John Clouse IMD HW 4 Problem 1

Table of Contents

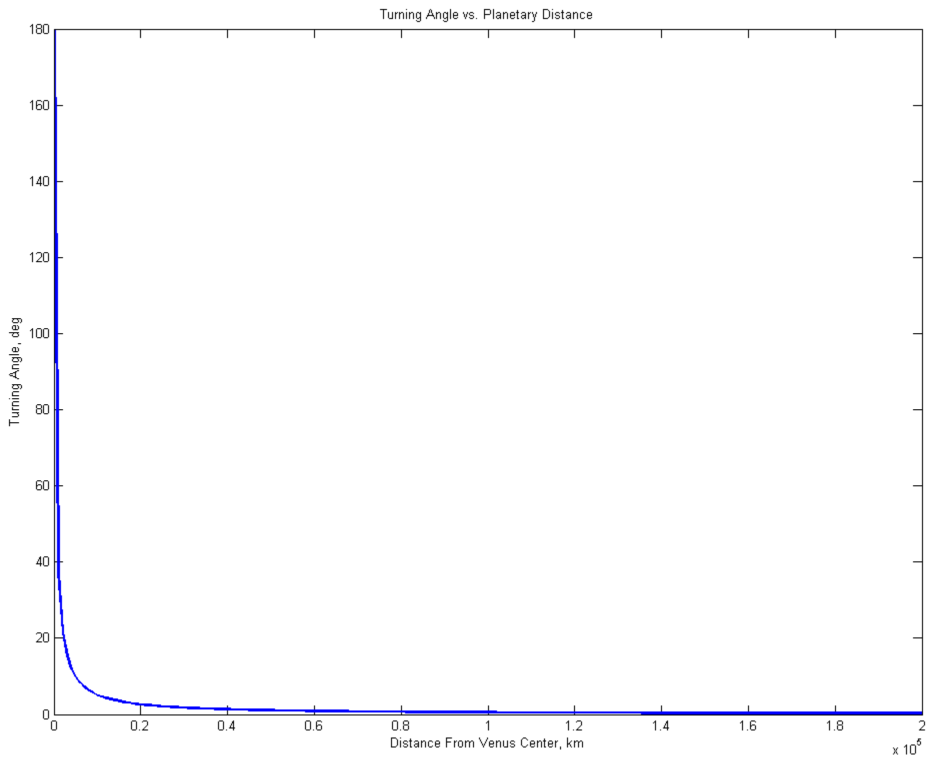
Initialize

```
clearvars -except hw_pub function_list

CelestialConstants

V_spacecraft_wrt_sun=[-10.8559 -35.9372]'; %km/s
v_venus = [15.1945 -31.7927]';%km/s
r_venus = [96948447.3751  46106976.1901]'; %km
mu_sun=1.32712440018e11; %km3/s2
mu_Venus=3.257e5; %km3/s2
R_Venus=    6052; %km

specific_energy_pre = norm(V_spacecraft_wrt_sun)^2/2-mu_sun/norm(r_venus);
fprintf('Heliocentric Energy: %3f km^2/s^2\n',specific_energy_pre);
```

```
Heliocentric Energy: -531.548249 km^2/s^2
```

b) turn angle

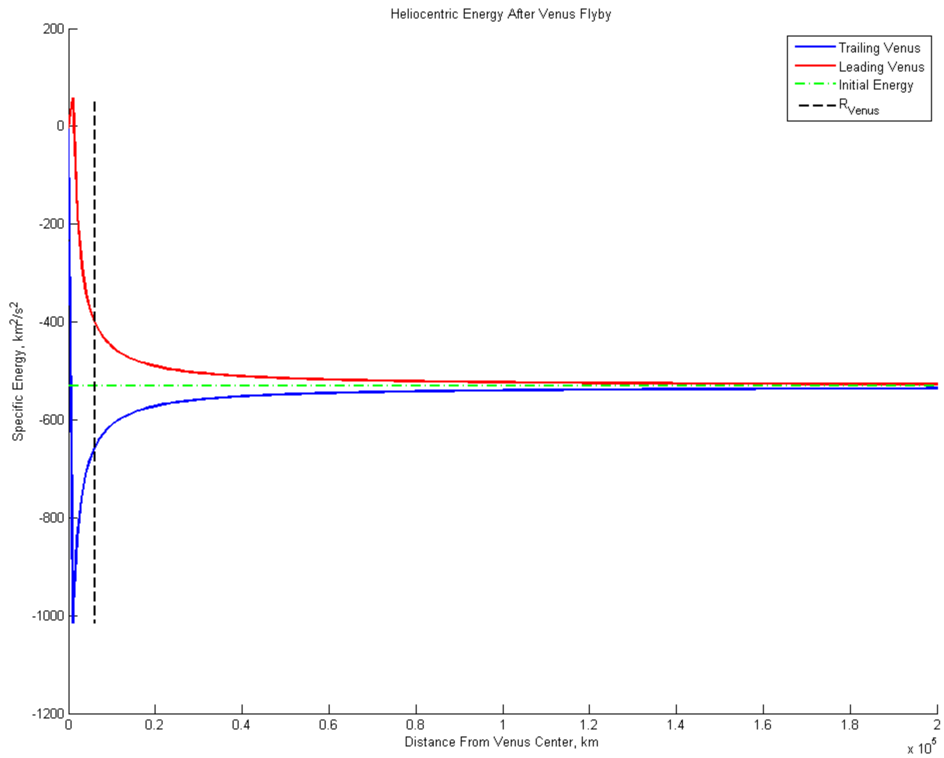
```
v_inf = V_spacecraft_wrt_sun - v_venus;
v_inf_2 = norm(v_inf)^2;
num_pts = 200;
rp = linspace(00,200000,num_pts);
turn_angle_store = zeros(1,num_pts);
for ii = 1:num_pts
    turn_angle_store(ii) = pi - 2*acos( 1/(1+v_inf_2*rp(ii)/mu_Venus));
end

figure('Position', hw_pub.figPosn);
plot(rp,turn_angle_store*180/pi,'LineWidth',hw_pub.lineWidth);
title('Turning Angle vs. Planetary Distance')
xlabel('Distance From Venus Center, km')
ylabel('Turning Angle, deg')
```



c) heliocentric energy

```
energy_leading_pass = zeros(1,num_pts);
energy_trailing_pass = zeros(1,num_pts);
for ii = 1:num_pts
    turn_DCM = Euler2DCM('3',turn_angle_store(ii));
    energy_leading_pass(ii) = norm(turn_DCM(1:2,1:2)*v_inf+v_venus)^2/2 ...
        -mu_sun/norm(r_venus);
    turn_DCM = Euler2DCM('3',-turn_angle_store(ii));
    energy_trailing_pass(ii) = norm(turn_DCM(1:2,1:2)*v_inf+v_venus)^2/2 ...
        -mu_sun/norm(r_venus);
end

figure('Position', hw_pub.figPosn);
hold on
plot(rp,energy_leading_pass,'b','LineWidth',hw_pub.lineWidth);
plot(rp,energy_trailing_pass,'r','LineWidth',hw_pub.lineWidth);
plot([rp(1) rp(end)], ...
    [specific_energy_pre specific_energy_pre],'g-.',...
    'LineWidth',hw_pub.lineWidth);
plot([R_Venus R_Venus], ...
    [max(energy_trailing_pass) min(energy_leading_pass)],'k--',...
    'LineWidth',hw_pub.lineWidth);
legend('Trailing Venus', 'Leading Venus', 'Initial Energy', 'R_{Venus}')
title('Heliocentric Energy After Venus Flyby')
```

```
xlabel('Distance From Venus Center, km')
ylabel('Specific Energy, km^2/s^2')
```



d) What does the plot of energy vs. flyby closest approach tell you?

The spacecraft cannot exit the solar system from just this flyby, as it would have to go below the radius of Venus to do so. In addition, the energy for both leading and trailing the planet asymptotically approach the original energy. This means that the furthest approaches don't add much to the heliocentric energy, so trajectories that need to raise their aphelian should fly closer.

Published with MATLAB® R2013b

John Clouse IMD HW 4 Problem 2

Table of Contents

Initialize

```
clearvars -except hw_pub function_list

V_inf_in=[-5.19425  5.19424 -5.19425];%(km/s)
V_inf_out=[-8.58481  1.17067 -2.42304];%(km/s)
mu_earth=3.986004415e5; %km3/s2
```

Find the B-plane

```
S_hat = V_inf_in/norm(V_inf_in);
k_hat = [0 0 1]';

T_hat = cross(S_hat,k_hat)/norm(cross(S_hat,k_hat));
R_hat = cross(S_hat, T_hat);
h_hat = cross(V_inf_in,V_inf_out)/norm(cross(V_inf_in,V_inf_out));
B_hat = cross(S_hat, h_hat);
```

The flyby parameters

```
psi = acos(dot(V_inf_in,V_inf_out)/norm(V_inf_in)/norm(V_inf_out));
rp = mu_earth/(norm(V_inf_in)^2)*(1/cos((pi-psi)/2)-1);

b = mu_earth/(norm(V_inf_in)^2)*sqrt((1+(norm(V_inf_in)^2)*rp/mu_earth)^2 ...
    - 1);
theta = atan2(dot(b*B_hat, R_hat),dot(b*B_hat, T_hat));
```

Results

```
fprintf('rp = %.3f km\n',rp);
fprintf('psi = %.3f deg\n', psi*180/pi);
fprintf('b = %.3f km\n', b);
fprintf('theta = %.3f deg\n',theta*180/pi);
```

```
rp = 9975.868 km
psi = 38.598 deg
b = 14063.156 km
theta = 20.922 deg
```

Published with MATLAB® R2013b

John Clouse IMD HW 4 Problem 3

Table of Contents

Initialize

```
clearvars -except hw_pub function_list

CelestialConstants;

JD_launch = 2447807.5;
JD_Venus = 2447932.5;
JD_Earth1 = 2448235.5;
```

Planet positions and velocities, V_inf in and out

```
[r_earth_L,~] = MeeusEphemeris(Earth, JD_launch,Sun);
[r_venus, v_venus] = MeeusEphemeris(Venus,JD_Venus,Sun);
[r_earth_1,~] = MeeusEphemeris(Earth, JD_Earth1,Sun);

[~,v_in] = lambert(r_earth_L, r_venus,(JD_Venus-JD_launch)*day2sec,1,Sun);
[v_out,~] = lambert(r_venus, r_earth_1,(JD_Earth1-JD_Venus)*day2sec,-1,Sun);

v_inf_in = v_in - v_venus;
v_inf_out = v_out - v_venus;
```

Flyby params

```
psi = acos(dot(v_inf_in,v_inf_out)/norm(v_inf_in)/norm(v_inf_out));
rp = Earth.mu/(norm(v_inf_in)^2)*(1/cos((pi-psi)/2)-1);
hp = rp - Venus.R; %km
```

Energy

```
energy_pre_flyby = norm(v_in)^2/2 - Sun.mu/norm(r_venus);
energy_post_flyby = norm(v_out)^2/2 - Sun.mu/norm(r_venus);
percent_change = (energy_post_flyby - energy_pre_flyby)...
    /abs(energy_pre_flyby)*100;
```

Results

The v-infinities do not match exactly because the events aren't precisely targeted. True event JDs can be found to make them line up even better. The altitude of closes approach to Venus is shown below. The energy changed by 15%, making a trajectory with a higher aphelion.

```
fprintf('Closest approch altitude: %.3f km\n',hp);
fprintf('Heliocentric energy before: %.3f km^2/s^2\n',energy_pre_flyby);
fprintf('Heliocentric energy after: %.3f km^2/s^2\n',energy_post_flyby);
```

```
Closest approch altitude: 17105.222 km
Heliocentric energy before: -528.482 km^2/s^2
Heliocentric energy after: -448.317 km^2/s^2
```

Published with MATLAB® R2013b

CelestialConstants

Table of Contents

Description

All sorts of constants for orbital mechanics purposes

```
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app
```

Celestial units

```
au2km = 149597870.7;
```

Physical constants

```
day2sec = 86400; % sec/day
speed_of_light = 299792458; %m/s
```

Earth

```
Earth.name = 'Earth';
Earth.mu = 3.986004415e5; %km3/s2
Earth.R = 6378; %km
Earth.a = 149598023; %km
Earth.spin_rate = 7.2921158553e-05; %rad/s
Earth.flattening = 1/298.25722; %WGS-84
Earth.oblate_ecc = 0.081819221456; %WGS-84
Earth.J2 = 0.0010826267;
Earth.P_days = 365.2421897; %days
Earth.P_years = 0.99997862; %days
Earth.m = 5.9742e24; %kg
% Meeus ephemeris parameters
Earth.Meeus.J200.L = [100.466449 35999.3728519 -0.00000568 0.0]; %deg
Earth.Meeus.J200.a = 1.000001018*au2km; %km
```

```
Earth.Meeus.J200.e = [0.01670862 -0.000042037 -0.0000001236 0.00000000004];
Earth.Meeus.J200.i = [0 0.0130546 -0.00000931 -0.000000034]; % deg
Earth.Meeus.J200.RAAN = [174.873174 -0.2410908 0.00004067 -0.000001327]; %deg
Earth.Meeus.J200.Pi = [102.937348 0.3225557 0.00015026 0.000000478]; %deg
```

Moon

```
Moon.name = 'Moon';
Moon.R = 1738.0; %km
Moon.J2 = 0.0002027;
Moon.P_days = 27.321582; %days
Moon.mu = 4902.799; %km3/s2
Moon.m = 7.3483e22; %kg
Moon.a = 384400; %km
```

Sun

```
Sun.mu = 1.32712428e11; %km3/s2
Sun.m = 1.9891e30; %kg
```

Mercury

```
Mercury.name = 'Mercury';
Mercury.R = 2439.0; %km
Mercury.J2 = 0.00006;
Mercury.P_days = 87.9666; %days
Mercury.mu = 2.2032e4; %km3/s2
```

Venus

```
Venus.name = 'Venus';
Venus.a = 108208601; %km
Venus.R = 6052.0; %km
Venus.J2 = 0.000027;
Venus.P_days = 224.6906; %days
Venus.mu = 3.257e5; %km3/s2
Venus.m = 4.869e24; %km
Venus.Meeus.J200.L = [181.979801 58517.8156760 0.00000165 -0.000000002];%deg
Venus.Meeus.J200.a = 0.72332982*au2km; % km
Venus.Meeus.J200.e = [0.00677188 -0.000047766 0.0000000975 0.00000000044];
Venus.Meeus.J200.i = [3.394662 -0.0008568 -0.00003244 0.000000010];%deg
Venus.Meeus.J200.RAAN = [76.679920 -0.2780080 -0.00014256 -0.000000198];%deg
Venus.Meeus.J200.Pi = [131.563707 0.0048646 -0.00138232 -0.000005332];%deg
```

Mars

```
Mars.name = 'Mars';
Mars.a = 227939186; %km
Mars.R = 3397.2; %km
Mars.J2 = 0.001964;
Mars.P_days = 686.9150; %days
```

```
Mars.mu = 4.305e4; %km3/s2
Mars.m = 6.4191e23; %kg

% Meeus ephemeris parameters
Mars.Meeus.J200.L = [355.433275 19140.2993313 0.00000261 -0.000000003]; %deg
Mars.Meeus.J200.a = 1.523679342*au2km; %km
Mars.Meeus.J200.e = [0.09340062 0.000090483 -0.0000000806 -0.00000000035];
Mars.Meeus.J200.i = [1.849726 -0.0081479 -0.00002255 -0.000000027]; %deg
Mars.Meeus.J200.RAAN = [49.558093 -0.2949846 -0.00063993 -0.000002143]; %deg
Mars.Meeus.J200.Pi = [336.060234 0.4438898 -0.00017321 0.000000300]; %deg
```

Jupiter

```
Jupiter.name = 'Jupiter';
Jupiter.a = 778298361; %km
Jupiter.R = 71492; %km
Jupiter.J2 = 0.01475;
Jupiter.P_years = 11.856525; %days
Jupiter.P_days = Jupiter.P_years/Earth.P_years*Earth.P_days; %days
Jupiter.mu = 1.268e8; %km3/s2
Jupiter.m = 1.8988e27; %kg
Jupiter.Meeus.J200.L = [34.351484 3034.9056746 -0.00008501 0.000000004 ];
Jupiter.Meeus.J200.a = [5.202603191 0.0000001913 ]*au2km;
Jupiter.Meeus.J200.e = [0.04849485 0.000163244 -0.0000004719 -0.00000000197 ];
Jupiter.Meeus.J200.i = [1.303270 -0.0019872 0.00003318 0.000000092 ];
Jupiter.Meeus.J200.RAAN = [100.464441 0.1766828 0.00090387 -0.000007032 ];
Jupiter.Meeus.J200.Pi = [14.331309 0.2155525 0.00072252 -0.000004590 ];
```

Saturn

```
Saturn.name = 'Saturn';
Saturn.a = 1429394133; %km
Saturn.R = 60268; %km
Saturn.J2 = 0.01645;
Saturn.P_years = 29.423519; %days
Saturn.P_days = Saturn.P_years/Earth.P_years*Earth.P_days; %days
Saturn.mu = 3.794e7; %km3/s2
Saturn.m = 5.685e26; %kg
```

Uranus

```
Uranus.name = 'Uranus';
Uranus.R = 25559; %km
Uranus.J2 = 0.012;
Uranus.P_years = 83.747406; %days
Uranus.P_days = Uranus.P_years/Earth.P_years*Earth.P_days; %days
Uranus.mu = 5.794e6; %km3/s2
```

Neptune

```
Neptune.name = 'Neptune';
```

```
Neptune.R = 24764; %km
Neptune.J2 = 0.004;
Neptune.P_years = 163.7232045; %days
Neptune.P_days = Neptune.P_years/Earth.P_years*Earth.P_days; %days
Neptune.mu = 6.809e6; %km3/s2
```

Published with MATLAB® R2013b

```
function DCM = Euler2DCM( seq_string, angle_vector )
%Euler2DCM Turn an Euler Angle set into a DCM
%   Angle vector in radians
fcnPrintQueue(mfilename('fullpath'))

DCM = eye(3);
%get the trig functions
num_rot = length(seq_string);
c = zeros(num_rot,1);
s = zeros(num_rot,1);

for idx = 1:num_rot
c(idx) = cos(angle_vector(idx));
s(idx) = sin(angle_vector(idx));
end


for idx = num_rot:-1:1
    if strcmp(seq_string(idx),'1')
        M = [1 0 0; 0 c(idx) s(idx); 0 -s(idx) c(idx)];
        DCM = DCM*M;
    elseif strcmp(seq_string(idx),'2')
        M = [c(idx) 0 -s(idx); 0 1 0; s(idx) 0 c(idx)];
        DCM = DCM*M;
    elseif strcmp(seq_string(idx),'3')
        M = [c(idx) s(idx) 0; -s(idx) c(idx) 0; 0 0 1];
        DCM = DCM*M;
    else
        fprintf('%s is not a valid axis\n', seq_string(idx))
    end
end


end
```

Published with MATLAB® R2013b

```
function [ r, v ] = MeeusEphemeris( planet, JD , Sun)
%MeeusEphemeris Calculate planetary ephemeris. Works with
%CelestialConstants.m file
%   Outputs PV in km, km/s
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

T = (JD - 2451545)/36525;

if length(planet.Meeus.J200.a) == 1
    a = planet.Meeus.J200.a;%*au2km;
else
    T_pow = 1;
    a = 0;
    for ii = 1:length(planet.Meeus.J200.a)
        a = a + planet.Meeus.J200.a(ii)*T_pow;
        T_pow = T_pow*T;
    end
end

L = planet.Meeus.J200.L(1) ...
    + planet.Meeus.J200.L(2)*T ...
    + planet.Meeus.J200.L(3)*T*T ...
    + planet.Meeus.J200.L(4)*T*T*T;
e = planet.Meeus.J200.e(1) ...
    + planet.Meeus.J200.e(2)*T ...
    + planet.Meeus.J200.e(3)*T*T ...
    + planet.Meeus.J200.e(4)*T*T*T;

i = planet.Meeus.J200.i(1) ...
    + planet.Meeus.J200.i(2)*T ...
    + planet.Meeus.J200.i(3)*T*T ...
    + planet.Meeus.J200.i(4)*T*T*T;

RAAN = planet.Meeus.J200.RAAN(1) ...
    + planet.Meeus.J200.RAAN(2)*T ...
    + planet.Meeus.J200.RAAN(3)*T*T ...
    + planet.Meeus.J200.RAAN(4)*T*T*T;

Pi = planet.Meeus.J200.Pi(1) ...
    + planet.Meeus.J200.Pi(2)*T ...
    + planet.Meeus.J200.Pi(3)*T*T ...
    + planet.Meeus.J200.Pi(4)*T*T*T;

% Convert everything to radians!
L = L*pi/180;
i = i*pi/180;
RAAN = RAAN*pi/180;
Pi = Pi*pi/180;

w = Pi - RAAN;

M = L - Pi;
```

```
e2 = e*e;
e3 = e*e2;
e4 = e*e3;
e5 = e*e4;

C_cen = (2*e-e3/4+5/96*e5)*sin(M) + (5/4*e2-11/24*e4)*sin(2*M) ...
    + (13/12*e3-43/64*e5)*sin(3*M) + 103/96*e4*sin(4*M) ...
    + 1097/960*e5*sin(5/M);

f = M + C_cen;

[r, v] = OE2cart(a, e, i, RAAN, w, f, Sun.mu);

end
```

Published with MATLAB® R2013b

```
function [r, v ] = OE2cart( a,e,i,RAAN,w,f,mu)
%cart2OE return classical orbital elements from cartesian coords
% Only valid for e < 1
% units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% First find r,v in the perifocal coord system.
p = a*(1-e*e);
r_pqw = [p*cos(f);p*sin(f);0]/(1+e*cos(f));
v_pqw = [-sqrt(mu/p)*sin(f); sqrt(mu/p)*(e+cos(f));0];

r = Euler2DCM('313', -[w,i,RAAN])*r_pqw;
v = Euler2DCM('313', -[w,i,RAAN])*v_pqw;
```

Published with MATLAB® R2013b

```
function [vi, vf, psi] = lambert(ri_vec, rf_vec, dt, DM, Sun, psi_in)
%lambert Solve lambert problem using universal variables method
%   Output initial and final velocities given respective position vectors
%   Inputs in km, Results in km/s
%   DM = Direction of Motion (short way or long way)

fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

tol = 1e-6;

ri = norm(ri_vec);
rf = norm(rf_vec);

cos_df = dot(ri_vec, rf_vec)/(ri*rf);

A = DM*sqrt(ri * rf * (1+cos_df));

if nargin < 6
    psi = 0;
else
    psi = psi_in;
end
c2 = 1/2;
c3 = 1/6;
psi_up_i = 4*pi*pi + psi; % Doubled from Vallado for higher TOF
psi_low_i = -4*pi; % Doubled from Vallado for lower TOF

dt_calc = 0;

first_pass = true;
% while abs(dt_calc-dt) > tol
    if first_pass
        psi_up = psi_up_i;
        psi_low = psi_low_i;
        first_pass = false;
%     elseif psi_up < 0% hit the lower boundary
%         psi_up_i = psi_low_i; % Upper bound is last time's lower bound
%         psi_low_i = 4*psi_low_i; % decrease lower bound
%         psi_up = psi_up_i;
%         psi_low = psi_low_i;
%     elseif psi_low > 0 %hit upper boundary
%         psi_low_i = psi_up_i; % lower bound is last time's upper
%         psi_up_i = 4*psi_up_i; % increase upper
%         psi_up = psi_up_i;
%         psi_low = psi_low_i;
    end


    while abs(dt_calc-dt) > tol
        y = ri + rf + A*(psi*c3-1)/sqrt(c2);
        if A > 0 && y < 0
            while y < 0
```

```
                psi = psi + 0.1;
                y = ri + rf + A*(psi*c3-1)/sqrt(c2);
            end
        end

        X = sqrt(y/c2);
        dt_calc = (X*X*X*c3 + A*sqrt(y))/sqrt(Sun.mu);

%         y_prime = A*(c3-1)/sqrt(c2);
%         psi = psi - y/y_prime;
        if (dt_calc <= dt)
            psi_low = psi;
        else
            psi_up = psi;
        end

%         if abs(psi_up_i - psi_low) < 1e-10 || abs(psi_low_i - psi_up) < 1e-10
%             break; %we hit one of the boundaries
%         end

        psi = (psi_up + psi_low)/2;

        if psi > 1e-6
            c2 = (1-cos(sqrt(psi)))/psi;
            c3 = (sqrt(psi) - sin(sqrt(psi)))/sqrt(psi*psi*psi);
        elseif psi < -1e6
            c2 = (1-cosh(sqrt(psi)))/psi;
            c3 = (sinh(sqrt(-psi)) - sqrt(-psi))/sqrt(-psi*psi*psi);
        else
            c2 = 1/2;
            c3 = 1/6;
        end

        if (psi_up-psi_low) < 1e-10 && abs(dt_calc-dt) > 100
            %Get out of here! fell into a bad minimum.
            fprintf('Lamber solver fell into a bad minimum, returning.\n')
            fprintf('psi = %.3f\n',psi)
            psi = 0;
            vi = zeros(3,1);
            vf = zeros(3,1);
            return
        end
    end

    f = 1-y/ri;
    g_dot = 1-y/rf;
    g = A*sqrt(y/Sun.mu);

    vi = (rf_vec-f*ri_vec)/g;
    vf = (g_dot*rf_vec-ri_vec)/g;
% end


end
```

Published with MATLAB® R2013b

```matlab
function fcnPrintQueue( filename )
global function_list;
if exist('function_list', 'var')
    file_in_list = 0;
    for idx = 1:length(function_list)
        if strcmp(function_list(idx), filename);
            file_in_list = 1;
            break
        end
    end
    if ~file_in_list
        function_list = [function_list, filename];
    end
end
end
```

Published with MATLAB® R2013b