HW2 Problem 1: Orbit propagation with J2 perturbation

Table of Contents

1a) Compute the Cartesian partial derivatives of U

See appendix for hand-derived confirmation of results

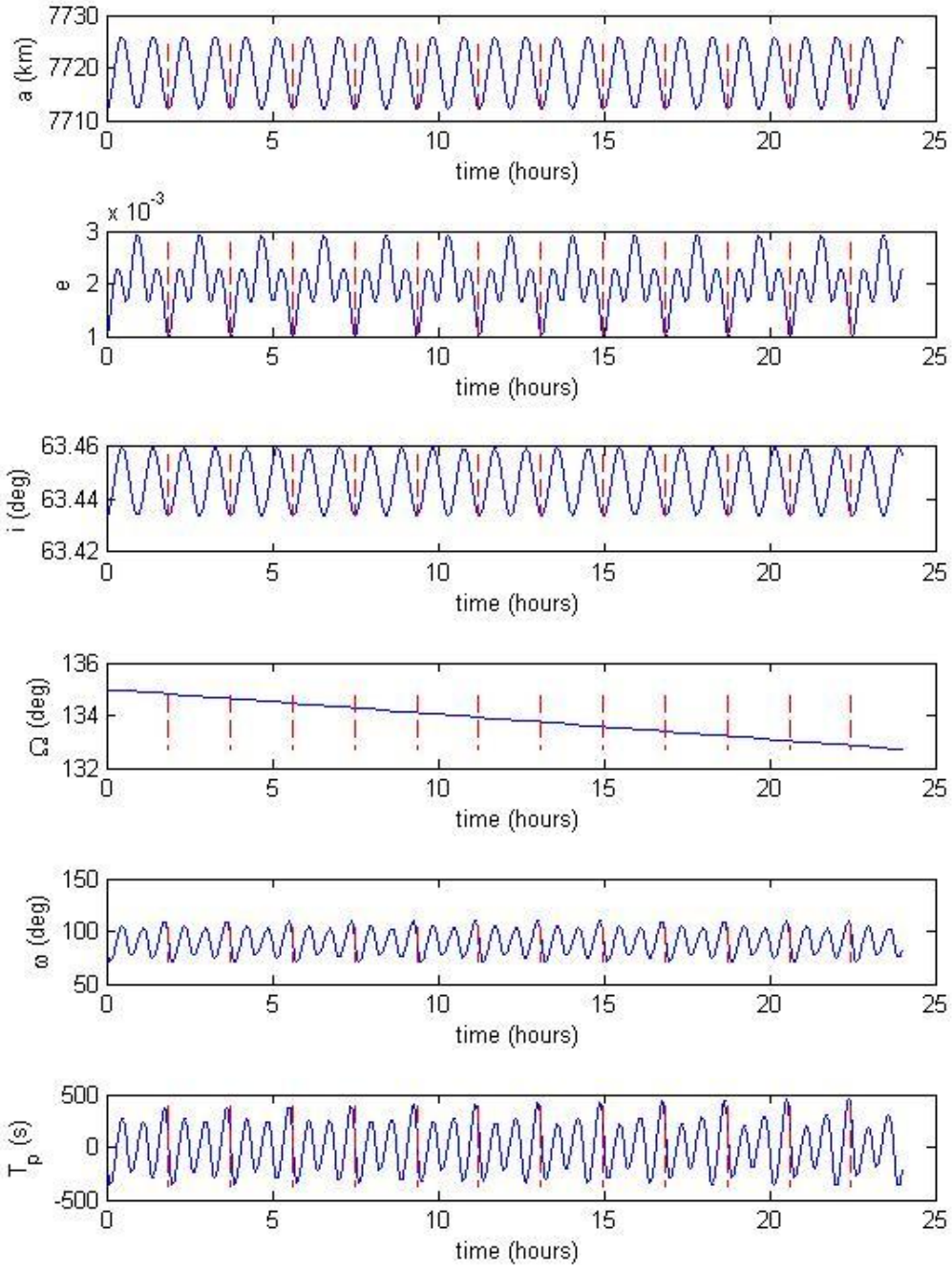
```
fprintf('\n');
clearvars -except function_list pub_opt
close all

syms mu J2 Re x y z
r = (x^2+y^2+z^2)^(1/2);

U = mu/r*(-J2*Re^2/r^2*(3/2*z^2/r^2-1/2));

dU_dx = simplify(diff(U,x))
dU_dy = simplify(diff(U,y))
dU_dz = simplify(diff(U,z))
clearvars -except function_list pub_opt
```

```
dU_dx =

-(3*J2*Re^2*mu*x*(x^2 + y^2 - 4*z^2))/(2*(x^2 + y^2 + z^2)^(7/2))


dU_dy =

-(3*J2*Re^2*mu*y*(x^2 + y^2 - 4*z^2))/(2*(x^2 + y^2 + z^2)^(7/2))


dU_dz =

-(3*J2*Re^2*mu*z*(3*x^2 + 3*y^2 - 2*z^2))/(2*(x^2 + y^2 + z^2)^(7/2))
```

1b) Propagate orbit, show orbital elements

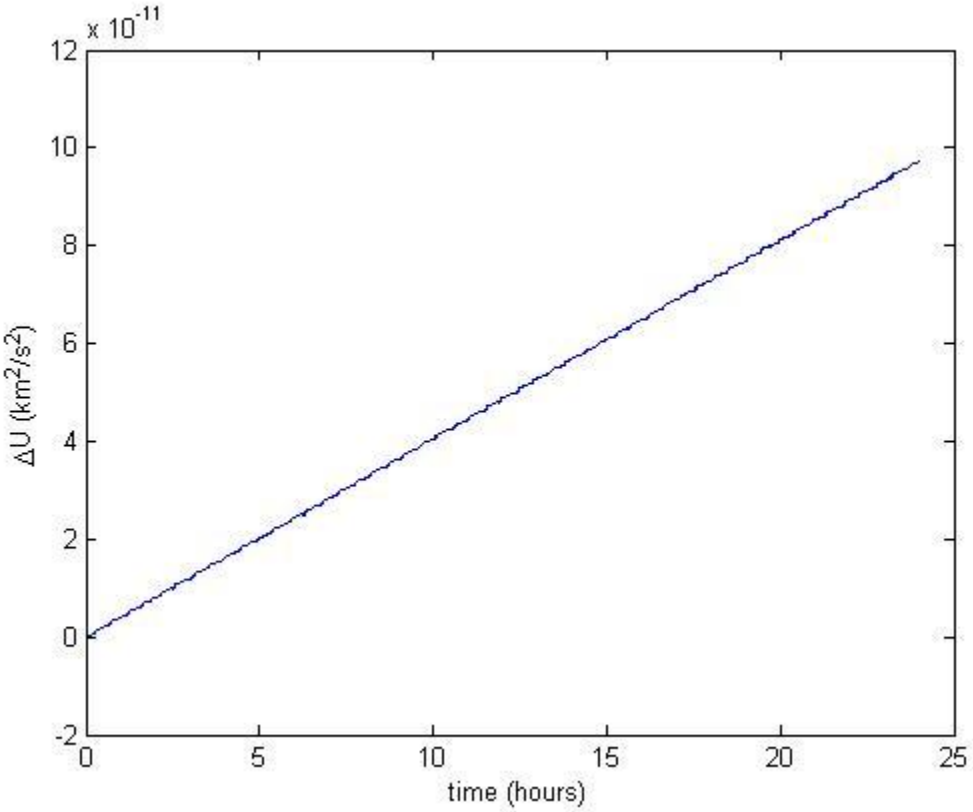
All elements but Ω vary sinusoidally, but the long-term trend is constant (same value at the same point in the period). Ω decreases with J2 effects.

```
ode_opts = odeset('RelTol', 1e-12, 'AbsTol', 1e-20);
propagator_opts.J2 = 1;
r = [-2436.45; -2436.45; 6891.037]; % km
v = [5.088611; -5.088611; 0.0]; % km/s
state = [r;v];

times = 0:20:3600*24;

[T,X] = ode45(@two_body_state_dot, times, state, ode_opts, propagator_opts);

oe_vec = zeros(6,length(times));
for ii = 1:length(times)
    oe_vec(:,ii) = cart2oe(X(ii,:)');
end

plot_oe(oe_vec, times, '')
```


a (km)
7730
7720
7710
0
5
10
15
20
25
time (hours)
x 10^-3
e
3
2
1
time (hours)
i (deg)
63.46
63.44
63.42
time (hours)
Ω (deg)
136
134
132
time (hours)
ω (deg)
150
100
50
time (hours)
T_p (s)
500
0
-500
time (hours)

1c) Compute change in energy

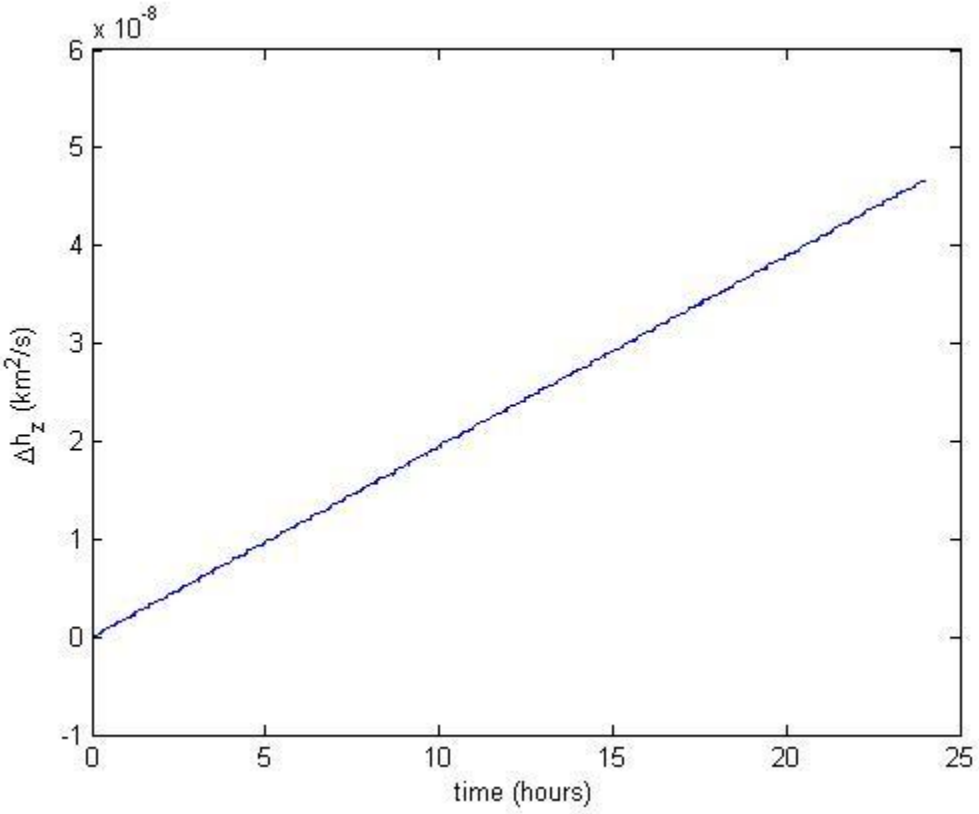
```
J2 = 0.00108248;
mu = 398600.4; %km3/s2
Re = 6378.145; %km
r_mag = zeros(1,length(times));
v_mag = zeros(1,length(times));
for i = 1:length(times)
    r_mag(i) = norm(X(i,1:3));
    v_mag(i) = norm(X(i,4:6));
end
KE = v_mag.*v_mag/2;%
PE = -3.986e5./r_mag + ...
    mu./r_mag.*(J2*Re*Re./(r_mag.*r_mag).*(3/2*(X(:,3).*X(:,3))'./...
    (r_mag.*r_mag)-1/2));
deltaE = KE + PE - (KE(1)+PE(1));
figure
plot(times/3600, deltaE)
ylabel('\DeltaU (km^2/s^2)')
xlabel('time (hours)')
```



1d) Compute h_k

```
h = zeros(length(times),3);
for i = 1:length(times)
```

```
   h(i,:) = cross(X(i,1:3), X(i,4:6));
end
figure
plot(times/3600, h(:,3)-h(1,3))
ylabel('\Deltah_z (km^2/s)')
xlabel('time (hours)')
```



Published with MATLAB® R2013b

Problem 1 derivation

1. $U_{J_2} = -\mu J_2 \frac{R_\oplus^2}{r^3}\left(\frac{3}{2}\sin^2\phi - \frac{1}{2}\right) = \mu J_2 \frac{R_\oplus^2}{r^3}\left(\frac{3}{2}\frac{z^2}{r^2} - \frac{1}{2}\right)$

$\ddot{\vec{r}}_{J_2} = \nabla U_{J_2}$

$\frac{\partial U}{\partial x} = \frac{\partial}{\partial x}\left(\mu J_2 R_\oplus^2\left(\frac{3}{2}\frac{z^2}{r^5} - \frac{1}{2}\cdot\frac{1}{r^3}\right)\right) = -\mu J_2 R_\oplus^2\left(\frac{3}{2}z^2\frac{\partial}{\partial x}\left((x^2+y^2+z^2)^{-5/2}\right) - \frac{1}{2}\frac{\partial}{\partial x}\left((x^2+y^2+z^2)^{-3/2}\right)\right)$

$= \mu J_2 R_\oplus^2\left(\frac{3}{2}z^2\cdot\frac{-5}{2}\cdot(x^2+y^2+z^2)^{\frac{-7}{2}}\cdot 2x + \frac{1}{2}\cdot\frac{3}{2}(x^2+y^2+z^2)^{-5/2}\cdot 2x\right)$

$= \mu J_2 R_\oplus^2\left(-\frac{15}{2}\frac{z^2 x}{r^7} + \frac{3}{2}\frac{x}{r^5}\right) = +\frac{3}{2}\mu J_2 \frac{R_\oplus^2}{r^5}\left(5\frac{z^2}{r^2} - 1\right)x$

$\frac{\partial U}{\partial y} = +\frac{3}{2}\mu J_2 \frac{R_\oplus^2}{r^5}\left(5\frac{z^2}{r^2} - 1\right)y$

$\frac{\partial U}{\partial z} = -\mu J_2 R_\oplus\left(-\frac{15}{2}\frac{z^2 z}{r^7} + \frac{3}{2}\cdot\frac{2z}{r^5} + \frac{3}{2}\frac{z}{r^5}\right) = +\frac{3}{2}\mu J_2 R_\oplus\left(5\frac{z^2}{r^2} - 3\right)z$

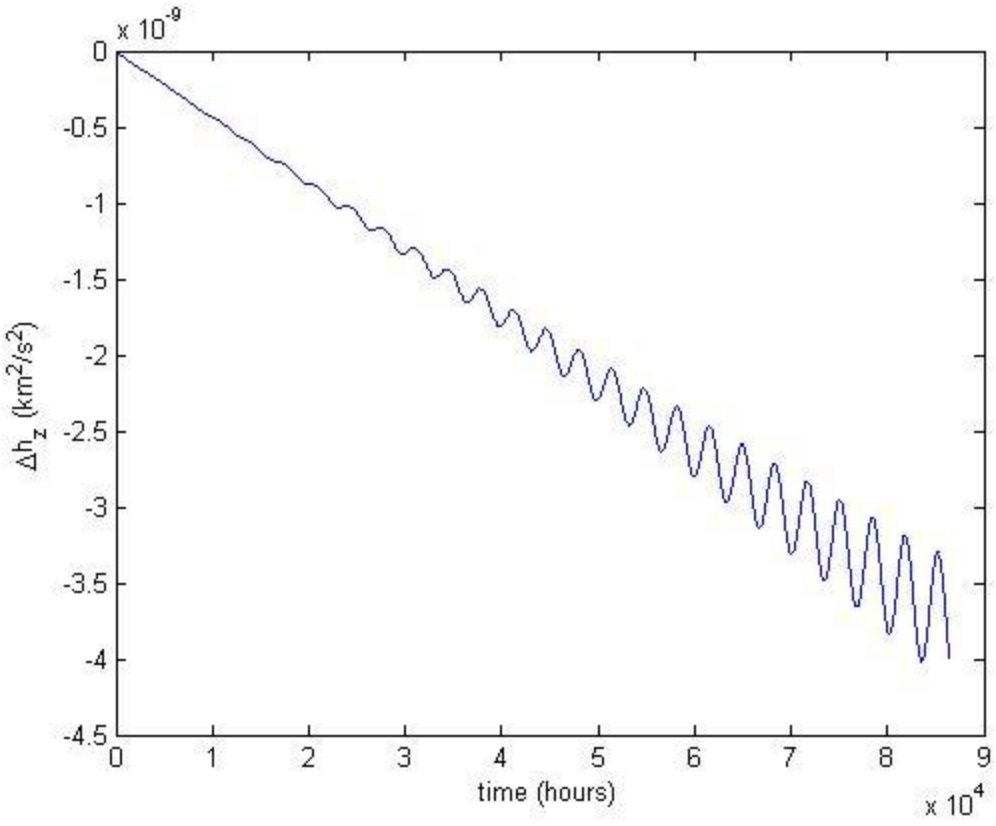
HW2 Problem 2: Orbit propagation with J2 perturbation and drag

Table of Contents

2a) Propagate orbit, compute change in energy

Energy is not conserved because drag is not a conservative force.

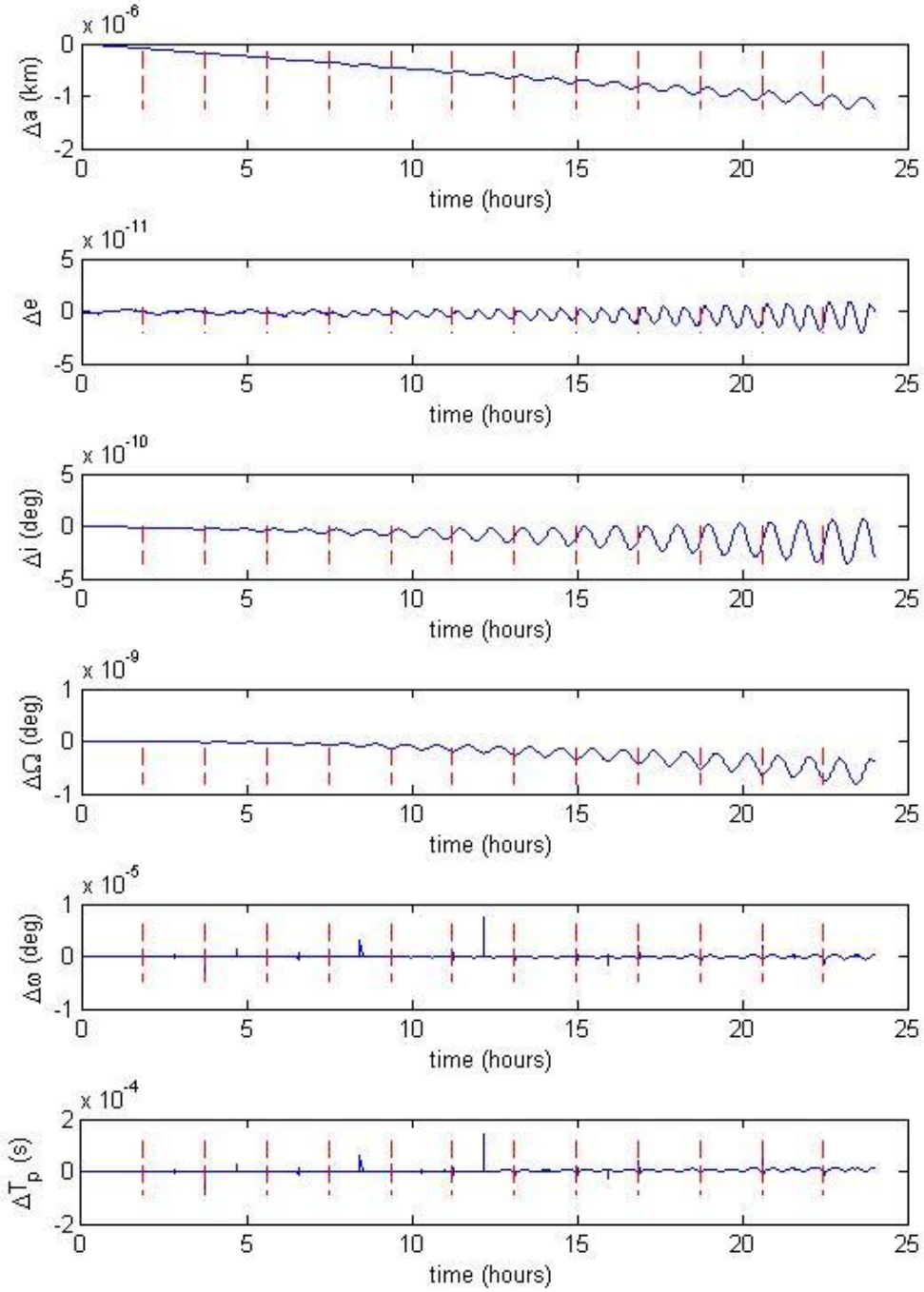
```
propagator_opts.drag.use = 1;
propagator_opts.drag.Cd = 2.0;
propagator_opts.drag.A = 3.6;
propagator_opts.drag.m = 1350;

[T2,X2] = ode45(@two_body_state_dot, times, state, ode_opts, ...
    propagator_opts);

oe_vec2 = zeros(6,length(times));
for ii = 1:length(times)
    oe_vec2(:,ii) = cart2oe(X2(ii,:)');
end

%Change in energy
r_mag = zeros(1,length(times));
v_mag = zeros(1,length(times));
for i = 1:length(times)
    r_mag(i) = norm(X2(i,1:3));
    v_mag(i) = norm(X2(i,4:6));
end
KE = v_mag.*v_mag/2;
PE = -3.986e5./r_mag ...
    + mu./r_mag.*(J2*Re*Re./(r_mag.*r_mag).*(3/2*(X(:,3).*X(:,3))'./...
    (r_mag.*r_mag)-1/2));
deltaE = KE + PE - (KE(1)+PE(1));
figure
plot(times, deltaE)
ylabel('\Deltah_z (km^2/s^2)')
xlabel('time (hours)')
```



2b) Plot the OE diffs from Problem 1

The semi-major axis reduces as a result of the drag.

```
plot_oe(oe_vec, times, '\Delta',...
    2*pi/sqrt(mu/(oe_vec2(1)*oe_vec2(1)*oe_vec2(1))), oe_vec2)
```



Published with MATLAB® R2013b

HW 2 Master Script

Table of Contents

Initialize

```
if ispc
    addpath('C:\Users\John\Documents\ASEN5070_SOD\tools')
end
clear all
clc

% Cell array to track what functions are used, so they can be published
% later
global function_list;
function_list = {};

% publishing options
pub_opt.format = 'pdf';
pub_opt.outputDir = '.\html';
pub_opt.imageFormat = 'jpeg';
pub_opt.figureSnapMethod = 'entireGUIWindow';
pub_opt.useNewFigure = true ;
pub_opt.maxHeight = Inf;
pub_opt.maxWidth = Inf;
pub_opt.showCode = true;
pub_opt.evalCode = true;
pub_opt.catchError = true;
pub_opt.createThumbnail = true;
pub_opt.maxOutputLines = Inf;
```

Run Problem scripts and publish them

```
% Problem 1
publish('HW2_P1', pub_opt);

% Problem 2
publish('HW2_P2', pub_opt);
```

Publishing tools and support code

```
pub_opt.outputDir = '.\tools';
pub_opt.evalCode = false;
```

```
%Publish all used functions
function_list = ...
    [function_list; 'C:\Users\John\Documents\ASEN5070_SOD\tools\fcnPrintQueue'];
for idx = 1:length(function_list)
    publish(function_list{idx}, pub_opt);
end
```

Published with MATLAB® R2013b

```
function OE = cart2oe(state)
%cart2oe   Return classical orbital elements given state vector. Earth
%   orbits only. State given in km, km/s.
%   OE = [a; e; i; RAAN; w; f] = cart2oe(state)
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

mu = 3.986e5; % km3/s2
r_vec = state(1:3);
r = norm(r_vec);
v_vec = state(4:6);
v = norm(v_vec);
h = cross(r_vec,v_vec);

% Specific Energy:
E = v*v/2 - mu/r;

a = -mu/2/E;
e = sqrt(1-dot(h,h)/a/mu);
i = acos(h(3)/norm(h));
RAAN = atan2(h(1), -h(2));

arg_lat = atan2(r_vec(3)/sin(i), (r_vec(1)*cos(RAAN)+r_vec(2)*sin(RAAN)));
cosf = (a*(1-e*e)-r)/e/r;
f = acos(max([min([1, cosf]), -1]));
if dot(r_vec,v_vec) < 0
    f = 2*pi - f;
end
w = arg_lat - f;
if w < 0
    w = w + 2*pi;
end

OE = [a; e; i; RAAN; w; f];
```

Published with MATLAB® R2013b

```
function accel = drag_accel( state, drag_data )
%drag_accel calculate drag on spacecraft
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

Cd = drag_data.Cd;
A = drag_data.A;
m = drag_data.m;

rho0 = 4e-13; %kg/m3
r0 = 7298.145; %km
H = 200.0; %km
theta_dot = 7.29211585530066e-5; %rad/s

r = norm(state(1:3));

rho = rho0*exp(-(r-r0)/H);
rel_wind = [state(4) + theta_dot*state(2);
    state(5) - theta_dot*state(1);
    state(6)];

accel = -0.5*Cd*A/m*rho*rel_wind*norm(rel_wind);

end
```

Published with MATLAB® R2013b

```
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
%         fprintf('%s\n', filename);
        function_list = [function_list; filename];
    end
end
end
```

Published with MATLAB® R2013b

```
function E = get_eccentric_anomaly(f, e)
%get_eccentric_anomaly Calculate eccentric anomaly
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

E = acos((e + cos(f))./(1+e.*cos(f)));

E(f > pi) = 2*pi - E(f > pi);
```

Published with MATLAB® R2013b

```
function M = get_mean_anomaly(E, e)
%get_mean_anomaly Calculate mean anomaly
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

M = E - e.*sin(E);
```

Published with MATLAB® R2013b

```
function Tp = get_time_of_periapsis(a,e,f,t)
%get_time_of_periapsis   Return time of periapsis
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

mu = 3.986004e5;
% a = oe_vec(1,:);
% e = oe_vec(2,:);
% f = oe_vec(6,:);
mean_motion = sqrt(mu./(a.*a.*a));
E = get_eccentric_anomaly(f,e);
Tp = t - get_mean_anomaly(E, e)./mean_motion;
% plot(Tp)
% plot(mod(Tp, (2*pi)./mean_motion))
time_periapse_angle = mod(t, (2*pi)./mean_motion).*mean_motion ...
    - (E - e.*sin(E));

time_periapse_angle = unwrap(time_periapse_angle);
% plot(time_periapse_angle)

Tp = time_periapse_angle./mean_motion;

% plot(times/3600, Tp)
```

Published with MATLAB® R2013b

```
function accel = J2_accel( pos )
%J2_accel Acceleration due to J2
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

J2 = 0.00108248;
mu = 398600.4; %km3/s2
Re = 6378.145; %km

r = norm(pos);
z = pos(3);
const = 1.5*mu*J2*Re*Re/(r*r*r*r*r);
sin_sq_phi = z*z/(r*r);

accel = const*[5*sin_sq_phi - 1;
    5*sin_sq_phi - 1;
    5*sin_sq_phi - 3].*pos;
end
```

Published with MATLAB® R2013b

```
function plot_oe(oe_vec, times, prefix, P, vec2)
%plot_oe   Plot OEs
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

mu = 3.986004e5;
a = oe_vec(1,:);
e = oe_vec(2,:);
f = oe_vec(6,:);
if nargin < 4
    P = 2*pi/sqrt(mu/(a(1)*a(1)*a(1)));
end
elem_vec = oe_vec;
Tp = get_time_of_periapsis(a,e,f, times);
if nargin == 5
    elem_vec = vec2 - oe_vec;
    Tp = get_time_of_periapsis(vec2(1,:),vec2(2,:),vec2(6,:), times)- Tp;
end
fighandle = figure;
set(fighandle, 'Position', [100, 100, 600, 850])
subplot(6,1,1)
plot(times/3600, elem_vec(1,:))
period_lines(elem_vec(1,:), P, times)
ylabel(sprintf('%sa (km)', prefix))
xlabel('time (hours)')
subplot(6,1,2)
plot(times/3600, elem_vec(2,:))
period_lines(elem_vec(2,:), P, times)
ylabel(sprintf('%se', prefix))
xlabel('time (hours)')
subplot(6,1,3)
plot(times/3600, elem_vec(3,:)*180/pi)
period_lines(elem_vec(3,:)*180/pi, P, times)
ylabel(sprintf('%si (deg)', prefix))
xlabel('time (hours)')
subplot(6,1,4)
plot(times/3600, elem_vec(4,:)*180/pi)
period_lines(elem_vec(4,:)*180/pi, P, times)
ylabel(sprintf('%s\\Omega (deg)', prefix))
xlabel('time (hours)')
subplot(6,1,5)
plot(times/3600, elem_vec(5,:)*180/pi)
period_lines(elem_vec(5,:)*180/pi, P, times)
ylabel(sprintf('%s\\omega (deg)', prefix))
xlabel('time (hours)')
subplot(6,1,6)
plot(times/3600, Tp)
period_lines(Tp, P, times)
ylabel(sprintf('%sT_p (s)', prefix))
xlabel('time (hours)')

end
```

```
function period_lines(vec, P, times)
hold on
ymax = max(vec);
ymin = min(vec);
num_p = floor(times(end)/P);

for ii = 1:num_p
    [~, x] = min(abs(times - P*ii));
    plot([times(x)/3600, times(x)/3600], [ymax, ymin], 'r--')
end

hold off
end
```

Published with MATLAB® R2013b

```
function state_dot = two_body_state_dot(t, state, opts)
%two_body_state_dot   Return state_dot given state. Used for numerical
%integration
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

state_dot = zeros(6,1);
state_dot(1:3) = state(4:6);
mu = 3.986e5; % km3/s2

r_vec = (state(1:3));
r = norm(r_vec);
state_dot(4:6) = -mu * r_vec/(r*r*r);

if isfield(opts, 'J2')
    if opts.J2 == 1
        state_dot(4:6) = state_dot(4:6) + J2_accel( state(1:3) );
    end
end
if isfield(opts, 'drag')
%     opts.drag.Cd
%     opts.drag.A
%     opts.drag.m
    if opts.drag.use == 1
        state_dot(4:6) = state_dot(4:6) + drag_accel( state, opts.drag );
    end
end
```

Published with MATLAB® R2013b