Solar Sail Trajectory Analysis with MATLAB

This document describes an interactive MATLAB script named `ss2d_opt.m` that can be used to analyze and optimize two-dimensional, heliocentric solar sail trajectories between the orbits of Earth and Venus (inner transfer) and between the Earth and Mars (outer transfer). In this script, the heliocentric planet orbits are assumed to be circular and coplanar. The optimal steering angles for minimum transfer time are modeled as discretized, piecewise-linear variations as suggested in the technical paper, "Near Minimum-Time Trajectories for Solar Sails", by Michiel Otten and Colin R. McInnes, AIAA *Journal of Guidance, Control and Dynamics*, Vol. 24, No. 3.

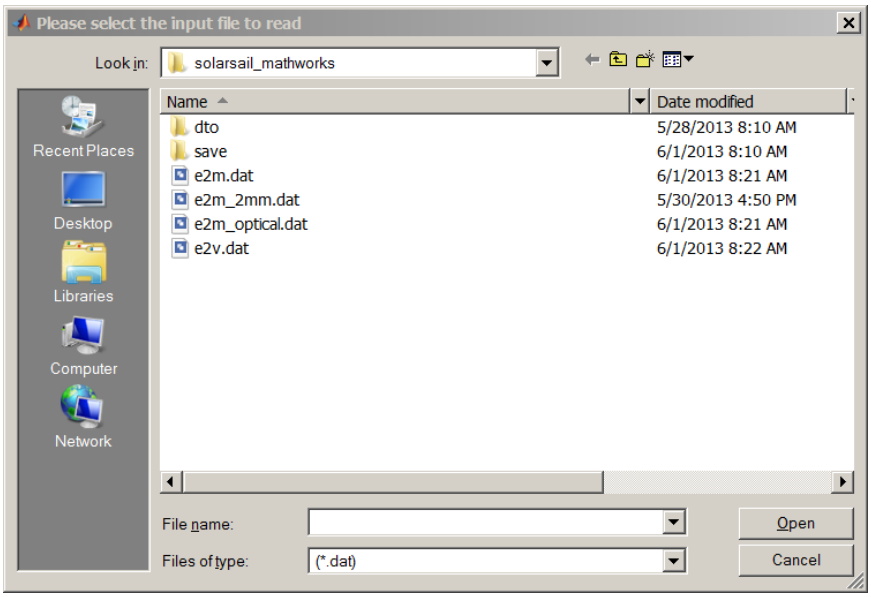
The optimization of these steering angles is performed using the SNOPT nonlinear programming (NLP) algorithm. MATLAB versions of SNOPT for several computer platforms can be found at Professor Philip Gill's web site which is located at http://ccom.ucsd.edu/~optimizers/. Professor Gill's web site also includes a PDF version of the SNOPT software user's guide.

This MATLAB script also estimates the departure and arrival calendar dates based on an initial calendar date guess provided by the user. The `ss2d_opt` script provides both numerical and graphical information about the trajectory analysis.

Interacting with the script

To execute the `ss2d_opt` script, log into the directory containing the source code and type **`ss2d_opt`** in the MATLAB command window.

The `ss2d_opt` MATLAB script is "data driven" by a simple text file created by the user. The script will prompt the user for the name of the data file with a screen similar to



The file type defaults to names with a `*.dat` filename extension. However, you can select any compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

The following are the contents of a typical `ss2d_opt` compatible input data file. Please note the proper units. User input is denoted in bold font.

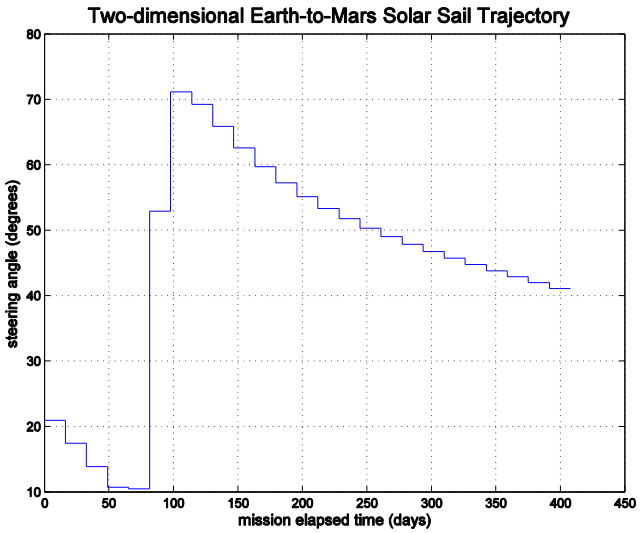
```
*************************************
* input data file for ss2d_opt.m
* Earth-to-Mars trajectory - ideal sail
*************************************

number of trajectory segments
25

characteristic acceleration (meters/second**2)
0.001

optical force model coefficient b1 (b1 = 0 = ideal)
0.0

optical force model coefficient b2 (b2 = 1 = ideal)
1.0

optical force model coefficient b3 (b3 = 0 = ideal)
0.0

target planet (1 = Venus, 2 = Mars)
2

initial guess for mission duration (days)
400

lower bound for mission duration (days)
250

upper bound for mission duration (days)
500

initial guess for steering angles (degrees)
20.0

lower bound for steering angles (degrees)
0.0

upper bound for steering angles (degrees)
90.0

departure calendar date initial guess (month, day, year)
1,1,2020
```

The following is the numerical information output by the script for this example.

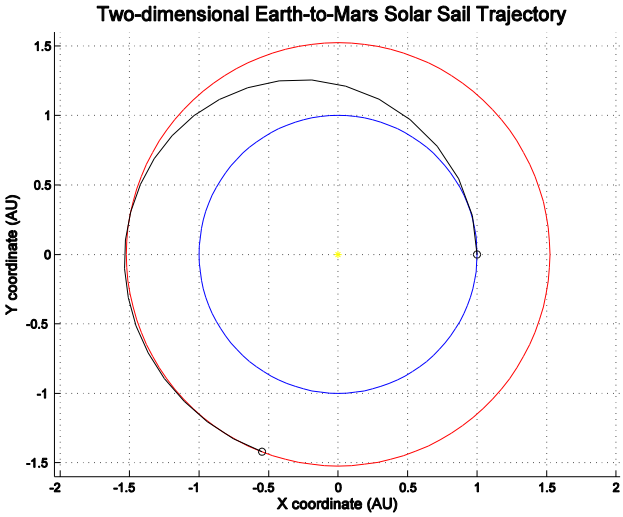
```
program ss2d_opt - Earth-to-Mars

number of segments 25

initial state vector

  radius                  1.00000000 (AU)

  radial velocity         0.00000000 (AU/day)

  transverse velocity     1.00000000 (AU/day)


final state vector
```

```
  radius                    1.52368000 (AU)

  radial velocity           0.00000000 (AU/day)

  transverse velocity       0.81012702 (AU/day)

total transfer time      7.01572813 (non-dimensional)

total transfer time    407.84140084 days

total transfer angle   248.92292164 degrees

synodic period             779.948575 days
                             2.135383 years

departure calendar date       10-Jul-2020

arrival calendar date         21-Aug-2021
```

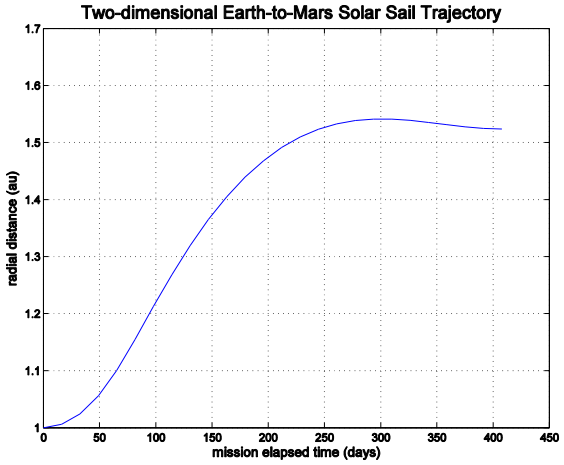
The following is a plot of the optimal piecewise-linear steering angles during the transfer.

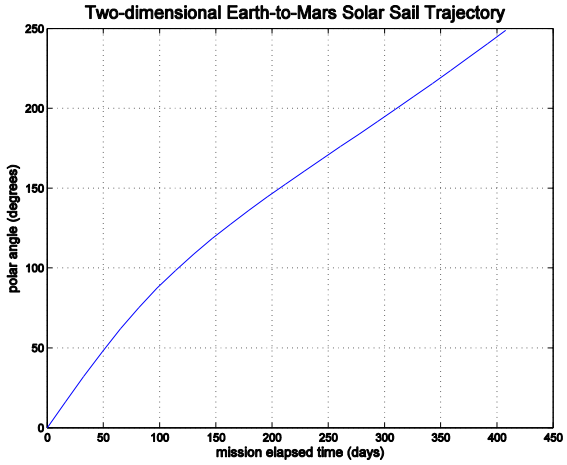


This plot illustrates the transfer trajectory between the Earth (blue) and Mars (red). The beginning and end of the transfer is marked with a small **o**. Please note the scale for the x- and y-coordinates are Astronomical Units.

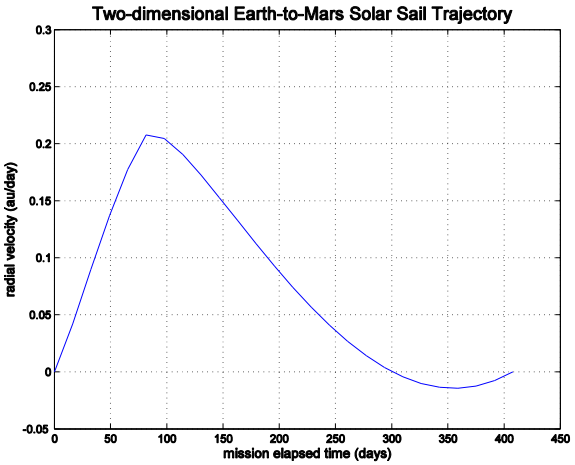


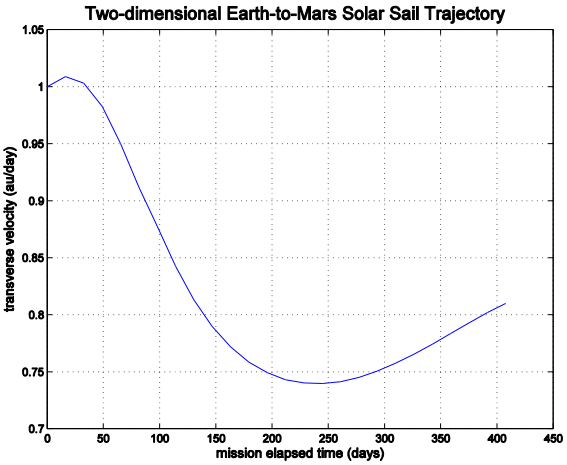
The following two plots illustrate the behavior of the radial distance and polar angle during the mission. The scale for the radial distance is Astronomical Units and the polar angle scale is degrees.





These next two plots illustrate the behavior of the radial and tangential or transverse velocities during the transfer. The scale for both velocity plots is AU/day.





The `ss2d_opt` MATLAB script will create color Postscript disk files of these graphic images. These images include a TIFF preview and are created with MATLAB code similar to

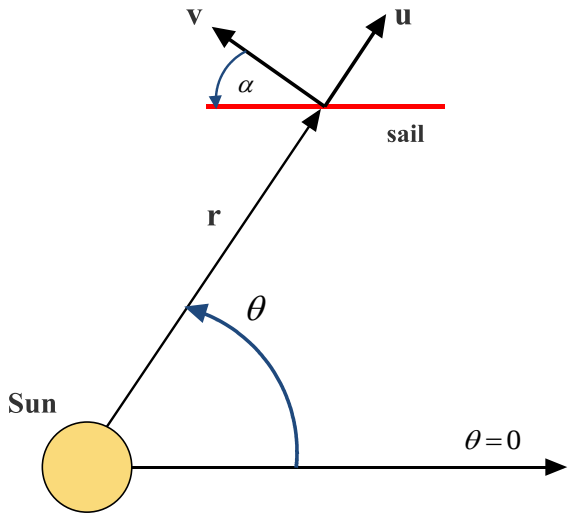
```
print -depsc -tiff -r300 polar_angle.eps
```

Additional script examples can be found in Appendix A.

Technical Discussion

This trajectory optimization problem is modeled in a two-dimensional polar coordinate system. The planets are assumed to be in circular and coplanar heliocentric orbits. No allowance is made for the eccentricity or orbital inclination of the planetary orbits. Therefore, all orbital motion is confined to the ecliptic plane.

The following diagram illustrates the geometry of this coordinate system along with the orientation of the steering angle.



In this diagram, **r** is the heliocentric distance of the solar sail, θ is the polar angle, **v** is the transverse or tangential component of the velocity, **u** is the radial component of the solar sail's velocity and α is the steering or solar sail orientation angle. The steering angle is measured relative to the tangential direction and is positive in the counter-clockwise direction.

Solar radiation pressure

Space travel by solar sail is made possible by solar radiation pressure (SRP). The solar radiation force results from the impingement of photons on the reflective, Sun-facing surface of the solar sail.

The solar radiation pressure at any heliocentric distance *r* is given by

$$P = P_{\oplus}\left(\frac{r_{\oplus}}{r}\right)^2 = \frac{S_0}{c}\left(\frac{r_{\oplus}}{r}\right)^2$$

Where

$P_{\oplus} = 4.563\,\mu N/m^2$ = SRP at 1 AU

$S_0 = 1368\,w/m^2$ at 1 AU

$r_{\oplus}$ = distance at 1 AU

c = speed of light

Dimensional analysis

To "streamline" the numerical calculations, the fundamental distance, velocity and time in the equations of motion are normalized. In this MATLAB script, all heliocentric distances are normalized with respect to the Astronomical Unit which is equal to `149597870.691` kilometers. Likewise, all velocity values are normalized with respect to the "local circular velocity" at the heliocentric distance of the Earth's circular orbit, r_0. Therefore, the velocity unit is $\sqrt{\mu_{\oplus}/r_0} = \sqrt{\mu_{\oplus}}$ since r_0 is equal to 1 AU.

Finally, all time values are normalized with respect to $\sqrt{r_0^3/\mu_\oplus} = \sqrt{1/\mu_\oplus}$ since again r_0 is equal to 1 AU. In these equations, $\mu_\oplus$ is the gravitational constant of the Sun. The corresponding value for this astronomical constant is $\mu_\oplus = 0.00029591220828559 12\, AU^3/day^2$.

Equations of motion

The two-dimensional equations of motion in the polar coordinate system are given by

$$\dot{r} = u$$

$$\dot{\theta} = \frac{v}{r}$$

$$\dot{u} = \frac{v^2}{r} - \frac{1}{r^2} + \left(\frac{\tilde{a}}{r^2}\right)\cos\alpha\left(b_1 + b_2\cos^2\alpha + b_3\cos\alpha\right)$$

$$\dot{v} = -\frac{uv}{r} + \left(\frac{\tilde{a}}{r^2}\right)\cos\alpha\sin\alpha\left(b_2\cos\alpha + b_3\right)$$

where

$r =$ radial distance

$\theta =$ polar angle

$u =$ radial velocity

$v =$ transverse (tangential) velocity

$\alpha =$ solar sail steering angle

$\tilde{a} =$ acceleration ratio

$b_1, b_2, b_3 =$ sail optical properties

The acceleration ratio is the ratio of the of the acceleration due to SRP and the acceleration due to the point-mass gravity of the Sun, both evaluated at a distance of 1 AU. The SRP acceleration is also called the *characteristic* acceleration which is defined to be the acceleration experienced by an ideal solar sail oriented perpendicular to the direction of the Sun at a heliocentric distance of 1 AU.

Therefore, the acceleration ratio $\tilde{\alpha}$ is equal to $a_c/a_\oplus$ where a_c is the characteristic acceleration and $a_\oplus$ is the solar acceleration. Values for the characteristic acceleration are typically 0.5 to 1.0 millimeters per second^2. The acceleration due to the Sun is equal to μ/r^2. If we convert the characteristic acceleration in meters per second to normalized units according to

$$a_c \rightarrow \frac{m}{s^2} \bullet \left[\frac{(86400\, s/day)^2}{1000\, m/km \bullet 149597870.691 km/AU}\right]$$

we will have characteristic acceleration in units of AU per day. Likewise, we can convert the solar acceleration to the same unit according to

$$a_{\oplus} = \frac{\mu}{r^2} \rightarrow \frac{\left(km^3/s^2\right)}{km^2} = \left(\frac{km}{s^2}\right) \bullet \frac{\left(86400\, s/day\right)^2}{149597870.691 km/AU}$$

The sail optical properties are a function of the method and material used to manufacture the sail. For an ideal solar sail, $b_1 = b_3 = 0$ and $b_2 = 1$. According to "Solar sail trajectories with piecewise-constant steering laws", by Giovanni Mengali and Alessandro A. Quarta, *Aerospace Science and Technology*, 13 (2009) 431-441, the values for a solar sail with a highly reflective aluminum-coated front side and a highly emissive chromium-coated back side are $b_1 = 0.0864, b_2 = 0.8272$ and $b_3 = -0.00545$.

Trajectory optimization

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

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

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

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

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

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

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

For any mission time *t* there are also simple bounds on the state variables

$$\mathbf{y}_l \le \mathbf{y}(t) \le \mathbf{y}_u$$

the control variables

$$\mathbf{u}_l \le \mathbf{u}(t) \le \mathbf{u}_u$$

and the problem parameters

$$\mathbf{p}_l \le \mathbf{p}(t) \le \mathbf{p}_u$$

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

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

while satisfying all the user-defined mission constraints.

In this MATLAB script, the total transfer time is the objective function which we are attempting to minimize. The control variables are the steering angles in each time segment. The final boundary conditions or equality constraints are the heliocentric distance and velocities at the destination planet.

The initial conditions are fixed to the normalized values $r_0 = 1, u_0 = 0, v_0 = 1, \theta_0 = 0$. The final boundary conditions are $r_f = 0.723331$ for Venus, $r_f = 1.52368$ for Mars, $u_f = 0, v_f = \sqrt{1/r_f}$. The final polar angle is not constrained since we are solving a minimum time orbital transfer problem.

The following is the MATLAB source code that initializes the optimization problem. It establishes the proper initial and final conditions, calculates initial guesses for the steering angles and objective function, and also sets lower and upper bounds on the final dynamic variables and objective function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initial and final times and states
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% define state vector at initial time

xinitial(1) = 1.0d0;

xinitial(2) = 0.0d0;

xinitial(3) = 1.0d0;

xinitial(4) = 0.0;

% final conditions

if (iplanet == 1)

    % Venus

    xfinal(1) = 0.723331;

else

    % Mars

    xfinal(1) = 1.52368d0;

end

xfinal(2) = 0.0d0;

xfinal(3) = sqrt(1.0d0 / xfinal(1));

% initial guess for non-dimensional transfer time
```

```
xg(1) = time_g / tfactor;

% initial guess for steering angles (radians)

xg(2:nsegments + 1) = alpha_g;

% transpose initial guess

xg = xg';

% upper and lower bounds for non-dimensional transfer time

xlb(1) = time_lb / tfactor;

xub(1) = time_ub / tfactor;

% upper and lower bounds for steering angles (radians)

xlb(2:nsegments + 1) = alpha_lb;

xub(2:nsegments + 1) = alpha_ub;

% transpose bounds

xlb = xlb';

xub = xub';

% define lower and upper bounds on objective function (transfer time)

flow(1) = 0.0d0;

fupp(1) = +Inf;

% define bounds on final state vector equality constraints

flow(2) = xfinal(1);
fupp(2) = xfinal(1);

flow(3) = xfinal(2);
fupp(3) = xfinal(2);

flow(4) = xfinal(3);
fupp(4) = xfinal(3);


flow = flow';

fupp = fupp';

xmul = zeros(nsegments + 1, 1);

xstate = zeros(nsegments + 1, 1);

fmul = zeros(4, 1);

fstate = zeros(4, 1);
```

The following is the call to the SNOPT algorithm to solve the problem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% solve solar sail shooting problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

snscreen('on');

[x, f, inform, xmul, fmul] = snopt(xg, xlb, xub, xmul, xstate, ...
    flow, fupp, fmul, fstate, 'ss2d_shoot');
```

The following is the MATLAB source code for the function that performs the shooting calculations. This function starts with the initial conditions and integrates the equations of motion along each trajectory segment using the current values of the steering angles for each segment.

```
function [f, g] = ss2d_shoot (x)

% objective function and equality constraints

% simple shooting method

% inputs

%  x(1) = current value of transfer time (objective function)
%  x(2, nsegments) = current values of steering angle alpha

% outputs

%  f = vector of equality constraints and
%      objective function evaluated at x

% Orbital Mechanics with MATLAB

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

global tfactor nsegments alpha_wrk xinitial

% compute duration of each time interval (non-dimensional)

deltat = x(1) / nsegments;

% specify number of differential equations

neq = 4;

% truncation error tolerance

tetol = 1.0e-10;

% initialize initial time

ti = -deltat;

% total non-dimensional time of flight


tof = x(1);

% set initial conditions
```

```
yi(1) = xinitial(1);
yi(2) = xinitial(2);
yi(3) = xinitial(3);
yi(4) = xinitial(4);
% step size guess (non-dimensional time)
h = (1200.0 / 86400.0) / tfactor;
% integrate for all segments
for i = 1:1:nsegments
    alpha_wrk = x(i + 1);
    % increment initial and final times
    ti = ti + deltat;
    tf = ti + deltat;
    % integrate from current ti to tf
    yfinal = rkf78('ss2d_eqm_opt', neq, ti, tf, h, tetol, yi);
    % reset integration vector
    yi = yfinal;
    % check for end of simulation
    if (tf >= tof)
        break;
    end
end
% objective function (minimize non-dimensional transfer time)
f(1) = x(1);
% compute equality constraints (final state boundary conditions)
f(2) = yfinal(1);
f(3) = yfinal(2);
f(4) = yfinal(3);
% transpose
f = f';
% no derivatives
```

```
g = [];
```

The following is the MATLAB function that evaluates the two-dimensional equations of motion.

```
function ydot = ss2d_eqm_opt (t, y)

% two-dimensional solar sail polar equations of motion

% required by ss2d_opt.m

% input

%  t    = non-dimensional simulation time
%  y(1) = radial distance (r)
%  y(2) = radial component of velocity (u)
%  y(3) = tangential component of velocity (v)
%  y(4) = polar angle (radians)

% output

%  ydot(1) = r-dot
%  ydot(2) = u-dot
%  ydot(3) = v-dot
%  ydot(4) = theta-dot

% Orbital Mechanics with MATLAB

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

global b1 b2 b3 acc_srp alpha_wrk

% evaluate equations of motion at current conditions

r = y(1);

u = y(2);

v = y(3);

afactor = (acc_srp / r^2) * cos(alpha_wrk);

% r-dot

ydot(1) = u;

% u-dot

ydot(2) = (v^2 / r) - (1.0 / r^2) + afactor * (b1 + b2 * cos(alpha_wrk)^2 ...
    + b3 * cos(alpha_wrk));

% v-dot


ydot(3) = -(u * v / r) + afactor * sin(alpha_wrk) * (b2 * cos(alpha_wrk) + b3);

% theta-dot

ydot(4) = v / r;
```

Predicting the departure and arrival calendar dates

In order to determine a rendezvous trajectory with Mars, this script computes the "delta-time" to add to the user-defined guess for the departure calendar date. The following equation defines this offset based on the user's initial guess and the orbital geometry determined by the `ss2d_opt` MATLAB script.

$$\Delta t = \frac{\theta_M\left(JD_G\right) + \omega_M t_T - \theta_E\left(JD_G\right) - \theta_T}{\omega_E - \omega_M}$$

where

JD_G = user-defined initial Julian Date guess
θ_E = celestial longitude of Earth
θ_M = celestial longitude of Mars
ω_E = heliocentric mean motion of Earth
ω_M = heliocentric mean motion of Mars
θ_T = heliocentric transfer angle

The mean motion of each planet (in degrees per day) is determined from $\omega = 360^\circ / \tau$ where τ is the orbital period of the planet (in days) computed from $\tau = 2\pi\sqrt{r^3/\mu}$. In the orbital period equation, *r* is the radius of the planet's heliocentric orbit and μ is the gravitational constant of the sun. The arrival calendar date is equal to the "solved-for" departure date plus the total transfer time in days.

The synodic period is the time required to repeat a given relative geometry between any two planets in heliocentric circular orbits. The synodic period between the Earth and Mars is given by $\tau_{SYN} = 360^\circ / \left|\omega_E - \omega_M\right|$ where the angular rates in this equation are in degrees per day.

The following is the MATLAB function that performs these calculations.

```
function [jd_depart, tsynodic] = departure(jd_guess, transfer_time, transfer_angle)

% estimate departure calendar date

% input

%  jd_guess       = departure julian date initial guess
%  transfer_time  = Earth-to-Mars transfer time (days)
%  transfer_angle = Earth-to-Mars heliocentric transfer angle (degrees)

% output

%  jd_depart = departure julian date
%  tsynodic  = Earth/Mars synodic period (days)

% Orbital Mechanics with MATLAB

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

global aunit xmu

pi2 = 2.0 * pi;
```

```
% time arguments

t = (jd_guess - 2451545.0) / 36525.0;

t2 = t * t;

% celestial longitude of the Earth and Mars at jd_guess (degrees)

theta_earth =  mod(100.466449 + 35999.3728519 * t - 0.00000568 * t2, 360.0);

theta_mars =  mod(355.433275 + 19140.2993313 * t + 0.00000261 * t2, 360.0);

% orbital period of the Earth and Mars (days)

tau_earth = pi2 * sqrt(aunit^3 / xmu) / 86400.0;

tau_mars = pi2 * sqrt((aunit * 1.52368)^3 / xmu) / 86400.0;

% orbital rate of the Earth and Mars (degrees/day)

omega_earth = 360.0 / tau_earth;

omega_mars = 360.0 / tau_mars;

% compute change to initial guess (days)

delta_t = (theta_mars + omega_mars * transfer_time - theta_earth - ...
    transfer_angle) / (omega_earth - omega_mars);

% departure julian date

jd_depart = jd_guess + abs(delta_t);

% synodic period (days)

tsynodic = 360.0 / abs(omega_earth - omega_mars);
```

The derivation and MATLAB implementation of the equations required for Earth-to-Venus trajectories is left as an exercise for the user.

Algorithm resources

Colin R. McInnes, *Solar Sailing:Technology, Dynamics and Mission Applications*, Springer-Verlag, Berlin, 1999.

J. L. Wright, *Space Sailing*, Gordon and Breach, New York, 1992.

C. G. Sauer, Jr., "Optimum Solar Sail Interplanetary Trajectories", AIAA Paper 76-0792, 1976.

Kirill Simon and Yuri Zakharov, "Optimization of Interplanetary Trajectories with Solar Sail", IAF-95-A.2.08.

Bernd Dachwald, "Optimal Solar Sail Trajectories for Mission to the Outer Solar System", AIAA *Journal of Guidance, Control and Dynamics*, Vol. 28, No. 1, 2005, pp. 173-177.

Giovanni Mengali and Alessandro A. Quarta, "Semi-Analytical Method for the Analysis of Solar Sail Heliocentric Orbit Raising", AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 1, January-February 2012.

Giovanni Mengali and Alessandro A. Quarta, "Optimal Three-Dimensional Interplanetary Rendezvous Using Nonideal Solar Sail", AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 1, January-February 2005.

Bernd Dachwald and Malcolm Macdonald, "Parametric Model and Optimal Control of Solar Sails with Optical Degradation", AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 5, September-October 2006.

Guido Colasurdo and Lorenzo Casalino, "Optimal Control for Interplanetary Trajectories with Nonideal Solar Sail", AIAA *Journal of Spacecraft and Rockets*, Vol. 40, No. 2, March-April 2003.

Victoria L. Coverstone and John E. Prussing, "Technique for Escape from Geosynchronous Transfer Orbit Using a Solar Sail, AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 4, July-August 2003.

Appendix A

Additional Script Examples

This appendix summarizes input data files and script results for two additional solar sail trajectory optimization examples. The first example is an Earth-to-Venus inner transfer (`e2v.dat`) with an ideal solar sail and the second example illustrates an Earth-to-Mars outer transfer mission (`e2m_optical.dat`) with a non-ideal solar sail.

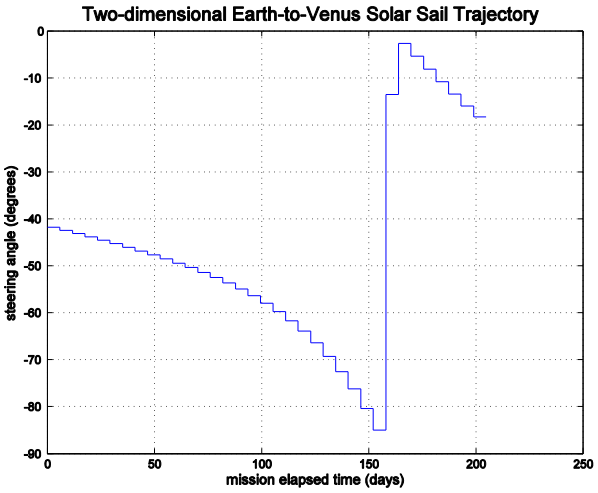
Earth-to-Venus with ideal solar sail

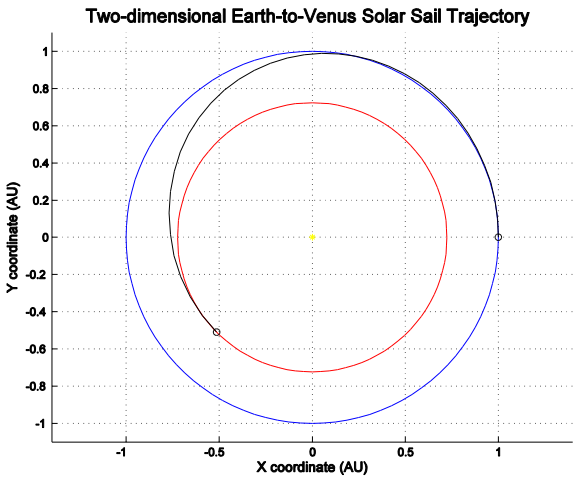
For this example, the steering angles are negative since they are bounded by $-90^\circ \le \alpha \le 0^\circ$. These bounds force the script to fly an interplanetary transfer to an inner planet.

```
**************************************
* input data file for ss2d_opt.m
* Earth-to-Venus trajectory - ideal sail
**************************************

number of trajectory time segments
35

characteristic acceleration (meters/second**2)
0.001

optical force model coefficient b1 (b1 = 0 = ideal)
0.0

optical force model coefficient b2 (b2 = 1 = ideal)
1.0

optical force model coefficient b3 (b3 = 0 = ideal)
0.0

target planet (1 = Venus, 2 = Mars)
1

initial guess for mission duration (days)
200

lower bound for mission duration (days)
150

upper bound for mission duration (days)
250

initial guess for steering angles (degrees)
-20.0

lower bound for steering angles (degrees)
-90.0

upper bound for steering angles (degrees)
0.0

departure calendar date initial guess (month, day, year)
1,1,2020
```

The following is the script output for this example.

```
program ss2d_opt - Earth-to-Venus

number of segments 35

initial state vector

  radius                  1.00000000 (AU)

  radial velocity         0.00000000 (AU/day)

  transverse velocity     1.00000000 (AU/day)

final state vector

  radius                  0.72333100 (AU)

  radial velocity        -0.00000001 (AU/day)

  transverse velocity     1.17579460 (AU/day)


total transfer time     3.52308069 (non-dimensional)

total transfer time   204.80527993 days

total transfer angle  224.78590028 degrees
```





Earth-to-Mars with optical solar sail

This example uses 50 trajectory segments and models a typical non-ideal solar sail.

```
*****************************************
* input data file for ss2d_opt.m
* Earth-to-Mars trajectory - non-ideal sail
*****************************************

number of trajectory segments
50

characteristic acceleration (meters/second**2)
0.001

optical force model coefficient b1 (b1 = 0 = ideal)
0.0864

optical force model coefficient b2 (b2 = 1 = ideal)
0.8272

optical force model coefficient b3 (b3 = 0 = ideal)
-5.45e-3

target planet (1 = Venus, 2 = Mars)
2

initial guess for mission duration (days)
400

lower bound for mission duration (days)
250

upper bound for mission duration (days)
500

initial guess for steering angles (degrees)
20.0

lower bound for steering angles (degrees)
0.0

upper bound for steering angles (degrees)
90.0

departure calendar date initial guess (month, day, year)
1,1,2020
```

The following is the script output for this example.

```
program ss2d_opt - Earth-to-Mars

number of segments 50

initial state vector

  radius                  1.00000000 (AU)

  radial velocity         0.00000000 (AU/day)

  transverse velocity     1.00000000 (AU/day)
```

```
final state vector

  radius                    1.52367998 (AU)

  radial velocity           0.00000000 (AU/day)

  transverse velocity       0.81012702 (AU/day)

total transfer time      7.70083775 (non-dimensional)

total transfer time    447.66849522 days

total transfer angle   271.13778225 degrees

synodic period             779.948575 days
                             2.135383 years

departure calendar date       07-Jul-2020

arrival calendar date         27-Sep-2021
```