AE 502 Project 3

Linyi Hou | linyih2 | github.com/TigerHou2

April 20, 2021

Problem 1

Problem Statement: Given the following information for Mars:

$$\mu = 4.282837 \times 10^4 km^3/s^2 \tag{1}$$

$$r_{eq} = 3389.5 \text{ km} \tag{2}$$

$$\Omega_{Mars} = 7.0879 \times 10^{-5} rad/s \tag{3}$$

$$J_2 = 1960.45 \times 10^{-6} \tag{4}$$

$$P_{Mars} = 687 \text{ days} \tag{5}$$

Part a

Q: Using the mean rate equation for right ascension of ascending node, compute the inclination required for a circular sun-synchronous orbit with an altitude of 300 km around Mars.

A: The orbit precession rate equation expresses the change in right ascension of ascending node as follows:

$$\frac{d\bar{\Omega}}{dt} = -\frac{3nJ_2r_{eq}^2}{2p^2}\cos(i) \tag{6}$$

where $n = \sqrt{\mu/a^3}$ and $p = a(1-e^2)$. Since the orbit is circular with an altitude of 300 km, a and e may be trivially determined. Setting $\frac{d\bar{\Omega}}{dt} = \Omega_{Mars}$ allows us to solve for the sun-synchronous inclination, i_{SSO}. We find that $i_{SSO} = 92.65^o$.

Part b

Q: Propagate the orbit for 10 days considering the J_2 perturbation and a rotating Mars. Plot and comment on the orbit.

A: To consider the rotation of Mars, construct the rotation matrix R_{PCPF}, which converts a vector from the planet-centered inertial (PCI) frame to the planet-centered planet-fixed

(PCPF) frame:

$$R_{PCPF} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where $\theta = t \cdot \Omega_{Mars}$ and t is the time since epoch. Then the rotation matrix to convert from the PCPF frame back to the PCI frame, R_{PCI}, is simply R^T_{PCPF}. We could use the same equations of motion to propagate the orbit:

$$\dot{\mathbf{r}} = \mathbf{v} \tag{8}$$

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{a}_{J_2} \tag{9}$$

with the new definition of $\mathbf{a}_{J_2}$, where $'$ denotes positions in the PCPF frame:

$$\mathbf{a}_{J_2} = R_{PCI} \cdot \left(-\frac{3}{2}\right) J_2 \left(\frac{\mu}{r'^2}\right)\left(\frac{r_{eq}}{r'}^2\right) \begin{bmatrix} \left(1 - 5\left(\frac{z'}{r'}\right)^2\right)\frac{x'}{r'} \\ \left(1 - 5\left(\frac{z'}{r'}\right)^2\right)\frac{y'}{r'} \\ \left(3 - 5\left(\frac{z'}{r'}\right)^2\right)\frac{z'}{r'} \end{bmatrix} \tag{10}$$

The results are shown in Fig. 1-2. As expected, the orbit precesses slowly and matches the rate at which Mars orbits the Sun, as indicated by the slow, linear increase in the RAAN. While it is difficult to see the per-orbit variation in the other orbital parameters, they do appear to oscillate between bounded values.

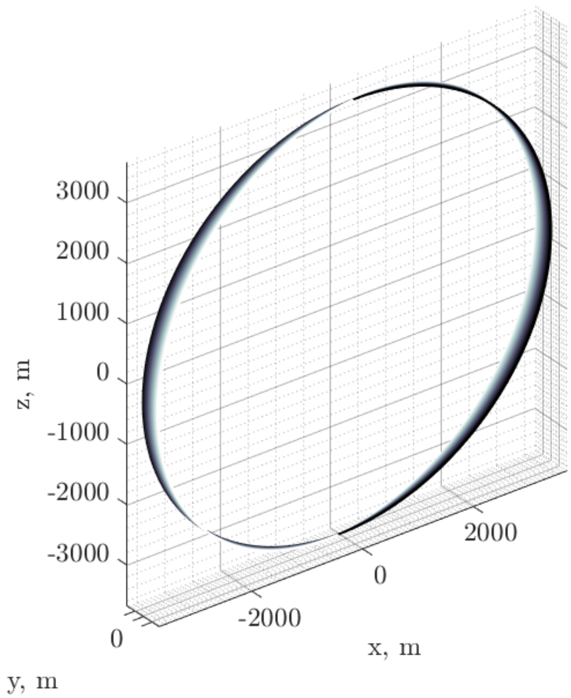


Figure 1: 10-day propagation result of a Mars circular 300-km sun-synchronous orbit.

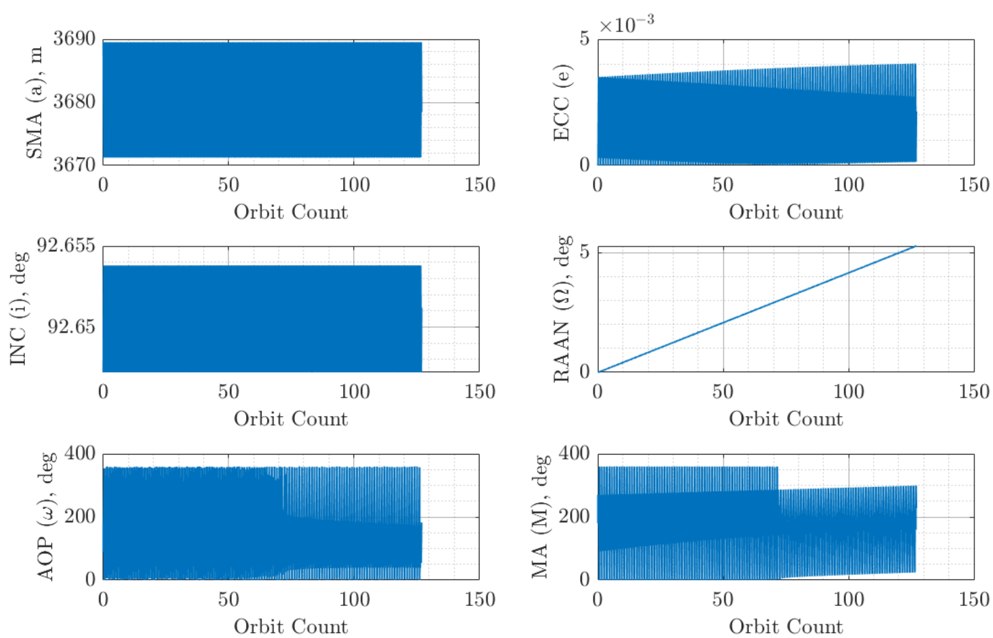


Figure 2: Corresponding orbital parameters of the Mars sun-synchronous orbit.

Part c

Q: Compute the variation in energy as a function of time. That is, $\Delta E = \frac{E(t) - E(t_0)}{E(t_0)}$. Plot and comment on the result. Is energy conserved?

A: The orbit energy is computed by the following:

$$E(t) = \frac{1}{2} V_{PCPF}(t)^2 - \frac{1}{2}\left(r_x(t)^2 + r_y(t)^2\right) + V(r_x(t), r_y(t), r_z(t)) \tag{11}$$

where the potential, V, is given by

$$V = -\frac{\mu}{r}\left[1 - \left(\frac{r_{eq}}{r}\right)^2 J_2 p_2(\frac{z}{r})\right] \tag{12}$$

and the function $p_2()$ is the Legendre polynomial

$$p_2(v) = (3v^2 - 1)/2 \tag{13}$$

With the above definitions, the orbit energy over 10 days was computed and shown in Fig. 3. Energy is conserved because the errors arise from integration error as opposed to an actual change in the energy of the orbit. It was found that stricter integration tolerances would yield a lower relative energy variation, which indicates that the change is purely due to the limits of numerical integration.

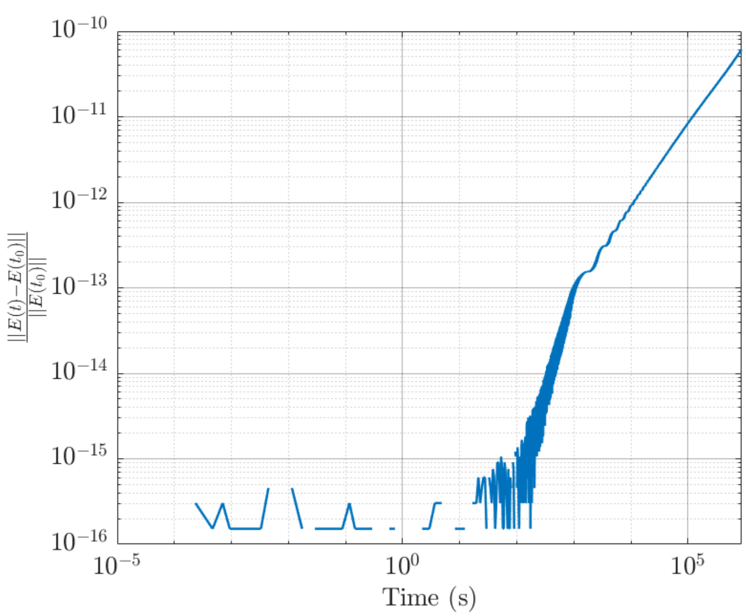


Figure 3: Relative orbit energy variation over 10 days for the Mars sun-synchronous orbit.

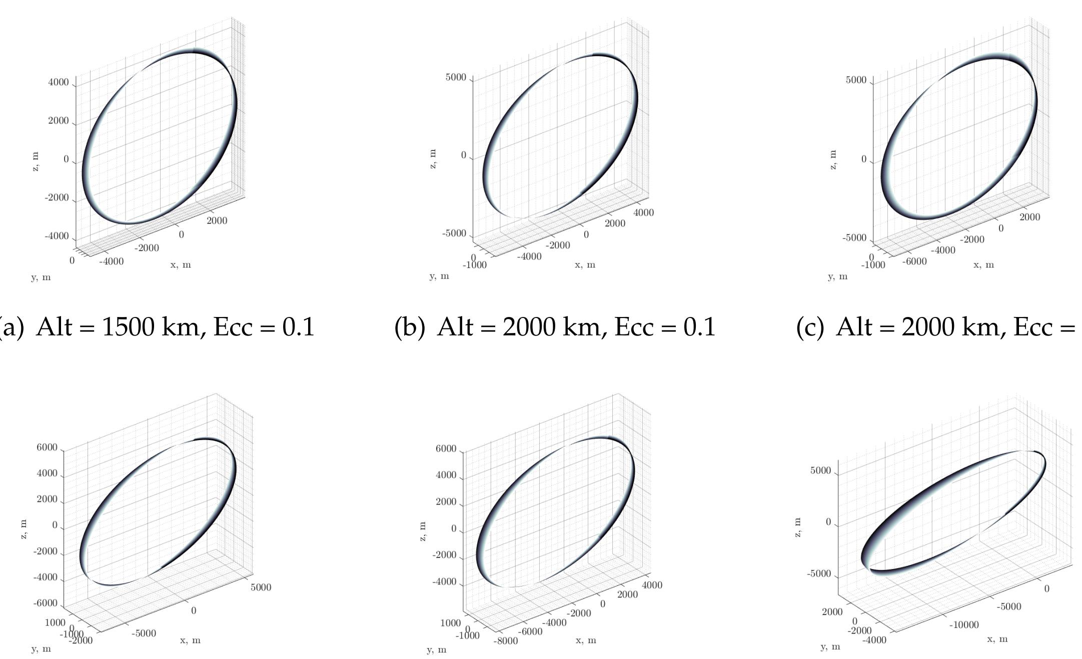
Part d

Q: Compute and plot five more sun-synchronous orbits around Mars, using different values for the semi-major axis and eccentricity. Comment on the results. In particular, comment on how inclination varies as a function of semi-major axis.

A: Simulation results are tabulated in Table 1. The corresponding orbit plots are shown in Fig. 1.

Table 1: Sun-Synchronous Mars Orbits

Altitude (km)	Eccentricity	i_{SSO} (^{o})
1500	0.1	96.97
2000	0.1	99.82
2000	0.3	98.28
3000	0.1	108.02
3000	0.3	105.15
6000	0.6	119.83



(a) Alt = 1500 km, Ecc = 0.1
(b) Alt = 2000 km, Ecc = 0.1
(c) Alt = 2000 km, Ecc = 0.3
(d) Alt = 3000 km, Ecc = 0.1
(e) Alt = 3000 km, Ecc = 0.3
(f) Alt = 6000 km, Ecc = 0.6

Figure 4: Sun-Synchronous Mars Orbits.

Problem 2

Problem Statement: A 400 kg spacecraft departs a geostationary orbit using a low thrust engine with the following parameters: specific impulse ($I_{sp} = 3000s$), engine thrust ($T = 0.136N$), exhaust velocity $c = I_{sp} \cdot g_0$, where g_0 is the gravitational acceleration at the surface of the Earth. The mass flow rate is given by $\dot{m} = T/c$, and $m = m_0 - \dot{m}t$.Assume that the direction of the thruster is always aligned with the instantaneous velocity vector and that the thrust is on constantly. Propagate the equations of motion, including the J2 perturbation and the perturbation due to the low thrust engine. Note, that the mass of the spacecraft changes during the mission (fuel is consumed). How long does it take the spacecraft to reach escape velocity? Plot the resulting trajectory.

A: The equations of motion for the problem are as follows:

$$\dot{\mathbf{r}} = \mathbf{v} \tag{14}$$

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{a}_{J_2} + \frac{\mathbf{v}}{v} \cdot \frac{T}{c} \tag{15}$$

Integrating the equations of motion yielded the following trajectory:

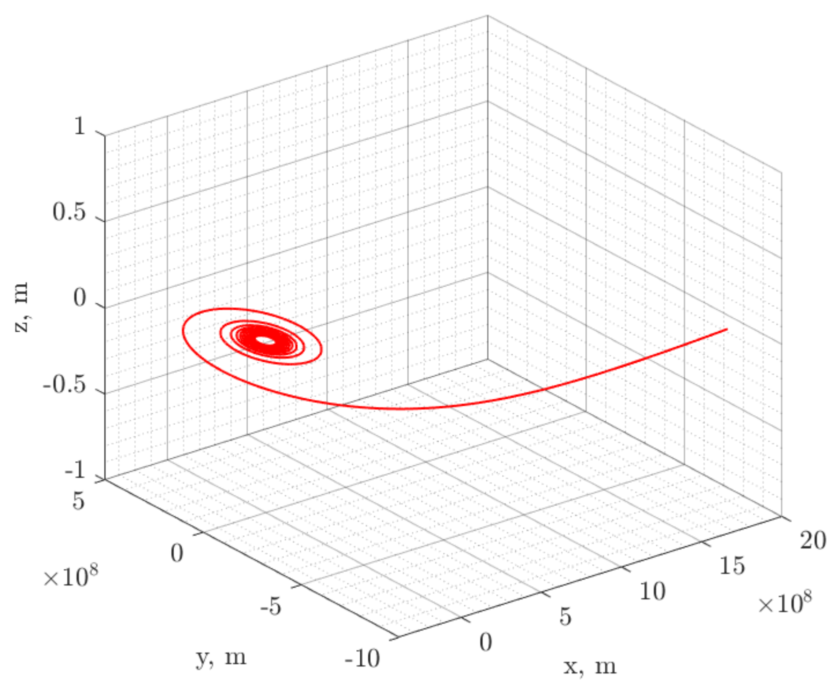


Figure 5: Constant low thrust trajectory from GEO to Earth escape.

The spacecraft took 84.0533 days to each escape velocity.

Appendix A: Code for Problem 1

```
%% AE 502 HW3 Problem 1, Spring 2021
%   Tiger Hou

close all
clear;clc
latexify

%% Part a

% Mars general parameters
mu = 4.282837e4; % km^3/s^2, Mars gravitational parameter
J2 = 1960.45e-6; % J2 perturbation coefficient
req = 3389.5; % km, equatorial radius
Omars = 7.0879e-5; % rad/s, Mars spin rate
P = 687 * 24 * 3600; % seconds, Mars orbit period around Sun

% orbit parameters
e = 0.1;
alt = 1000; % km, orbit altitude
a = alt + req;

n = sqrt(mu/a^3); % mean motion
p = a*(1-e^2); % semi-latus rectum

dodt_sso = 2*pi / P; % orbit precession rate of SSO at Mars
i_sso = acos( dodt_sso / (-3/2*J2*n*(req/p)^2) );

disp(['The sun-synchronous inclination is ' ...
        num2str(rad2deg(i_sso)) ' deg.'])

%% Part b

% initial conditions
i = i_sso;
o = deg2rad(0);
w = deg2rad(0);
M0 = deg2rad(0);
E0 = kepler(M0,e);
```

```
f0 = 2 * atan(sqrt((1+e)/(1-e))*tan(E0/2));

% find initial position and velocity vectors
[r0,v0] = Get_Orb_Vects([a,e,i,o,w,f0],mu);

% define the perturbation equation as p(rv)
p = @(rv) -3/2 * J2 * (mu/norm(rv)^2) * (req/norm(rv))^2 * ...
    [ ( 1-5*(rv(3)/norm(rv))^2 ) * rv(1)/norm(rv); ...
      ( 1-5*(rv(3)/norm(rv))^2 ) * rv(2)/norm(rv); ...
      ( 3-5*(rv(3)/norm(rv))^2 ) * rv(3)/norm(rv) ];

% calculate orbit period
T = 2*pi * sqrt(a^3/mu); % seconds

% final time
tf = 10 * 24 * 3600;

% ode45
rv0 = [r0;v0];
options = odeset('RelTol',1e-12,'AbsTol',1e-12);
[tOut,rvOut] = ode45(   @(t,rv)ff(rv,mu,1,p,t,Omars),...
                        [0,tf],rv0,options);

% plotting
rvOut = rvOut';
figure(1)
colorplot(rvOut(1,:), rvOut(2,:),rvOut(3,:),...
            'colormap','bone',...
            'linewidth',1.2)
xlabel('x, m')
ylabel('y, m')
zlabel('z, m')
axis equal
setgrid
latexify(16,14,14)

% convert position, velcoity data into orbit parameters
N = size(rvOut,2);
paramOut = nan(size(rvOut));
for j = 1:N
```

```
    [a_,e_,i_,o_,w_,f_] = Get_Orb_Params(rvOut(1:3,j),rvOut(4:6,j),mu);
    E_ = 2 * atan(sqrt((1-norm(e_))/(1+norm(e_)))*tan(f_/2));
    M_ = E_ - norm(e_)*sin(E_);
    paramOut(:,j) = [a_,norm(e_),i_,o_,w_,M_]';
end

% plot results
ylabel_vec = {'SMA (a), m', ...
              'ECC (e)', ...
              'INC (i), deg', ...
              'RAAN ($\Omega$), deg', ...
              'AOP ($\omega$), deg', ...
              'MA (M), deg'};
xlabel_val = 'Orbit Count';
figure(2)
for j = 1:6
    subplot(3,2,j)
    dat = paramOut(j,:);
    if j == 4 % RAAN loop-around after 2*pi
        dat = mod(dat+pi,2*pi)-pi;
    end
    if j == 6 % mean anomaly loop-around after 2*pi
        dat = mod(dat,2*pi);
    end
    if j >= 3 % conversion to degrees
        dat = rad2deg(dat);
    end
    plot(tOut/T,dat,'Linewidth',1.2)
    xlabel(xlabel_val)
    ylabel(ylabel_vec{j})
    setgrid
end
latexify(26,16,14)


%% Part c

v_PCPF = nan(size(rvOut(4:6,:)));

for j = 1:size(v_PCPF,2)
```

```
%     theta = tOut(j) * Omars;
%     R_eci2ecef = [  cos(theta), sin(theta), 0; ...
%                    -sin(theta), cos(theta), 0; ...
%                     0         , 0         , 1];
%     v_PCPF(:,j) = R_eci2ecef * rvOut(4:6,j);
    v_PCPF(:,j) = rvOut(4:6,j) + cross(Omars*[0;0;1],rvOut(1:3,j));
end

rr = vecnorm(rvOut(1:3,:));
p2 = rvOut(3,:) ./ rr;

E = 1/2 * vecnorm(v_PCPF).^2 ...
  - 1/2 * Omars^2 * sum(rvOut(1:2,:).^2) ...
  - mu ./ rr ...
       .* ( 1 - (req./rr).^2 .* J2 .* (3*p2.^2-1)/2 );

E0 = E(1);

figure;
loglog(tOut, abs( (E-E0)/E0 ), 'LineWidth', 1.5)
xlabel('Time (s)')
ylabel('$\frac{||E(t)-E(t_0)||}{||E(t_0)||}$')
setgrid
latexify(18,14,16)


%% function definitions
function rv_dot = ff(rv,mu,N,p,t,Omars)
% takes the 6xN position & velocity vector and computes the derivative
%   where N is the number of particles to track
%   also applies perturbation in the form of a function handle p
%   which takes argument p(r)

rv_dot = zeros(6,N);

for i = 1:N
    theta = t*Omars;
    R_eci2ecef = [  cos(theta), sin(theta), 0; ...
                   -sin(theta), cos(theta), 0; ...
                    0         , 0         , 1];
```

```
    R_ecef2eci = R_eci2ecef';
    % velocity
    rv_dot(1:3,i) = rv(4:6,i);
    % acceleration
    rv_dot(4:6,i) = -mu/norm(rv(1:3,i))^3*rv(1:3,i) ...
                  + R_ecef2eci*p(R_eci2ecef*rv(1:3,i));
end

end
```

Appendix B: Code for Problem 2

```
%% AE 502 HW3 Problem 2, Spring 2021
%   Tiger Hou

close all
clear;clc
latexify

%% problem setup

% Earth orbit general parameters
mu = 3.986e14; % m^3/s^2, Earth gravitational parameter
J2 = 1082.63e-6; % J2 perturbation coefficient
req = 6378.137e3; % m, equatorial radius

% orbit parameters
alt_geo = 35786e3; % m, GEO altitude
a = alt_geo + req;
e = 0;
i = 0;
o = 0;
w = 0;
f = 0;

m0 = 400; % kg, initial mass of spacecraft
thrust = 0.136; % N, low thrust
Isp = 3100; % s, specific impulse
g0 = 9.81; % m/s^2, Earth gravity
c = Isp * g0; % exhaust velocity

% find initial position and velocity vectors
[r0,v0] = Get_Orb_Vects([a,e,i,o,w,f],mu);

% define the low thrust engine as a perturbation
% with current mass as the fourth state
pT = @(v,m) [ v / norm(v) * thrust / m; -thrust/c ];

% define the perturbation equation as p(r,v,m,t)
p = @(r,v,m) -3/2 * J2 * (mu/norm(r)^2) * (req/norm(r))^2 * ...
```

```
    [ ( 1-5*(r(3)/norm(r))^2 ) * r(1)/norm(r); ...
      ( 1-5*(r(3)/norm(r))^2 ) * r(2)/norm(r); ...
      ( 3-5*(r(3)/norm(r))^2 ) * r(3)/norm(r); ...
      0 ] + ...
    pT(v,m);

% calculate orbit period
T = 2*pi * sqrt(a^3/mu); % seconds


%% propagate orbit

% ode45
rv0 = [r0;v0;m0];
options = odeset('RelTol',1e-12,'AbsTol',1e-12);
[tOut,rvOut] = ode45(   @(t,rv)ff(rv,mu,p),...
                        [0,24*3600*100],rv0,options);
rvOut = rvOut';

%% check escape

% check escape velocity
escaped = floor(vecnorm(rvOut(4:6,:)) ...
                ./ sqrt( 2*mu ./ vecnorm(rvOut(1:3,:))));

idx = find(escaped,1,'first');
if ~isempty(idx)
    disp([  'The spacecraft escaped at t = ' ...
            num2str(tOut(idx)/24/3600) ' days.'])
else
    disp([  'The spacecraft did not escape after' ...
            num2str(tOut(end)/24/3600) ' days.'])
end

%% plot results

% plotting
figure(1)
plot3(rvOut(1,:), rvOut(2,:),rvOut(3,:),'r','LineWidth',1.2)
xlabel('x, m')
```

```
ylabel('y, m')
zlabel('z, m')
% axis equal
setgrid
latexify(16,14,14)


%% plot orbit parameters

% convert position, velcoity data into orbit parameters
N = idx-1;
paramOut = nan(size(rvOut,1),N);
for j = 1:N
    [a_,e_,i_,o_,w_,f_] = Get_Orb_Params(rvOut(1:3,j),rvOut(4:6,j),mu);
    E_ = 2 * atan(sqrt((1-norm(e_))/(1+norm(e_)))*tan(f_/2));
    M_ = E_ - norm(e_)*sin(E_);
    paramOut(:,j) = [a_,norm(e_),i_,o_,w_,M_,rvOut(7,j)]';
end

% plot results
ylabel_vec = {'SMA (a), m', ...
              'ECC (e)', ...
              'INC (i), deg', ...
              'RAAN ($\Omega$), deg', ...
              'AOP ($\omega$), deg', ...
              'MA (M), deg'};
xlabel_val = 'Orbit Count';
figure(2)
for j = 1:6
    subplot(3,2,j)
    dat = paramOut(j,:);
    if j == 4 % RAAN loop-around after 2*pi
        dat = mod(dat+pi,2*pi)-pi;
    end
    if j == 6 % mean anomaly loop-around after 2*pi
        dat = mod(dat,2*pi);
    end
    if j >= 3 % conversion to degrees
        dat = rad2deg(dat);
    end
```

```
    plot(tOut(1:N)/T,dat,'Linewidth',1.2)
    xlabel(xlabel_val)
    ylabel(ylabel_vec{j})
    setgrid
end
latexify(20,18,14)


%% function definitions
function rv_dot = ff(rv,mu,p)
% takes the 6xN position & velocity vector and computes the derivative
%   where N is the number of particles to track
%   also applies perturbation in the form of a function handle p
%   which takes argument p(r)

rv_dot = zeros(7,1);

% velocity
rv_dot(1:3) = rv(4:6);
% acceleration and mass
rv_dot(4:7) = - mu/norm(rv(1:3))^3*[rv(1:3);0] ...
                + p(rv(1:3), rv(4:6), rv(7));

end
```