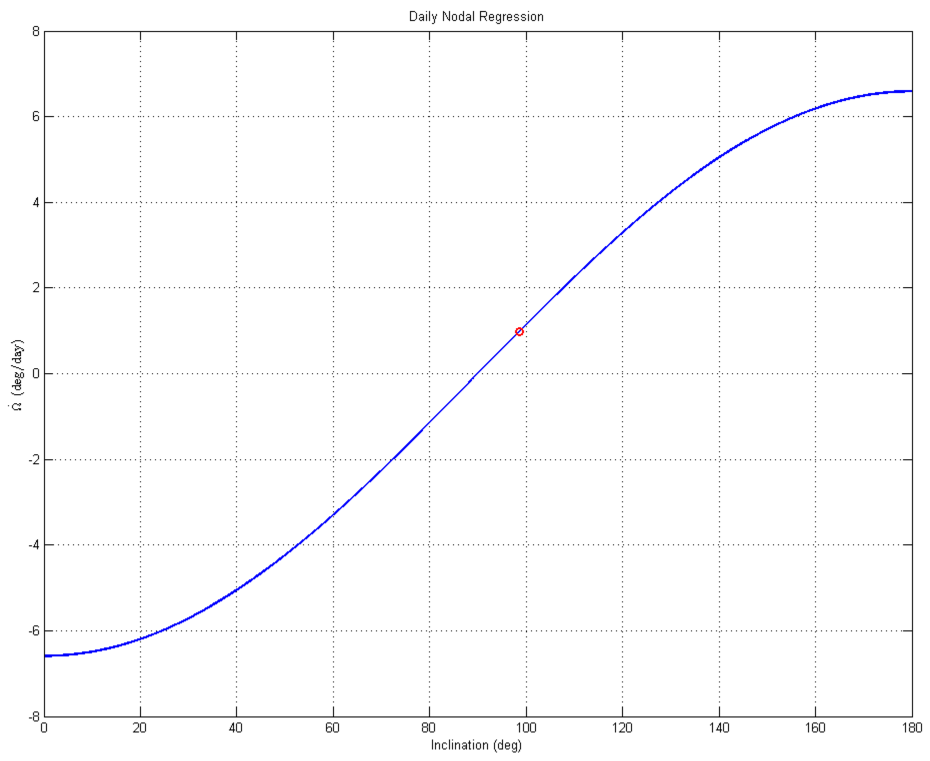
HW8 Problem 1

```
fprintf('\n');
clearvars -except function_list hw_pub toolsPath
close all
CelestialConstants; % import useful constants

r = 6378+800; %km
p = r;
e = 0;
n = sqrt(Earth.mu/r^3)*180/pi; %rad/s

i_vec = 0:0.01:180;
nodal_regression_rate = -3*n*Earth.R^2*Earth.J2/2/p^2*cosd(i_vec)*day2sec;

figure('Position',[0 0 hw_pub.figWidth hw_pub.figHeight])
plot(i_vec, nodal_regression_rate,'LineWidth',2)
xlabel('Inclination (deg)')
ylabel('$\dot{\Omega}$ (deg/day)','interpreter','latex')
title('Daily Nodal Regression')
grid on

sun_sync_node_rate = 360/365.2421897; %deg/day
sun_sync_incl = acosd(sun_sync_node_rate*2*p^2/...
    (-3*n*Earth.R^2*Earth.J2*day2sec));

hold on
plot([sun_sync_incl],[sun_sync_node_rate],'ro','LineWidth',2)

fprintf('Inclination for this orbit to be sun-synchronous: %.2f deg\n',...
    sun_sync_incl)
```

```
Inclination for this orbit to be sun-synchronous: 98.60 deg
```



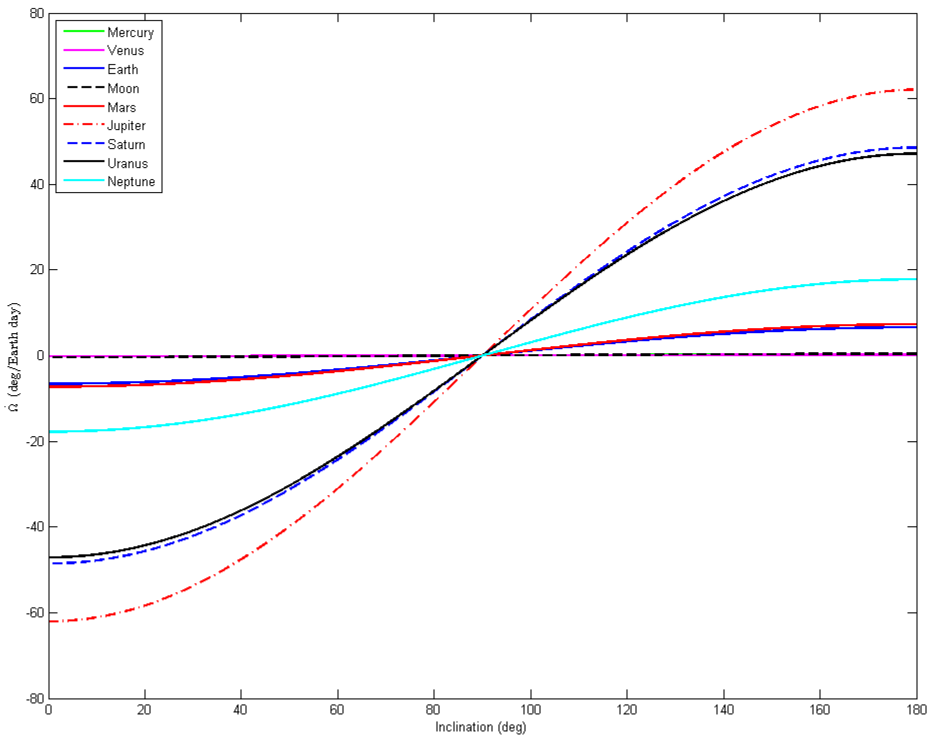
Published with MATLAB® R2013b

HW8 Problem 2

```
fprintf('\n');
clearvars -except function_list hw_pub toolsPath
close all
CelestialConstants; % import useful constants

planets = {Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn,...
    Uranus, Neptune};
line_style = {'g', 'm', 'b', 'k--', 'r', 'r-.', 'b--',...
    'k', 'c'};
h = 800;
figure('Position',[0 0 hw_pub.figWidth hw_pub.figHeight])
for idx = 1:length(planets)
    planet = planets{idx};
    ss_rate{idx} = 360/planet.P_days; %deg/day
    fprintf(['Sun-synchronous nodal regression rate for ' ...
        planet.name ...
        ': %.3e deg/Earth day\n'], ss_rate{idx})
    r = h + planet.R;
    p = r;
    n = sqrt(planet.mu/r^3)*180/pi*day2sec;
    i_vec = 0:0.01:180;
    nodal_regression_rate = -3*n*planet.R^2*planet.J2/2/p^2*cosd(i_vec);
    reg_rate_cell_array{idx} = nodal_regression_rate;
    plot(i_vec, nodal_regression_rate, line_style{idx},'LineWidth',2)
    hold on
    target_i{idx} = acosd(ss_rate{idx}*2*p^2/(-3*n*planet.R^2*planet.J2));
    names{idx} = planet.name;
end
legend(names, 'Location', 'northwest')
xlabel('Inclination (deg)')
ylabel('$\dot{\Omega}$ (deg/Earth day)','interpreter', 'latex')

for idx = 1:length(planets)
    planet = planets{idx};
    fprintf(['\n' planet.name ': Target Inclination: '])
    if ss_rate{idx} > max(reg_rate_cell_array{idx})
        fprintf('Not achievable')
    else
        fprintf('%.2f degrees', target_i{idx})
    end
end
fprintf('\n')
```

```
Sun-synchronous nodal regression rate for Mercury: 4.092e+00 deg/Earth day
Sun-synchronous nodal regression rate for Venus: 1.602e+00 deg/Earth day
Sun-synchronous nodal regression rate for Earth: 9.856e-01 deg/Earth day
Sun-synchronous nodal regression rate for Moon: 1.318e+01 deg/Earth day
Sun-synchronous nodal regression rate for Mars: 5.241e-01 deg/Earth day
Sun-synchronous nodal regression rate for Jupiter: 8.313e-02 deg/Earth day
```

```
Sun-synchronous nodal regression rate for Saturn: 3.350e-02 deg/Earth day

Sun-synchronous nodal regression rate for Uranus: 1.177e-02 deg/Earth day
Sun-synchronous nodal regression rate for Neptune: 6.020e-03 deg/Earth day

Mercury: Target Inclination: Not achievable
Venus: Target Inclination: Not achievable
Earth: Target Inclination: 98.60 degrees
Moon: Target Inclination: Not achievable
Mars: Target Inclination: 94.12 degrees
Jupiter: Target Inclination: 90.08 degrees
Saturn: Target Inclination: 90.04 degrees
Uranus: Target Inclination: 90.01 degrees
Neptune: Target Inclination: 90.02 degrees
```



Published with MATLAB® R2013b

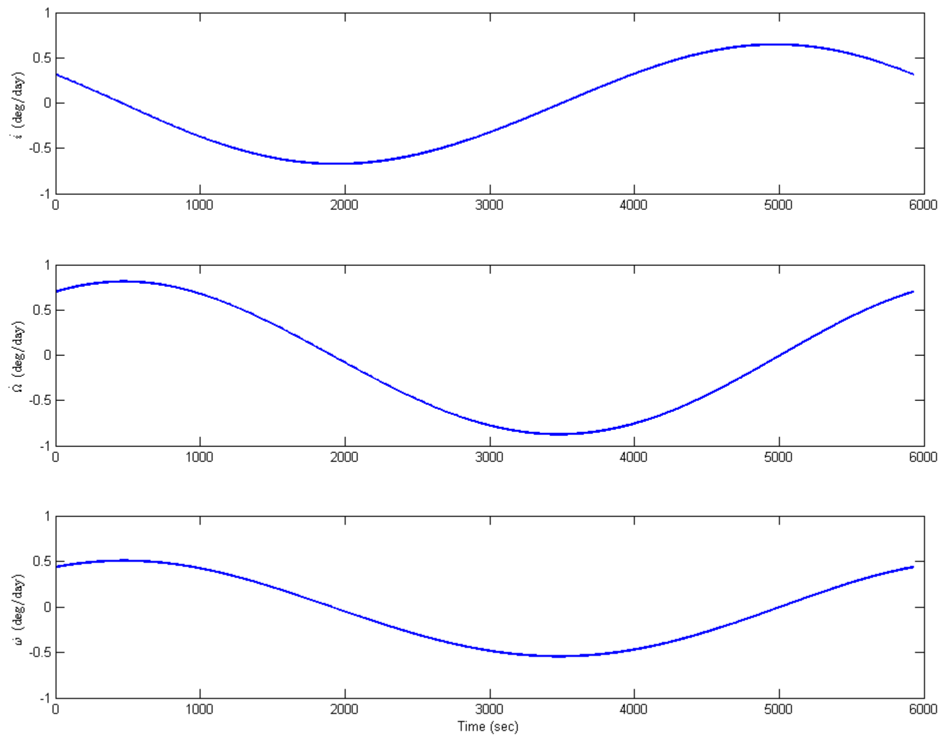
HW8 Problem 3

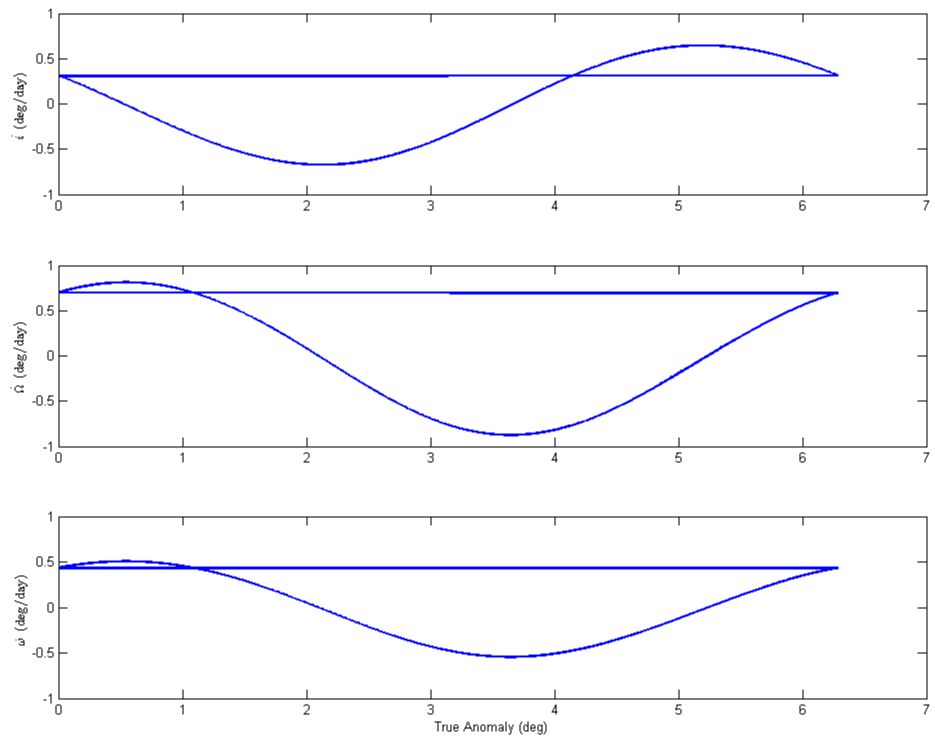
```
fprintf('\n');
clearvars -except function_list hw_pub toolsPath
close all
CelestialConstants; % import useful constants

% Orbit normal vector
normal_vec = @(X) cross(X(1:3),X(4:6))/norm(cross(X(1:3),X(4:6)));

% Anonymous function to calculate 2-body accel
two_body = @(t,X) [X(4);X(5);X(6);...
    -Earth.mu*X(1)/norm(X(1:3))^3;...
    -Earth.mu*X(2)/norm(X(1:3))^3;...
    -Earth.mu*X(3)/norm(X(1:3))^3] + 1e-6*[0;0;0;normal_vec(X)];

hp = 400;
ha = 1000;
a = (ha+hp+2*Earth.R)/2;
e = (ha+Earth.R-a)/a;
i = 51.5*pi/180;
RAAN = 0;
w = 60*pi/180;
f = 0;
P = 2*pi*sqrt(a^3/Earth.mu);

[r,v] = OE2cart(a,e,i,RAAN,w,f,Earth.mu);
X0 = [r;v];

tol=1e-12;
options=odeset('RelTol',tol,'AbsTol',[tol tol tol tol tol tol]);

[t_array,X_array]=ode45(two_body,[0 P],X0,options);

OE_array = zeros(6,length(t_array));
di_dt = zeros(length(t_array),1);
dRAAN_dt = zeros(length(t_array),1);
dw_dt = zeros(length(t_array),1);
for ii = 1:length(t_array)
    [OE_array(1,ii),...
        OE_array(2,ii),...
        OE_array(3,ii),...
        OE_array(4,ii),...
        OE_array(5,ii),...
        OE_array(6,ii)] = cart2OE(X_array(ii,1:3)',X_array(ii,4:6)',Earth.mu);

    r = norm(X_array(ii,1:3));
    h = norm(cross(X_array(ii,1:3),X_array(ii,4:6)));
    di_dt(ii) = r*cos(OE_array(5,ii)+OE_array(6,ii))/...
        (sqrt(Earth.mu/OE_array(1,ii)^3)...
        *OE_array(1,ii)^2*sqrt(1-OE_array(2,ii)^2))*1e-6;
    dRAAN_dt(ii) = r*sin(OE_array(5,ii)+OE_array(6,ii))/...
```

```
            (sqrt(Earth.mu/OE_array(1,ii)^3)...
            *OE_array(1,ii)^2*sqrt(1-OE_array(2,ii)^2)*sin(OE_array(3,ii)))...
            *1e-6;
        dw_dt(ii) = r*cot(OE_array(3,ii))*sin(OE_array(5,ii)+OE_array(6,ii))...
            /(h)*1e-6;

    end
    % plot(t_array,OE_array(3,:));
    % figure
    % plot(t_array,[OE_array(4,OE_array(4,:)<pi), OE_array(4,OE_array(4,:)>=pi)-2*pi])
    % figure
    % plot(t_array,OE_array(5,:));

    fprintf(['a) The inclination, RAAN, and argument of periapse are\n'...
             'directly affected by this force, according to Gaussian VOP.\n'])
    fprintf(['b) The inclination, RAAN, and argument of periapse will\n'...
             'all experience secular drift because energy is constantly\n'...
             'added to the system.\n'])
    fprintf(['c) When the orbit is exagerated, you can see evidence of\n'...
             'secular drift. The rate of change is biased toward either side\n'...
             'of zero over the course of an orbit.\n'])
    figure('Position',[0 0 hw_pub.figWidth hw_pub.figHeight])
    subplot(3,1,1)
    plot(t_array,di_dt*180/pi*day2sec,'LineWidth',2)
    ylabel('$\dot{i}$ (deg/day)','interpreter','latex')
    subplot(3,1,2)
    plot(t_array,dRAAN_dt*180/pi*day2sec,'LineWidth',2)
    ylabel('$\dot{\Omega}$ (deg/day)','interpreter','latex')
    subplot(3,1,3)
    plot(t_array,dw_dt*180/pi*day2sec,'LineWidth',2)
    ylabel('$\dot{\omega}$ (deg/day)','interpreter','latex')
    xlabel('Time (sec)')

    figure('Position',[0 0 hw_pub.figWidth hw_pub.figHeight])
    subplot(3,1,1)
    plot(OE_array(6,:),di_dt*180/pi*day2sec,'LineWidth',2)
    ylabel('$\dot{i}$ (deg/day)','interpreter','latex')
    subplot(3,1,2)
    plot(OE_array(6,:),dRAAN_dt*180/pi*day2sec,'LineWidth',2)
    ylabel('$\dot{\Omega}$ (deg/day)','interpreter','latex')
    subplot(3,1,3)
    plot(OE_array(6,:),dw_dt*180/pi*day2sec,'LineWidth',2)
    ylabel('$\dot{\omega}$ (deg/day)','interpreter','latex')
    xlabel('True Anomaly (deg)')
```

```
a) The inclination, RAAN, and argument of periapse are
directly affected by this force, according to Gaussian VOP.
b) The inclination, RAAN, and argument of periapse will
all experience secular drift because energy is constantly
added to the system.
c) When the orbit is exagerated, you can see evidence of
secular drift. The rate of change is biased toward either side
of zero over the course of an orbit.
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored

Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored
```





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
```

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.J2 = 0.0010826267;
Earth.P_days = 365.2421897; %days
Earth.P_years = 0.99997862; %days
```

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
```

Sun

```
Sun.mu = 1.32712428e11; %km3/s2
```

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
```

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
```

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
```

Saturn

```
Saturn.name = 'Saturn';
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
```

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
```

Celestial units

```
au2km = 149597870.7;
```

Physical constants

```
day2sec = 86400; % sec/day
```

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 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 [a,e,i,RAAN,w,f] = cart2OE( r, v ,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

h = cross(r,v);
n = cross([0;0;1],h);
ecc_vec = ((norm(v)*norm(v)-mu/norm(r))*r - dot(r,v)*v)/mu;

e = norm(ecc_vec);
a = 0;
if e < 1.0
    specific_energy = norm(v)*norm(v)/2-mu/norm(r);
    a = -mu/2/specific_energy;
end

i = acos(h(3)/norm(h));
RAAN = acos(n(1)/norm(n));
if n(2) < 0
    RAAN = 2*pi-RAAN;
end
w = acos(dot(n,ecc_vec)/(norm(n)*norm(ecc_vec)));
if ecc_vec(3) < 0
    w = 2*pi-w;
end
f = acos(dot(ecc_vec,r)/(norm(ecc_vec)*norm(r)));
if dot(r,v) < 0
    f = 2*pi-f;
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
        function_list = [function_list, filename];
    end
end
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
        function_list = [function_list, filename];
    end
end
end
```

Published with MATLAB® R2013b