HW4 Problem 1

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

hp1 = 250; %km
ha1 = 600; %km
hp2 = 2000; %km
ha2 = 5000; %km

a1 = (2*Earth.R + hp1 + ha1)/2;
a2 = (2*Earth.R + hp2 + ha2)/2;

% quick function to compute velocity on the fly:
visviva = @(h,a) sqrt(2*Earth.mu/(Earth.R + h) - Earth.mu/a);
% quick function for pretty output
printout = @(xfer, dv1, dv2, dvtot) fprintf([xfer ' Transfer: ' ...
    'dV1 = ' num2str(dv1, '%.3f') ' km/s, '...
    'dV2 = ' num2str(dv2, '%.3f') ' km/s, '...
    'Total dV = ' num2str(dvtot, '%.3f') ' km/s\n']);

vp1 = visviva(hp1, a1);
va1 = visviva(ha1, a1);
vp2 = visviva(hp2, a2);
va2 = visviva(ha2, a2);
dV_tot_array = [];

% a) initial peri to target apo
a_xfer = (2*Earth.R + hp1 + ha2)/2;
v_xfer_i = abs(visviva(hp1, a_xfer) - vp1);
v_xfer_f = abs(visviva(ha2, a_xfer) - va2);
dV_total = v_xfer_i + v_xfer_f;
printout('P-A', v_xfer_i, v_xfer_f, dV_total);
dV_tot_array = [dV_tot_array dV_total];

% b) initial peri to target peri
a_xfer = (2*Earth.R + hp1 + hp2)/2;
v_xfer_i = abs(visviva(hp1, a_xfer) - vp1);
v_xfer_f = abs(visviva(hp2, a_xfer) - vp2);
dV_total = v_xfer_i + v_xfer_f;
printout('P-P', v_xfer_i, v_xfer_f, dV_total);
dV_tot_array = [dV_tot_array dV_total];

% c) initial apo to target apo
a_xfer = (2*Earth.R + ha1 + ha2)/2;
v_xfer_i = abs(visviva(ha1, a_xfer) - va1);
v_xfer_f = abs(visviva(ha2, a_xfer) - va2);
dV_total = v_xfer_i + v_xfer_f;
printout('A-A', v_xfer_i, v_xfer_f, dV_total);
dV_tot_array = [dV_tot_array dV_total];
```

```
% d) initial apo to target peri
a_xfer = (2*Earth.R + ha1 + hp2)/2;
v_xfer_i = abs(visviva(ha1, a_xfer) - va1);
v_xfer_f = abs(visviva(hp2, a_xfer) - vp2);
dV_total = v_xfer_i + v_xfer_f;
printout('A-P', v_xfer_i, v_xfer_f, dV_total);
dV_tot_array = [dV_tot_array dV_total];

[y,i] = min(dV_tot_array);
fprintf(['The minimum dV is for option ' num2str(i) ', total of '...
    num2str(y,'%.3f') ' km/s\n'])
```

```
P-A Transfer: dV1 = 0.864 km/s, dV2 = 0.372 km/s, Total dV = 1.236 km/s
P-P Transfer: dV1 = 0.341 km/s, dV2 = 0.920 km/s, Total dV = 1.260 km/s
A-A Transfer: dV1 = 0.955 km/s, dV2 = 0.290 km/s, Total dV = 1.245 km/s
A-P Transfer: dV1 = 0.435 km/s, dV2 = 0.827 km/s, Total dV = 1.262 km/s
The minimum dV is for option 1, total of 1.236 km/s
```

Published with MATLAB® R2013b

HW4 Problem 2

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

r_eci = [-5634 -2645 2834]'; % km
theta_GST = 82.75*pi/180;

[lat, lon, alt] = ECEF2latlonalt(eci2ecef(r_eci, theta_GST));
fprintf('Latitude: %.2f deg\n', lat*180/pi);
fprintf('Longitude: %.2f deg\n', lon*180/pi);
fprintf('Altitude: %.2f km\n', alt);

% ecef2eci(latlonalt2ECEF(lat, lon, alt), theta_GST)
```

```
Latitude: 24.48 deg
Longitude: 122.40 deg
Altitude: 460.83 km
```

Published with MATLAB® R2013b

HW4 Problem 3

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

lat = 40.01*pi/180; % rad
lon = 254.83*pi/180; % rad
h = 1.615; %km
theta_GST = 103*pi/180; % rad

r_eci = ecef2eci(latlonalt2ECEF(lat, lon, h), theta_GST);
fprintf('r_eci = %.2f \n', r_eci(1));
fprintf('        %.2f km\n', r_eci(2));
fprintf('        %.2f \n', r_eci(3));
```

```
r_eci = 4882.85
        -185.02 km
        4101.59
```

Published with MATLAB® R2013b

HW4 Problem 4

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

% Boulder, CO
lat = 40.01*pi/180; % rad
lon = 254.83*pi/180; % rad
h = 1.615; %km

r_sat_ecef = [-1681;-5173;4405];

topo = ecef2topo( r_sat_ecef, lat, lon, h);
fprintf('Azimuth: %.2f deg\n', topo(1)*180/pi);
fprintf('Elevation: %.2f deg\n', topo(2)*180/pi);
fprintf('Range: %.2f km\n', topo(3));
```

```
Azimuth: 246.15 deg
Elevation: 64.41 deg
Range: 680.22 km
```

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.mu = 3.986e5; %km3/s2
Earth.R = 6378; %km
```

Celestial units

```
au2km = 149597870.7;
```

Physical constants

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

Published with MATLAB® R2013b

```
function r_ecef = eci2ecef( r_eci, greenwich_time)
%eci2ecef return ECEF position from ECI coords
% units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Just rotate about the z axis.
r_ecef = Euler2DCM('3', [greenwich_time]) * r_eci;
```

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 [lat, lon, alt] = ECEF2latlonalt( r_ecef)
%ECEF2latlonalt return geocentric latitude, longitude, and altitude
% from ECEF coords.
% units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Get some useful constants
CelestialConstants;

% Just subtract the earth radius to get altitude
r = norm(r_ecef);
alt = r - Earth.R;

lat = asin(r_ecef(3)/r);
% y/x = tan(lon)
lon = atan2(r_ecef(2), r_ecef(1));
```

Published with MATLAB® R2013b

```
function r_ECEF = latlonalt2ECEF( lat, lon, alt)
%latlonalt2ECEF return ECEF coords from geocentric latitude, longitude,
% and altitude.
% units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Get some useful constants
CelestialConstants;

r = alt + Earth.R;

x = r*cos(lat)*cos(lon);
y = r*cos(lat)*sin(lon);
z = r*sin(lat);

r_ECEF = [x;y;z];
```

Published with MATLAB® R2013b

```
function r_eci = ecef2eci( r_ecef, greenwich_time)
%ecef2eci return ECI position from ECEF coords
% units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Just a negative rotation about the z axis.
r_eci = Euler2DCM('3', [-greenwich_time]) * r_ecef;
```

Published with MATLAB® R2013b

```matlab
function topo = ecef2topo( r_ecef, lat, lon, alt)
%ecef2topo return topocentric parameters from ECEF position, and the
% origin's geocentric latitude, longitude, and altitude.
% Topo params are azimuth, elevation, and range (km)
% angle units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Get some useful constants
CelestialConstants;

% ECEF of tracking station/topo origin
O_ECEF = latlonalt2ECEF(lat, lon, alt);

r_topo = norm(r_ecef - O_ECEF);

% convert both r_ecef and O_ECEF to East-North-Up coords (Misra/Enge)
% Ideally this should be done with geodetic latitude... FIXME later
ECEF2ENU_DCM = Euler2DCM('31',[lon+pi/2, pi/2-lat]);
r_enu = ECEF2ENU_DCM*(r_ecef - O_ECEF);
el = asin(r_enu(3)/norm(r_enu));
az = atan2(r_enu(1),r_enu(2));
if az < 0
    az = az + 2*pi;
end

topo = [az; el; r_topo];
```

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