Professor J.M.A. Danby at North Carolina State University. Additional information about this algorithm can be found in "The Solution of Kepler's Equation", *Celestial Mechanics*, **31** (1983) 95-107, 317-328 and **40** (1987) 303-312.

The initial guess for Danby's method is

$$E_0 = M + 0.85 \operatorname{sign}(\sin M) e$$

The fundamental equation we want to solve is

$$f(E) = E - e \sin E - M = 0$$

which has the first three derivatives given by

$$f'(E) = 1 - e \cos E$$

$$f''(E) = e \sin E$$

$$f'''(E) = e \cos E$$

The iteration for an updated eccentric anomaly based on a current value E_n is given by the next four equations:

$$\Delta(E_n) = -\frac{f}{f'}$$

$$\Delta^*(E_n) = -\frac{f}{f' + \frac{1}{2} \Delta f''}$$

$$\Delta_n(E_n) = -\frac{f}{f' + \frac{1}{2} \Delta f'' + \frac{1}{6} \Delta^2 f'''}$$

$$E_{n+1} = E_n + \Delta_n$$

This algorithm provides quartic convergence of Kepler's equation. This process is repeated until the following convergence test involving the fundamental equation is satisfied:

$$|f(E)| \leq \varepsilon$$

where ε is the convergence tolerance. This tolerance is hardwired in the software to $\varepsilon = 1.0e-10$. Finally, the true anomaly can be calculated with the following two equations

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

$$\cos \theta = \cos E - e$$

and the four quadrant inverse tangent given by

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

If the orbit is hyperbolic, the initial guess is

$$H_0 = \log\left(\frac{2M}{e} + 1.8\right)$$

where H_0 is the hyperbolic anomaly. The fundamental equation and first three derivatives for this case are as follows:

$$f(H) = e \sinh H - H - M$$

$$f'(H) = e \cosh H - 1$$

$$f''(H) = e \sinh H$$

$$f'''(H) = e \cosh H$$

Otherwise, the iteration loop that calculates Δ, Δ^* , and so forth is the same. The true anomaly for hyperbolic orbits is determined with this next set of equations:

$$\sin \theta = \sqrt{e^2 - 1} \sinh H$$

$$\cos \theta = e - \cosh H$$

Finally, the true anomaly is determined from a four quadrant inverse tangent evaluation of these two equations.

Solving Lambert's problem

The Lambert algorithm used in this computer program is based on the method described in "A Practical Note on the Use of Lambert's Equation" by Geza Gedeon, *AIAA Journal*, Volume 3, Number 1, 1965, pages 149-150. This iterative solution is valid for elliptic, parabolic and hyperbolic transfer orbits which may be either prograde or retrograde, and involve one or more revolutions about the central body. Additional information can also be found in G. S. Gedeon, "Lambertian Mechanics", Proceedings of the 12th International Astronautical Congress, Vol. I, 172-190.

The *elliptic* form of the general Lambert Theorem is

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

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

The Gedeon algorithm introduces the following variable

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

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

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

This algorithm also makes use of the following constant:

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

The function to be solved iteratively is given by:

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

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

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

The iteration for z is as follows:

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

References and Bibliography

- “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.
- “Modern Astrodynamics”, Victor R. Bond and Mark C. Allman, Princeton Univeristy Press, 1996.
- “Fuel-Optimal, Low-Thrust, Three-Dimensional Earth-Mars Trajectories”, R. S. Nah, S. R. Vadali, and E. Braden, *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 6, November-December 2001.
- “Optimal Low-Thrust Interception of Earth-Crossing Asteroids”, Bruce A. Conway, *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 5, September-October 1997.
- “Electric Propulsion Mission Analysis”, NASA SP-210, 1969.
- “Possibilities of Combining High- and Low-Thrust Engines in Flights to Mars”, G. G. Fedotov, *Cosmic Research*, Vol. 39, No. 6, 2001, pp. 613-621.
- “Design and Optimization of Interplanetary Spacecraft Trajectories”, Thomas T. McConaghy, PhD thesis, Purdue University, December 2004.
- “Interplanetary Mission Design Handbook, Volume 1, Part 2”, JPL Publication 82-43, September 15, 1983.
- “Error Analysis of Multiple Planet Trajectories”, F. M. Sturms, Jr., JPL Space Programs Summary, No. 37-27, Vol. IV.
- “JPL Planetary Ephemeris DE410”, E. M. Standish, JPL IOM 312.N-03-009, 24 April 2003.
- “IERS Conventions (2003)”, IERS Technical Note 32, November 2003.
- “Planetary Constants and Models”, R. Vaughan, JPL D-12947, December 1995.

APPENDIX A

Compiling and Running the Software

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

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

```
df flyby_socs.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:

```
flyby_socs mars03.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 flyby-socs           *
*                                           *
*       gravity-assist trajectory         *
*       optimization with socs           *
*                                           *
*                July 20, 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 flyby-socs          *'
print *, ' *          *          *'
print *, ' *          gravity-assist trajectory      *'
print *, ' *          optimization with socs        *'
print *, ' *          *          *'
print *, ' *          July 20, 2006                  *'
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.

Important Note

The binary ephemeris files provided with this computer program were created for use on PC-compatible computers. For other platforms, you will need to create binary files specific to that system. Information and computer programs for creating these files can be found at the JPL solar system FTP site located at <ftp://ssd.jpl.nasa.gov/pub/eph/export/>.

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 `flyby_socs` software. All output except the coordinates of the launch hyperbola is computed and displayed in a heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system.

The simulation summary screen display contains the following information:

calendar date = calendar date of trajectory event

ephemeris time = ephemeris time of trajectory event

julian date = julian date of trajectory event

launch delta-vx = x-component of the impulsive velocity vector at launch in kilometers/second

launch delta-vy = y-component of the impulsive velocity vector at launch in kilometers/second

launch delta-vz = z-component of the impulsive velocity vector at launch in kilometers/second

launch delta-v = scalar magnitude of the impulsive delta-v at launch in kilometers/second

right ascension = right ascension of the launch hyperbola in degrees

declination = declination of the launch hyperbola in degrees

launch energy = launch energy in km/sec squared

sma (au) = semimajor axis in astronomical unit

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 (days) = orbital period in days

rx (km) = x-component of spacecraft or celestial body position vector in kilometers

ry (km) = y-component of spacecraft or celestial body position vector in kilometers

rz (km) = z-component of spacecraft or celestial body position vector in kilometers

rmag (km) = heliocentric radius magnitude of spacecraft or celestial body in kilometers

vx (kps) = x-component of spacecraft or celestial body velocity vector in kilometers per second

vy (kps) = y-component of spacecraft or celestial body velocity vector in kilometers per second

vz (kps) = z-component of spacecraft or celestial body velocity vector in kilometers per second

vmag (kps) = heliocentric velocity magnitude of spacecraft or celestial body in kilometers per second

incoming v-infinity = incoming "speed at infinity" in kilometers per second

outgoing v-infinity = outgoing "speed at infinity" in kilometers per second

flyby altitude = altitude of the flyby in kilometers

maximum turn angle = the turn angle if the spacecraft grazed the flyby planet's surface in degrees

actual turn angle = actual turn angle produced by the flyby in degrees

true anomaly at infinity = true anomaly of the spacecraft at an infinite distance from the flyby planet

flyby delta-vx = x-component of the heliocentric impulsive velocity vector due to the flyby in kilometers/second

flyby delta-vy = y-component of the heliocentric impulsive velocity vector due to the flyby in kilometers/second

flyby delta-vz = z-component of the heliocentric impulsive velocity vector due to the flyby in kilometers/second

flyby delta-v = scalar magnitude of the heliocentric impulsive delta-v due to the flyby in kilometers/second

maximum delta-v = the maximum possible heliocentric delta-v due to the flyby in kilometers/second

arrival delta-vx = x-component of the impulsive velocity vector at arrival in kilometers/second

arrival delta-vy = y-component of the impulsive velocity vector at arrival in kilometers/second

arrival delta-vz = z-component of the impulsive velocity vector at arrival in kilometers/second

arrival delta-v = scalar magnitude of the impulsive delta-v at arrival in kilometers/seconds

arrival energy = arrival energy in km/sec squared

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

`time (days)` = simulation time since launch in days
`rx (au)` = x-component of spacecraft position vector in astronomical units
`ry (au)` = y-component of spacecraft position vector in astronomical units
`rz (au)` = z-component of spacecraft position vector in astronomical units
`rmag (au)` = heliocentric radius magnitude of spacecraft in astronomical units
`vx (km/sec)` = x-component of spacecraft velocity vector in kilometers per second
`vy (km/sec)` = y-component of spacecraft velocity vector in kilometers per second
`vz (km/sec)` = z-component of spacecraft velocity vector in kilometers per second
`vmag (km/sec)` = heliocentric velocity of spacecraft in kilometers per second
`semimajor axis (au)` = semimajor axis in astronomical units
`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 (days)` = orbital period in days

The `planets.csv` file contains the following information:

`time (days)` = simulation time since launch in days
`rp1-x (au)` = x-component of the launch planet position vector in astronomical units
`rp1-y (au)` = y-component of the launch planet position vector in astronomical units
`rp1-z (au)` = z-component of the launch planet position vector in astronomical units
`rp2-x (au)` = x-component of the destination body position vector in astronomical units
`rp2-y (au)` = y-component of the destination body position vector in astronomical units
`rp2-z (au)` = z-component of the destination body position vector in astronomical units

APPENDIX C

Fortran Functions and Subroutines

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

- flyby_socS.f** - SOCS main executive program
- atan3.for** - four quadrant inverse tangent function
- eci2orb.for** - convert eci position and velocity vectors to classical orbital elements subroutine
- fbhyper.for** - compute orbital elements of a planet-centered flyby hyperbola at periapsis passage subroutine
- gdate.for** - compute calendar date from Julian date subroutine
- jd2str.for** - from a Julian date, print the character representation of calendar date and time subroutine
- jpleph.for** - subroutine that reads and interpolates a JPL ephemeris file
- julian.for** - subroutine to convert calendar date to Julian date
- kepler1.for** - solve Kepler's equation using Danby's method subroutine
- linput.for** - read and echo a line of text from an input file subroutine
- lambfunc.for** - solve Lambert's problem using Geodeon's method 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
- oeprint.for** - subroutine that displays classical orbital elements
- orb2eci.for** - convert classical orbital elements to position and velocity vectors subroutine
- p2000.for** - subroutine that returns a planet's or asteroid/comet position and velocity vectors in kilometers and kilometers/second
- 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
- slp96.for** - read and interpolate SLP96 ephemeris subroutine

svprint.for - subroutine that displays position and velocity vectors
twobody2.for - subroutine that solves the two-body initial value problem
utility.for - number and text manipulation functions and subroutines
uvector.for - unit vector subroutine
vcross.for - vector cross product subroutine
vdot.for - vector dot product subroutine
vecmag.for - vector scalar magnitude function
xmod.for - modulo 2 pi function

APPENDIX D

Example Fortran Subroutine

This appendix contains the source code for a single Fortran 77 routine and illustrates typical programming conventions used in the `flyby_socs` software. This subroutine is the point function routine required by the SOCS software.

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

c     evaluate phase dependent "position & velocity matching"
c     point functions and scalar delta-v objective function

c     *****

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

      save

      include 'socscom1.inc'

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

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

      dimension rp(3), vp(3), rp2sc(3), vp2sc(3), rhat(3), vhat(3)

      dimension vinf_in(3), vinf_out(3), vlxv2(3)

      data pi / 3.141592653589793d0 /

      data rtd / 57.295779513082323d0 /

c     initialize evaluation error flag

      iferr = 0

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

c     current julian date

      xjdate = xjdateil + time

c     compute launch planet position (km) and velocity (km/sec) vectors

      call p2000(ip1, xjdate, rp, vp)

c     position vector match point functions (au)

      ptf(1) = ydyn(1) - rp(1) / aunit

      ptf(2) = ydyn(2) - rp(2) / aunit
```

```

ptf(3) = ydyn(3) - rp(3) / aunit
c
velocity vector match point functions (au/day)
ptf(4) = ydyn(4) - (vmult * vp(1) + parm(1))
ptf(5) = ydyn(5) - (vmult * vp(2) + parm(2))
ptf(6) = ydyn(6) - (vmult * vp(3) + parm(3))

if (iopt .eq. 1 .or. iopt .eq. 3) then
c
c
c
*****
launch delta-v contribution to objective function
*****

ptf(7) = sqrt(parm(1)**2 + parm(2)**2 + parm(3)**2)

endif

end if

if (iphase .eq. 1 .and. iphend .eq. +1) then
c
c
c
c
*****
flyby planet position vector match and incoming
v-infinity point functions at the end of phase 1
*****

c
current julian date

xjdate = xjdate1 + time

c
compute launch planet position (km) and velocity (km/sec) vectors

call p2000(ip2, xjdate, rp, vp)

c
position vector match point functions (au)

ptf(1) = ydyn(1) - rp(1) / aunit

ptf(2) = ydyn(2) - rp(2) / aunit

ptf(3) = ydyn(3) - rp(3) / aunit

c
compute incoming v-infinity vector (km/sec)

do i = 1, 3
vinf_in(i) = ydyn(i + 3) / vmult - vp(i)
end do

c
load incoming v-infinity vector point functions

ptf(4) = vinf_in(1)

ptf(5) = vinf_in(2)

ptf(6) = vinf_in(3)

```

```

c      scale incoming v-infinity vector point functions

      ptf(4) = 1.0d-6 * ptf(4)

      ptf(5) = 1.0d-6 * ptf(5)

      ptf(6) = 1.0d-6 * ptf(6)

end if

if (iphase .eq. 2 .and. iphend .eq. -1) then
c      *****
c      "v-infinity and flyby altitude match" at the beginning of phase 2
c      *****

c      current julian date

      xjdate = xjdateil + time

c      compute flyby planet position (km) and velocity (km/sec) vectors

      call p2000(ip2, xjdate, rp, vp)

c      load incoming v-infinity vector from parameter vector (km/sec)

      do i = 1, 3
        vinf_in(i) = parm(i + 3)

c      scale point functions

        ptf(i) = 1.0d-6 * parm(i + 3)
      end do

c      compute outgoing v-infinity vector (km/sec)

      do i = 1, 3
        vinf_out(i) = ydyn(i + 3) / vmult - vp(i)
      end do

c      calculate v-infinity magnitudes (km/sec)

      vinfm_in = vecmag(vinf_in)

      vinfm_out = vecmag(vinf_out)

c      -----
c      v-infinity match constraint
c      -----

      ptf(4) = vinfm_in - vinfm_out

c      scale point function

      ptf(4) = 1.0d-2 * ptf(4)

      call vcross(vinf_in, vinf_out, vlxv2)

```

```

v1xv2m = vecmag(v1xv2)

c true anomaly at infinity (radians)

phil = 0.5d0 * pi + 0.5d0 * asin(v1xv2m
&      / (vinfm_in * vinfm_out))

c calculate planet-centered hyperbolic orbital elements

fbecc = -1.0d0 / cos(phil)

fbsma = -pmu(ip2) / (vinfm_in * vinfm_in)

fbrp = fbsma * (1.0d0 - fbecc)

c actual flyby altitude (kilometers)

fbaltitude = fbrp - rep(ip2)

c -----
c flyby altitude match constraint
c -----

ptf(5) = fbaltitude - fbalt

c scale constraint

ptf(5) = 1.0d-4 * ptf(5)

end if

if (iphase .eq. 2 .and. iphend .eq. +1) then
c *****
c arrival planet "position & velocity match" at the end of phase 2
c *****

c current julian date

xjdate = xjdatei1 + time

c compute arrival planet position (km) and velocity (km/sec) vectors

call p2000(ip3, xjdate, rp, vp)

c position vector match point functions (au)

ptf(1) = ydyn(1) - rp(1) / aunit
ptf(2) = ydyn(2) - rp(2) / aunit
ptf(3) = ydyn(3) - rp(3) / aunit

c if (itraj .eq. 2) then
velocity vector match point functions (au/day)

ptf(4) = ydyn(4) - (vmult * vp(1) + parm(1))

```

```

    ptf(5) = ydyn(5) - (vmult * vp(2) + parm(2))

    ptf(6) = ydyn(6) - (vmult * vp(3) + parm(3))
end if

if (iopt .eq. 2 .or. iopt .eq. 3) then
c      *****
c      arrival delta-v contribution to objective function
c      *****

    ptf(7) = sqrt(parm(1)**2 + parm(2)**2 + parm(3)**2)

endif

end if

return
end

```

APPENDIX E

Example Free Return Trajectory

The following is the graphics display for a typical free return trajectory computed with the flyby_socs software. It involves a flyby of Mars

Gravity-Assist Trajectory Analysis Earth-Mars Free Return Minimum Total Delta-V

