ASE387P.2 Mission Analysis and Design
Homework 4: Orbit Design

Junette Hsin
Masters Student, Aerospace Engineering and Engineering Mechanics, University of Texas, Austin, TX 78712

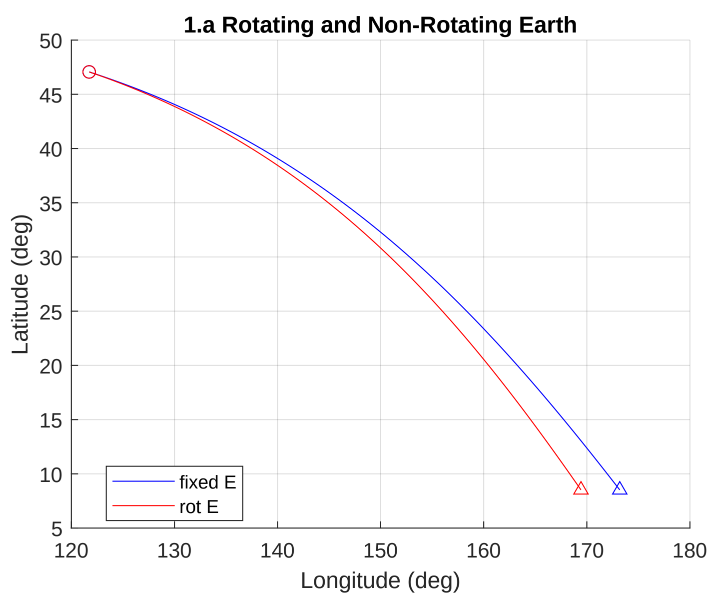
Problem 1: ISS Ground Tracks

A

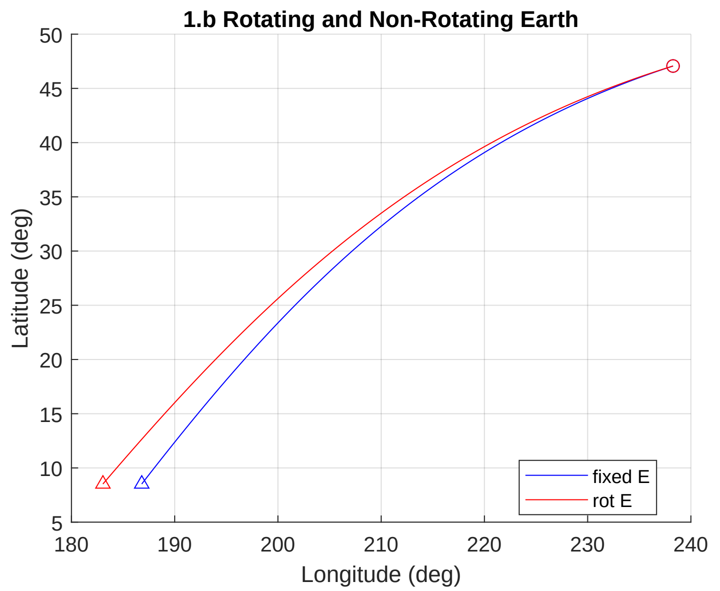
The following parameters were used to initialize the ISS orbit, which were taken from `https://in-the-sky.org/spacecraft_elements.php?id=25544` on April 19:

- $a = 6794.588$ km
- $e = 0.00049$
- $i = 51.627$ deg
- $\omega = 40.1116$ deg
- $\Omega = 0$ deg
- $M = 70.88$

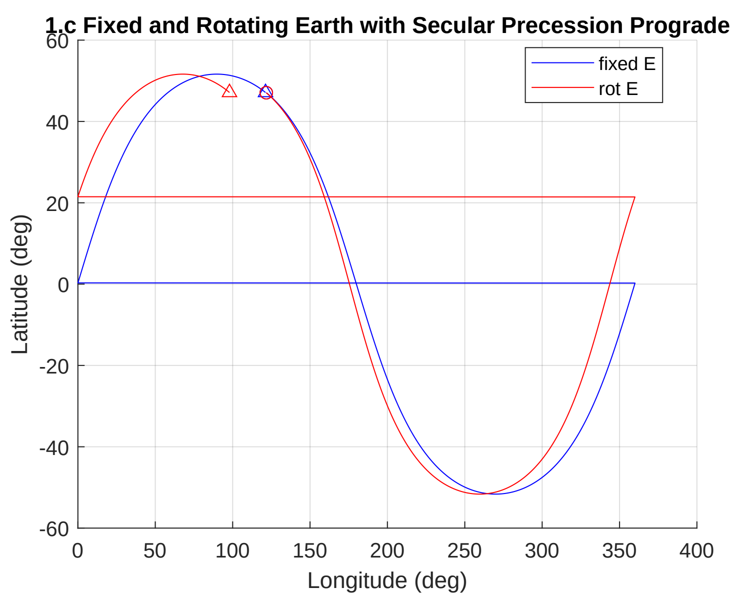
The right ascension of the ascending node, Ω, was set to 0 as stated in the problem statement.



B



C



To derive a formula for predicting the westward drift, first calculate the Draconitic period:

$$T_d = \frac{2\pi}{\dot{\omega} + \dot{M}} \tag{1}$$

Then calculate the rate of secular variation for the ascending node due to J2:

$$\dot{\Omega} = -\frac{3}{2}\bar{n}J_2(\frac{a_e}{\bar{a}})^2\frac{1}{(1-\bar{e}^2)^2}cos\bar{I} \tag{2}$$

$\bar{a}$, $\bar{e}$, and $\bar{I}$ can be taken from the initial ISS orbit parameters, and $\bar{n}$ is the mean motion of the orbit:

$$\bar{n} = \sqrt{\frac{\mu}{\bar{a}^3}} \tag{3}$$

Finally, the change in lambda was calculated as:

$$\Delta\lambda = -(w_E - \dot{\Omega})T_d \tag{4}$$

where w_E is the rotation rate of the Earth. $\Delta\lambda$ came out to **-22.870**°.

The calculated change in drift of the ascending node was taken from the difference between the initial longitude and the final longitude after one revolution, which came out to be **-22.790**°. The calculated drift was within 0.1° of the analytical prediction.

Problem 2: Orbit Design

Continuing from Equation 4, after m revolutions,

$$m\Delta\lambda = -(w_E - \dot{\Omega})T_d m \tag{5}$$

A condition for ground track repeat is that $m\Delta\lambda$ needs to be a multiple (let's say k) of 2π, or

$$k = \frac{m\Delta\lambda}{2\pi} = \frac{m(w_E - \dot{\bar{\Omega}})T_d}{2\pi} = m\frac{T_d}{D_n} \tag{6}$$

For a ground track repeat orbit, the conditions for which the sub-satellite point repeatedly passes a geographical location (ϕ, λ) at regular intervals are given by:

$$(\omega_e - \dot{\bar{\Omega}})D_n = k2\pi \tag{7}$$

$$(\dot{\bar{\omega}} + \dot{\bar{M}})T_d = m2\pi \tag{8}$$

As for frozen orbits, the largest perturbation on gravitational acceleration is due to J2, which is on the order of 10^{-3}. The next-largest perturbation is J3, which is on the order of 10^-6. Equation 2 gives the scalar secular variation on $\dot{\bar{\Omega}}$ due to J2 which grows indefinitely. Equations 9 and 10 give the secular variations due to J2 for $\dot{\bar{\omega}}$ and $\dot{\bar{M}}$:

$$\dot{\bar{\omega}} = -\frac{3}{4}\bar{n}J_2(\frac{a_e}{\bar{a}})^2\frac{1}{(1-\bar{e}^2)^2}(1-5cos^2\bar{I}) \tag{9}$$

$$\dot{\bar{M}} = \bar{n}\left[1 - \frac{3}{4}(\frac{a_e}{\bar{a}})^2 J_2\frac{1}{(1-\bar{e}^2)^{\frac{3}{2}}}(1-3cos^2\bar{I})\right] \tag{10}$$

J2 also leads to a short-period perturbation on all elements, but only semi-major axis is of interest:

$$\Delta a_{SP}(t) = \bar{a}J_2(\frac{a_e}{\bar{a}})^2\left[\left(1-\frac{3}{2}sin^2\bar{I}\right)\left((\frac{\bar{a}}{r})^3 - \frac{1}{(1-\bar{e}^2)^{\frac{3}{2}}}\right) + \frac{3}{2}(\frac{\bar{a}}{r})^3 sin^2\bar{I}cos2(\bar{w}+\bar{f})\right] \tag{11}$$

J3 exerts a long-period perturbation on all elements other than the semi-major axis, but only the eccentricity and perigee are of interest:

$$\Delta e_{LP}(t) = -\frac{1}{2}\frac{J_3}{J_2}\frac{a_e}{\bar{a}}sin\bar{I}sin\bar{w}(t) \tag{12}$$

$$\bar{e}\Delta\omega_{LP}(t) = -\frac{1}{2}\frac{J_3}{J_2}\frac{a_e}{\bar{a}}\frac{1}{(1-\bar{e}^2)}sin\bar{I}\ cos\bar{w}(t) \tag{13}$$

For frozen orbits, the average variation rates of e and w are set to 0. Thus, the mean argument of perigee should be around 90 °.

$$w = 90^\circ \tag{14}$$

An orbit is sun-synchronous when the precession rate equals the mean motion of the Earth around the Sun. the Ω nodal rate needs to match the average rate of the Sun's motion projected onto the Earth's equator:

$$\frac{d\Omega}{dt} = \dot{\bar{\Omega}} = \frac{360^\circ}{365.242\ days/year} = 0.9856\ {}^\circ/day \tag{15}$$

The angular precession for an Earth orbiting satellite is given by Equation 2. One can reform Equation 2 as a formula for inclination:

$$\bar{I} = cos^{-1}\left[-\frac{2}{3}\frac{d\Omega}{dt}\frac{1}{J_2\bar{n}}\Big(\frac{\bar{a}(1-\bar{e}^2)}{R_E}\Big)^2\right] \tag{16}$$

The process for determining the mean elements $\bar{a}$, $\bar{e}$, and $\bar{I}$ involves making initial guesses and then minimizing the misclosure rate through an optimization routine:

$$\epsilon = m(w_E - \dot{\bar{\Omega}}) - k(\dot{\bar{\omega}} - \dot{\bar{M}}) \tag{17}$$

Initial computed states for position and velocity will be integrated and iterated until a sun-synchronous, frozen, and repeated ground track orbit is found.

B

Problem 3

The calculations for this section were taken from the paper **Five Special Types of Orbits Around Mars** (2010 Liu, Baoyin, and Ma). A frozen orbit is possible on Mars. Mars' J2 = 1.95545e-3 and J3 = 3.14498e-5. In comparison, the Earth J2 = 1.08263e-3 and J3 = -2.53266e-6. The same positive sign between J2 and J3 must be accounted for to prevent the desired eccentricity from becoming negative. For Earth, ω is set to 90 °. For Mars, ω must be set to around 270 °.

The average variation rate of e from Equation 12 is given by:

$$\dot{\bar{e}} = \frac{3nJ_3R_p^{\bar{3}}sin\bar{I}}{4a^3(1-e^2)^2}\Big(\frac{5}{2}sin^2\bar{I} - 2\Big)cos\omega \tag{18}$$

And from Equation 13, the average variation rate of ω is given by:

$$\dot{\bar{\omega}} = \frac{3\bar{n}J_2R_p^2}{2a^2(1-e^2)^2}\left[\Big(2-\frac{5}{2}sin^2\bar{I}\Big)\Big(1+\frac{J_3R_p}{2J_2a(1-e^2)}\big(\frac{sin^2\bar{I}-\bar{e}^2cos^2\bar{I}}{sin\bar{I}}\big)\frac{sin\omega}{\bar{e}}\Big) + \frac{3J_2R_p^2}{2\bar{a}^2(1-e^2)^2}D\right] \tag{19}$$

where

$$D = (4+\frac{7}{12}\bar{e}^2+2\sqrt{1-e^2}) - sin^2\bar{I}(\frac{103}{102}+\frac{3}{8}\bar{e}^2+\frac{11}{2}\sqrt{1-\bar{e}^2}) + sin^4\bar{I}(\frac{215}{418}-\frac{15}{32}\bar{e}^2+\frac{15}{4}\sqrt{1-\bar{e}^2}) + ...H.O.T. \tag{20}$$

H.O.T. stands for higher order terms than J3. As stated in Problem 2, for frozen orbits, the average variation rates of e and ω are set to 0.

Solving the above equations yields the following for e and ω:

$$e = \frac{\frac{J_3R_p}{2J_2\bar{a}}sin\bar{I}sin\omega}{1-\frac{3J_2R_p^2E}{\bar{a}^2(5sin^2\bar{I}-4)}} \tag{21}$$

$$\omega = 270^\circ \tag{22}$$

Appendix

MATLAB code

```

        %% Junette Hsin

% 0 = no plot. 1 = plot animation. 2 = plot all at once
plot_option = 2;

% Earth
Earth.a0 =       1.00000261;
Earth.da =       0.00000562;
Earth.e0 =       0.01671123;
Earth.de =      -0.00004392;
Earth.I0 =      -0.00001531;
Earth.dI =      -0.01294668;
Earth.L0 =     100.46457166;
Earth.dL =   35999.37244981;
Earth.wbar0 = 102.93768193;
Earth.dwbar =    0.32327364;
Earth.Omega0 =   0;
Earth.dOmega =   0;


%% Problem 1

% (Ground Tracks) Consider the International Space Station (look up its orbital elements)
% in a Keplerian orbit. Assume that at t = 0, the ISS, its orbital ascending node, and the
% Greenwich Meridian are coincident. Note that all references to node crossings and ground
% tracks are from a viewpoint attached to the surface of the Earth.

% Draw and compare the ground tracks for 15-minute duration in two cases:
% i. A non-rotating Earth
% ii. A uniformly rotating Earth.

% t = 0, ascending node and Greenwich Meridian coincident

% ISS OEs (from https://in-the-sky.org/spacecraft_elements.php?id=25544)
% e0 = 0.00048;
% i0 = 51.644 * pi/180;     % deg --> rad
% w0 = 30.4757 * pi/180;    % deg --> rad
% w0 = 0;
% O0 = 0 * pi/180;          % deg --> rad
% M0 = 39.7178 * pi/180;    % deg --> rad (should be true anomaly)
% M0 = (2*pi - w0);         % rad (should be true anomaly)
% M0 = 0;

e0 = 0.00049;
i0 = 51.6427 * pi/180;
w0 = 40.1116 * pi/180;
O0 = 0;
M0 = 70.88 * pi/180;

% mean motion --> semimajor axis
n = 15.50094 / 86400 * (2*pi);   % rev/day --> rad/s
mu_E_m3 = 3.986004418e14;  % m^3/s^2
mu_E_km3 = mu_E_m3 * ( 1e-3 )^3 ;
a0 = (mu_E_km3 / n^2)^(1/3);

R_E = 6378.1370;        % km
w_E = 7.292115e-5;      % rad/s

oe0 = [a0; e0; i0; w0; O0; M0];
rv0 = rvOrb.orb2rv(oe0, mu_E_km3);

% set ode45 params
rel_tol = 1e-10;          % 1e-14 accurate; 1e-6 coarse
```

```
abs_tol = 1e-10;
options = odeset('reltol', rel_tol, 'abstol', abs_tol );

% propagate orbit
[t, rv_a] = ode45(@fn.EOM, [0 : 15*60], rv0, options);


% -------------------------------------------------------------------------
% problem 1.a.i

lla = [];
lla_1_a = [];
for i = 1:length(rv_a)

%     lla(i,:) = ecef2lla(rv(i,1:3));
    lla_1_a(i,:) = ecef2lla_1(rv_a(i,:)', R_E, mu_E_km3);
    lla_1_a(i, 1:2) = lla_1_a(i, 1:2) * 180/pi;

end
% fn.plot3_xyz(rv);

% -------------------------------------------------------------------------
% problem 1.a.ii

% Axis 3 rotation matrix
[lla_rot_a, rv_rot_a] = lla_rv_rot(t, rv_a, w_E, R_E, mu_E_km3);


% -------------------------------------------------------------------------
% PLOT
fname = '1.a Rotating and Non-Rotating Earth';
plot_option = 2;
plot_gt(plot_option, fname, lla_1_a, lla_rot_a)
fn.savePDF(gcf)


%% problem 1.b

i0 = (180 - 51.644) * pi/180;    % deg --> rad

% mean motion --> semimajor axis
n = 15.50094 / 86400 * (2*pi);   % rev/day --> rad/s
mu_E_m3 = 3.986004418e14;  % m^3/s^2
mu_E_km3 = mu_E_m3 * ( 1e-3 )^3 ;
a0 = (mu_E_km3 / n^2)^(1/3);

R_E = 6378.1370;        % km
w_E = 7.292115e-5;      % rad/s

oe0 = [a0; e0; i0; w0; O0; M0];
rv0 = rvOrb.orb2rv(oe0, mu_E_km3);

% set ode45 params
rel_tol = 1e-10;         % 1e-14 accurate; 1e-6 coarse
abs_tol = 1e-10;
options = odeset('reltol', rel_tol, 'abstol', abs_tol );

% propagate orbit
[t, rv_b] = ode45(@fn.EOM, [0 : 15*60], rv0, options);


% -------------------------------------------------------------------------
% Problem 1.b.i

lla = [];
lla_1_b = [];
for i = 1:length(rv_b)

```

```
%      lla_1_b(i,:) = ecef2lla(rv(i,1:3));
     lla_1_b(i,:) = ecef2lla_1(rv_b(i,:)', R_E, mu_E_km3);
     lla_1_b(i, 1:2) = lla_1_b(i, 1:2) * 180/pi;

end
% fn.plot3_xyz(rv);


% ------------------------------------------------------------------------
% problem 1.b.ii

% Axis 3 rotation matrix
[lla_rot_b, rv_rot_b] = lla_rv_rot(t, rv_b, w_E, R_E, mu_E_km3);


% ------------------------------------------------------------------------
% PLOT
fname = '1.b Rotating and Non-Rotating Earth';
plot_option = 2;
plot_gt(plot_option, fname, lla_1_b, lla_rot_b)
fn.savePDF(gcf)


%% problem 1.c.i

% PROGRADE

% Constants
mu = mu_E_km3;
J2 = 1.082e-3;

% O Precession Calcs
Odot = -(3/2)*n*(R_E / a0)^2 * J2 * (1/(1-e0^2)^(1/2)) * cos(i0); % O precession
sprintf('Odot precession: %.3f', Odot)

% ISS OEs (from https://in-the-sky.org/spacecraft_elements.php?id=25544)
i0 = 51.644 * pi/180;    % deg --> rad
% M0 = 0;

oe0 = [a0; e0; i0; w0; O0; M0];
rv0 = rvOrb.orb2rv(oe0, mu_E_km3);

% set ode45 params
rel_tol = 1e-10;         % 1e-14 accurate; 1e-6 coarse
abs_tol = 1e-10;
options = odeset('reltol', rel_tol, 'abstol', abs_tol );

% propagate orbit
T = 2*pi*sqrt(a0^3/mu_E_km3);
[t, rv_c] = ode45(@fn.EOM_J2, [0 : T], rv0, options);

% final oe
oef = rvOrb.rv2orb(rv_c(end,:), mu_E_km3);
Odot = oe0(5) - oef(5);

lla = [];
lla_1_c = [];
for i = 1:length(rv_c)

%      lla(i,:) = ecef2lla(rv(i,1:3));
     lla_1_c(i,:) = ecef2lla_1(rv_c(i,:)', R_E, mu_E_km3);
     lla_1_c(i, 1:2) = lla_1_c(i, 1:2) * 180/pi;

end
% fn.plot3_xyz(rv);


% ------------------------------------------------------------------------
```

```
% Axis 3 rotation matrix
[lla_rot_c, rv_rot_c] = lla_rv_rot(t, rv_c, w_E, R_E, mu_E_km3);


% -----------------------------------------------------------------------
% PLOT

fname = '1.c Fixed and Rotating Earth with Secular Precession Prograde';
plot_option = 2;
plot_gt(plot_option, fname, lla_1_c, lla_rot_c)
fn.savePDF(gcf)

% -----------------------------------------------------------------------
% RETROGRADE

% Constants
mu = mu_E_km3;
J2 = 1.082e-3;

% O Precession Calcs
Odot = -(3/2)*n*(R_E / a0)^2 * J2 * (1/(1-e0^2)^(1/2)) * cos(i0); % O precession
sprintf('Odot precession: %.3f', Odot)

% ISS OEs (from https://in-the-sky.org/spacecraft_elements.php?id=25544)
i0 = (180 - 51.644) * pi/180;    % deg --> rad

oe0 = [a0; e0; i0; w0; O0; M0];
rv0 = rvOrb.orb2rv(oe0, mu_E_km3);

% set ode45 params
rel_tol = 1e-10;          % 1e-14 accurate; 1e-6 coarse
abs_tol = 1e-10;
options = odeset('reltol', rel_tol, 'abstol', abs_tol);

% propagate orbit
T = 2*pi*sqrt(a0^3/mu_E_km3);
[t, rv_c] = ode45(@fn.EOM_J2, [0 : T], rv0, options);

% final oe
oef = rvOrb.rv2orb(rv_c(end,:), mu_E_km3);
Odot = oe0(5) - oef(5);

lla = [];
lla_1_c = [];
for i = 1:length(rv_c)

%     lla(i,:) = ecef2lla(rv(i,1:3));
    lla_1_c(i,:) = ecef2lla_1(rv_c(i,:)', R_E, mu_E_km3);
    lla_1_c(i, 1:2) = lla_1_c(i, 1:2) * 180/pi;

end
% fn.plot3_xyz(rv);


% -----------------------------------------------------------------------
% Axis 3 rotation matrix
[lla_rot_c, rv_rot_c] = lla_rv_rot(t, rv_c, w_E, R_E, mu_E_km3);


% -----------------------------------------------------------------------
% PLOT

fname = '1.c Fixed and Rotating Earth with Secular Precession Retrograde ';
plot_option = 2;
plot_gt(plot_option, fname, lla_1_c, lla_rot_c)
fn.savePDF(gcf)

%% Problem 1.c.ii
```

```
oe_c0 = rvOrb.rv2orb(rv0, mu_E_km3);
dt = 1;

Odot = -3/2 * n * J2 * ( R_E / norm(rv0(1:3)) )^2 * 1/( 1-e0^2 )^2 * cos(i0);
wdot = -3/4 * n * (R_E / norm(rv0(1:3)))^2 * J2 * (1 - 5*cosd(i0)^2) / (1-e0^2)^2;
Mdot = n * (1-3/4*(R_E / norm(rv0(1:3)))^2 * J2 * (1 - 3*cosd(i0)^2) / (1-e0^2)^(3/2));

oe_c = oe_c0;
rv_c2 = rv0';
for i = 1 : T

    oe = oe_c0;

    % augment mean anomaly
    M0 = oe_c0(6);
    M = M0 + Mdot * (dt*i);
    M = mod(M, 2*pi);
    oe(6) = M;

    % augment RAAN
    O0 = oe_c0(5);
    O = O0 + Odot * (dt*i);
    O = mod(O, 2*pi);
    oe(5) = O;

    % augment perigee
    w0 = oe_c0(4);
    w = w0 + wdot * (dt*i);
    w = mod(w, 2*pi);
    oe(4) = w;

    oe_c(i+1,:) = oe;
    rv_c2(i+1,:) = rvOrb.orb2rv(oe, mu_E_km3);

end

lla = [];
lla_1_c2 = [];
for i = 1:length(rv_c2)

%     lla(i,:) = ecef2lla(rv(i,1:3));
    lla_1_c2(i,:) = ecef2lla_1(rv_c2(i,:)', R_E, mu_E_km3);
    lla_1_c2(i, 1:2) = lla_1_c2(i, 1:2) * 180/pi;

end



% -----------------------------------------------------------------------
% Axis 3 rotation matrix
[lla_rot_c2, rv_rot_c2] = lla_rv_rot(t, rv_c2, w_E, R_E, mu_E_km3);

for i = 1:length(rv_rot_c2)
%     lla(i,:) = ecef2lla(rv(i,1:3));
    lla_rot_c2(i,:) = ecef2lla_1(rv_rot_c2(i,:)', R_E, mu_E_km3);
    lla_rot_c2(i, 1:2) = lla_rot_c2(i, 1:2) * 180/pi;

end

% fn.plot3_xyz(rv);
% -----------------------------------------------------------------------
% PLOT

fname = '1.c Fixed and Rotating Earth with Secular Precession';
plot_option = 2;
plot_gt(plot_option, fname, lla_1_c2, lla_rot_c2)
fn.savePDF(gcf)
```

```
%% Problem 1.c.ii check westward drift of ascending crossing of ground track

Td = 2*pi / (wdot + Mdot);
Dn = 2*pi / ( w_E - Odot );

u0 = w0 + M0;

u = u0 + udot * T + 2*pi/udot;

dlambda = -(w_E - Odot) * Td * 180/pi;

pred = lla_rot_c2(end,2) - lla_rot_c2(1,2);
% pred = pred / T;



%% PROBLEM 3 - MARS!!

clc

mu_M_km3 = 0.042828e6;

a0 = 3897;
i0 = 60 * pi/180;
e0 = 0.0063414;
w0 = 270 * pi/180;
O0 = 0;
M0 = 0;

oe0 = [a0; e0; i0; w0; O0; M0];
rv0 = rvOrb.orb2rv(oe0, mu_M_km3);


% set ode45 params
rel_tol = 1e-10;          % 1e-14 accurate; 1e-6 coarse
abs_tol = 1e-10;
options = odeset('reltol', rel_tol, 'abstol', abs_tol );

test = [3983.88414289582
          52.4285652190091
        -0.943967564657958
        0.0696070532710085
          1.63206285335835
          3.08662446538281] ;

% propagate orbit
T = 2*pi*sqrt(a0^3/mu_M_km3);
[t, rv_M] = ode45(@fn.EOM_Mars_J2_J3, [0 : 0.01 : T*10], test, options);

figure()
fn.plot3_xyz(rv_M);


%% subfunctions

function [lla_rot, rv_rot] = lla_rv_rot(t, rv_a, w_E, R_E, mu_E_km3)

    theta0 = 0;
    dt = t(2) - t(1);

    lla_rot = ecef2lla_1(rv_a(1,:)', R_E, mu_E_km3)';
    lla_rot(1, 1:2) = lla_rot(1, 1:2) * 180/pi;
    rv_rot = rv_a(1,:);

    for i = 2:length(rv_a)

        theta = -w_E * dt * (i-1) + theta0;
```

```
        r_rot = fn.rotate_xyz(rv_a(i,1:3), theta, 3);
        v_rot = fn.rotate_xyz(rv_a(i,4:6), theta, 3);
        rv_rot(i,:) = [r_rot; v_rot];

        lla_rot(i,:) = ecef2lla_1(rv_rot(i,:)', R_E, mu_E_km3);
        lla_rot(i, 1:2) = lla_rot(i, 1:2) * 180/pi;


    end

end

% ----------------------------------------------------------------------
function plot_gt(plot_option, fname, lla_1, lla_rot)

%     fname = '1.a Rotating Earth';
if plot_option == 1

    figure('name', fname)
    plot(lla_1(:, 2), lla_1(:, 1), 'b');
    hold on; grid on;
    lh = plot(lla_rot(:, 2), lla_rot(:, 1), 'r');
    xl = xlim; yl = ylim;

    cla;
    xlim(xl); ylim(yl);
    for i = 1:length(lla_rot)

        cla

        % fixed Earth
        plot(lla_1(1,2), lla_1(1,1), 'bo');
        lh1 = plot(lla_1(1:i, 2), lla_1(1:i, 1), 'b');
        plot(lla_1(i, 2), lla_1(i, 1), 'b^');

        % rotating Earth
        plot(lla_rot(1,2), lla_rot(1,1), 'ro');
        lh2 = plot(lla_rot(1:i, 2), lla_rot(1:i, 1), 'r');
        lh = plot(lla_rot(i, 2), lla_rot(i, 1), 'r^');

        legend([lh1 lh2], 'fixed E', 'rot E', 'location', 'best')

        title(sprintf( '%d / %d', i, length(lla_rot) ));
        pause(0.001)

    end

    title(fname)
    xlabel('Longitude (deg)');
    ylabel('Latitude (deg)');

elseif plot_option == 2

    figure('name', fname)
    hold on; grid on;

    % fixed Earth
    plot(lla_1(1,2), lla_1(1,1), 'bo');
    lh1 = plot(lla_1(:, 2), lla_1(:, 1), 'b');
    plot(lla_1(end, 2), lla_1(end, 1), 'b^');

    % rotating Earth
    plot(lla_rot(1,2), lla_rot(1,1), 'ro');
    lh2 = plot(lla_rot(:, 2), lla_rot(:, 1), 'r');
    lh = plot(lla_rot(end, 2), lla_rot(end, 1), 'r^');

    legend([lh1 lh2], 'fixed E', 'rot E', 'location', 'best')

```

```
    title(fname)
    xlabel('Longitude (deg)');
    ylabel('Latitude (deg)');

end

end

% ----------------------------------------------------------------------
% ECEF to geodetic (lat, lon)
function lla = ecef2lla_1(rv_ecef, R_E, mu)

% extract data
r_x = rv_ecef(1); r_y = rv_ecef(2); r_z = rv_ecef(3);
r_norm = norm(rv_ecef);

% obtain OEs
oe = rvOrb.rv2orb(rv_ecef, mu);
a = oe(1); e = oe(2); i = oe(3);
w = oe(4); O = oe(5); nu = oe(6);

% solve for longitude
r_delta = sqrt( r_x^2 + r_y^2 );
a_sin = asin(r_y / r_delta);
a_cos = acos(r_x / r_delta);

% find quadrant
if a_sin > 0
    if a_cos > 0
        a = abs(a_cos);
    else
        a = pi - abs(a_cos);
    end
else
    if a_cos > 0
        a = 2*pi - abs(a_cos);
    else
        a = pi + abs(a_cos);
    end
end

lon = mod(a, 2*pi);

% iterate for latitude
delta = asin(r_z / r_norm);

% assume first guess
lat0 = delta;
lat  = lat0;

C = R_E / sqrt( 1 - e^2 * ( sin(lat) )^2 );
S = R_E * (1-e^2) / sqrt( 1 - e^2*( sin(lat) )^2 );

tan_lat = (r_z + C * e^2 * sin(delta)) / r_delta;
lat = atan(tan_lat);
h = r_z / ( sin(delta) ) - S;

% loop
err = 1e-6;
while abs(lat - lat0) > 1e-4

    C = R_E / sqrt( 1 - e^2 * ( sin(lat) )^2 );
    S = R_E * (1-e^2) / sqrt( 1 - e^2*( sin(lat) )^2 );

    tan_lat = (r_z + C * e^2 * sin(delta)) / r_delta;
    lat = atan(tan_lat);
    if abs(90 - i) > 1
        h = r_z / ( cos(delta) ) - C;
```

```
    else
        h = r_z / ( sin(delta) ) - S;
    end

end

% h = h + R_E;
lla = [lat; lon; h];

end



% -----------------------------------------------------------------------
% ECEF2LLA - convert earth-centered earth-fixed (ECEF)
%            cartesian coordinates to latitude, longitude,
%            and altitude
%
% USAGE:
% [lat,lon,alt] = ecef2lla(x,y,z)
%
% lat = geodetic latitude (radians)
% lon = longitude (radians)
% alt = height above WGS84 ellipsoid (m)
% x = ECEF X-coordinate (m)
% y = ECEF Y-coordinate (m)
% z = ECEF Z-coordinate (m)
%
% Notes: (1) This function assumes the WGS84 model.
%        (2) Latitude is customary geodetic (not geocentric).
%        (3) Inputs may be scalars, vectors, or matrices of the same
%            size and shape. Outputs will have that same size and shape.
%        (4) Tested but no warranty; use at your own risk.
%        (5) Michael Kleder, April 2006
function [lla] = ecef2lla_2(ecef)

x = ecef(1); y = ecef(2); z = ecef(3);

% WGS84 ellipsoid constants:
a = 6378137;
e = 8.1819190842622e-2;
% calculations:
b   = sqrt(a^2*(1-e^2));
ep  = sqrt((a^2-b^2)/b^2);
p   = sqrt(x.^2+y.^2);
th  = atan2(a*z,b*p);
lon = atan2(y,x);
lat = atan2((z+ep^2.*b.*sin(th).^3),(p-e^2.*a.*cos(th).^3));
N   = a./sqrt(1-e^2.*sin(lat).^2);
alt = p./cos(lat)-N;
% return lon in range [0,2*pi)
lon = mod(lon,2*pi);
% correct for numerical instability in altitude near exact poles:
% (after this correction, error is about 2 millimeters, which is about
% the same as the numerical precision of the overall function)
k=abs(x)<1 & abs(y)<1;
alt(k) = abs(z(k))-b;

lla = [lat; lon; alt];

end
```