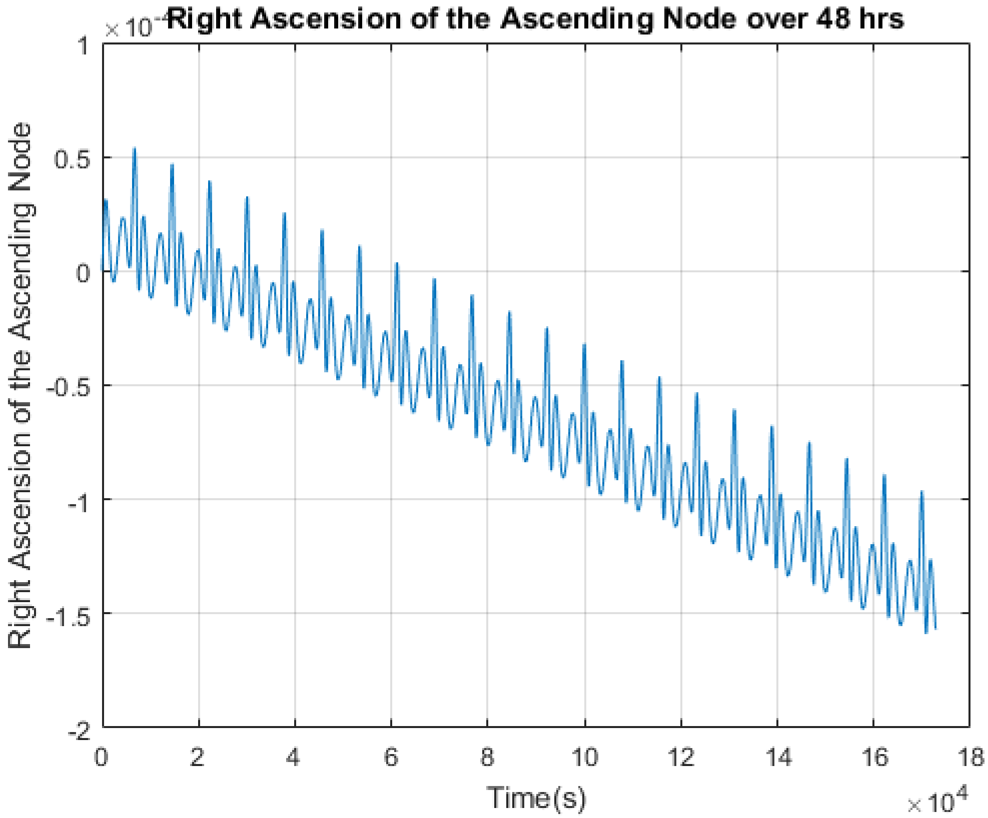
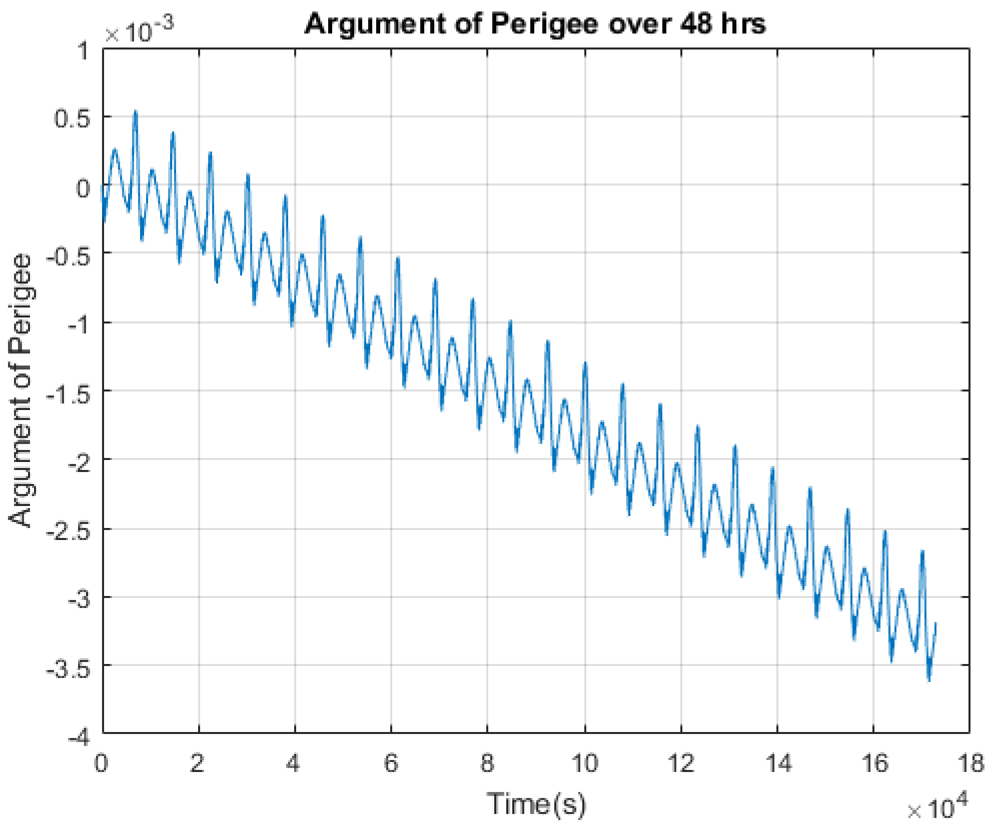
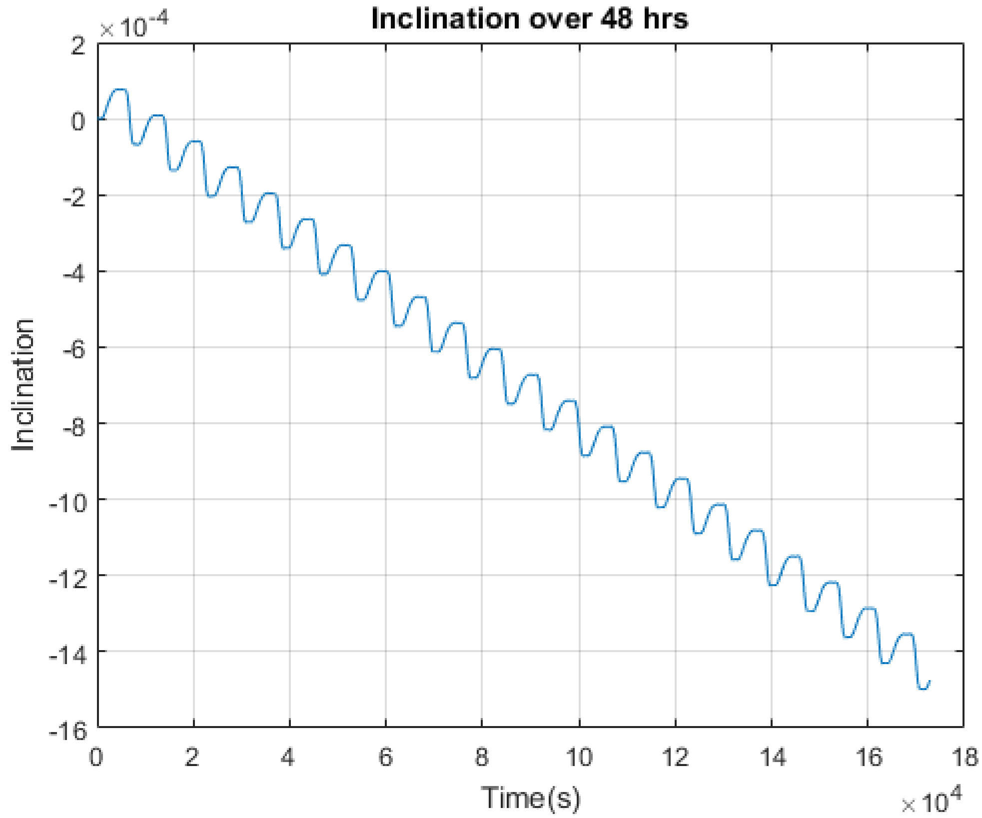
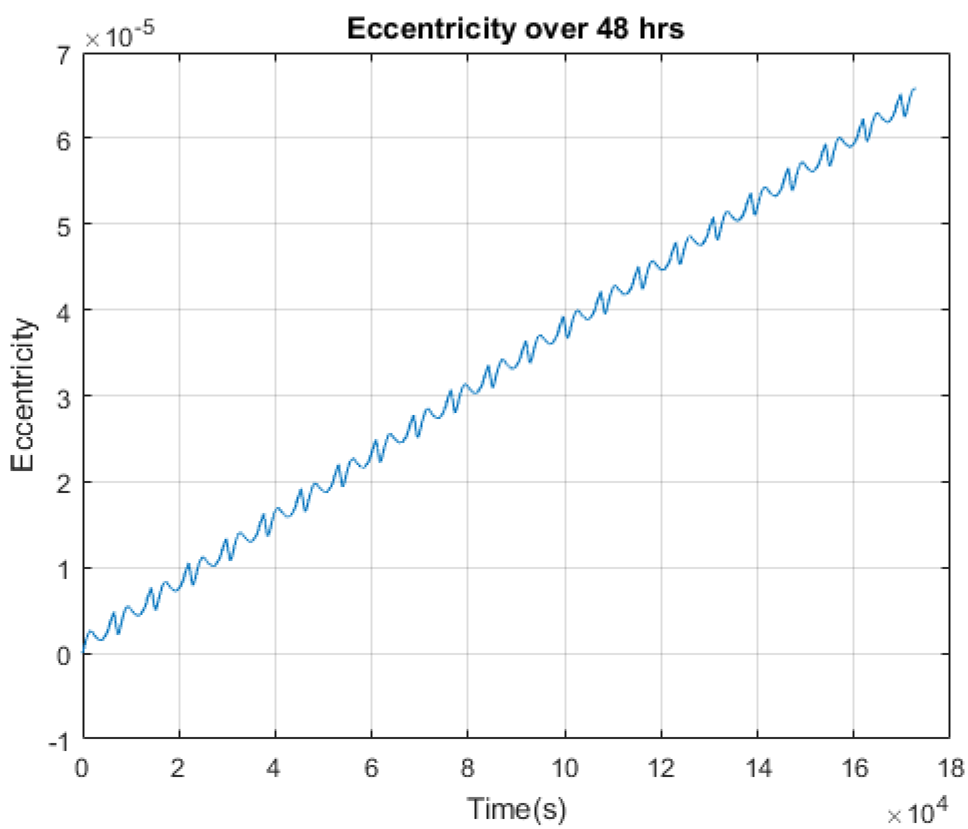
Problem 5;

```
clear;clc;close all;
mu = 3.9860044189e5;
Wo = 45*pi/180;
io = 30*pi/180;
wo = 30*pi/180;
thetao = 40*pi/180;
rp = 400+6378;
ra = 3800+6378;
dt = 48*3600;
eo = (ra-rp)/(ra+rp);
a = (rp+ra)/2;
ho = sqrt(a*mu*(1-eo^2));

[r,v] = OrbElem2StateVec(ho,eo,io,wo,Wo,thetao);

options = odeset('RelTol',1e-6);
[T,Z] = ode45('J3',[0 dt],[r v],options);
for k = 1:length(T)
    [h,e,i,w,W,true_ano] = stateVec2OrbElem(Z(k,(1:3)),Z(k,(4:6)));
    W_k(k) = W;
    w_k(k) = w;
    i_k(k) = i;
    e_k(k) = norm(e);
end

plot(T,W_k-Wo*180/pi); grid on
title('Right Ascension of the Ascending Node over 48 hrs')
xlabel('Time(s)');ylabel('Right Ascension of the Ascending Node')

figure
plot(T,w_k-wo*180/pi);grid on
title('Argument of Perigee over 48 hrs')
xlabel('Time(s)');ylabel('Argument of Perigee')

figure
plot(T,i_k-io*180/pi);grid on
title('Inclination over 48 hrs')
xlabel('Time(s)');ylabel('Inclination')

figure
plot(T,e_k-eo);grid on
title('Eccentricity over 48 hrs')
xlabel('Time(s)');ylabel('Eccentricity')
```


Right Ascension of the Ascending Node over 48 hrs
Right Ascension of the Ascending Node
Time(s)
1
0.5
0
-0.5
-1
-1.5
-2
0
2
4
6
8
10
12
14
16
18
$\times 10^4$


Argument of Perigee over 48 hrs
$\times 10^{-3}$
Argument of Perigee
Time(s)
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
0
2
4
6
8
10
12
14
16
18
$\times 10^4$


Inclination over 48 hrs
×10^-4
2
0
-2
-4
-6
-8
-10
-12
-14
-16
Inclination
0
2
4
6
8
10
12
14
16
18
Time(s)
×10^4


Eccentricity over 48 hrs
×10^-5
7
6
5
4
3
2
1
0
-1
Eccentricity
0
2
4
6
8
10
12
14
16
18
Time(s)
×10^4

Published with MATLAB® R2017a

```
function [vecDeriv] = J3(t,z)

mu = 3.9860044189e5;
r=sqrt(z(1)^2+z(2)^2+z(3)^2);
R = 6378;
J2 = 0.00108263;
J3 = -2.33936*10^(-3)*J2;

C = (1/2)*((J3*mu*R^3)/r^5);

apx = C*((5*z(1)/r)*((7*z(3)^3/r^3)-(3*z(3)/r)));
apy = C*((5*z(2)/r)*((7*z(3)^3/r^3)-(3*z(3)/r)));
apz = C*((35*z(3)^4/r^4)-(30*z(3)^2/r^2)+3);

vecDeriv(1) = z(4);
vecDeriv(2) = z(5);
vecDeriv(3) = z(6);
vecDeriv(4) = -mu/r^3*z(1) + apx;
vecDeriv(5) = -mu/r^3*z(2) + apy;
vecDeriv(6) = -mu/r^3*z(3) + apz;

vecDeriv = vecDeriv';

end
```

```
function [RX,VX,r,v,QXx] = OrbElem2StateVec(h,e,i,w,W,true_ano)
%This function will take orbital elements and compute two state vectors r
and v
% r and v must be 3-D vectors
mu = 3.9860044189e5;
r = (h^2/mu)/(1+e*cos(true_ano))*[cos(true_ano) sin(true_ano) 0]';
v = mu/h*[-sin(true_ano) (e+cos(true_ano)) 0]';
R3_W = [cos(W)  sin(W) 0;
       -sin(W) cos(W) 0;
       0       0      1;];
R1_i = [1 0 0;
        0 cos(i) sin(i);
        0 -sin(i) cos(i);];
R3_w = [ cos(w) sin(w) 0;
        -sin(w) cos(w) 0
        0 0 1;];
QXx = (R3_w) *(R1_i) *(R3_W);

RX = QXx.'*r;
VX = QXx.'*v;

end
```

```
function [h,e,i,w,W,true_ano,N] = stateVec2OrbElem(r,v)
%This function will take two state vectors r and v and compute the six
orbital elements
% r and v must be 3-D vectors
% Currently this is only for Geocentric orbits
mu = 3.9860044189e5;

h = cross(r,v); % specific angular momentum
i = acosd(h(3)/norm(h));  % inclination is hz divided by the magnitude
of h
N = cross([0 0 1],h); % line of nodes is the unit vector k cross h

    if N(2) >= 0
        % Right ascesnsion of the ascending Node
        W = acosd(N(1)/norm(N));
    else
        W = 360-acosd(N(1)/norm(N));
    end
 % eccentricity vector
    vr = dot(r,v)/norm(r);
    e = ((norm(v)^2 - mu/norm(r))*r - norm(r)*vr*v)/mu;

    if e(3) >= 0
        % argument of perigee
        w = acosd(dot(N,e)/(norm(N)*norm(e)));
    else
        w = 360-acosd(dot(N,e)/(norm(N)*norm(e)));
    end

    if vr >= 0
        %true anomaly
        true_ano = acosd(dot(e,r)/(norm(e)*norm(r)));
    else
        true_ano = 360-acosd(dot(e,r)/(norm(e)*norm(r)));
    end


end
```