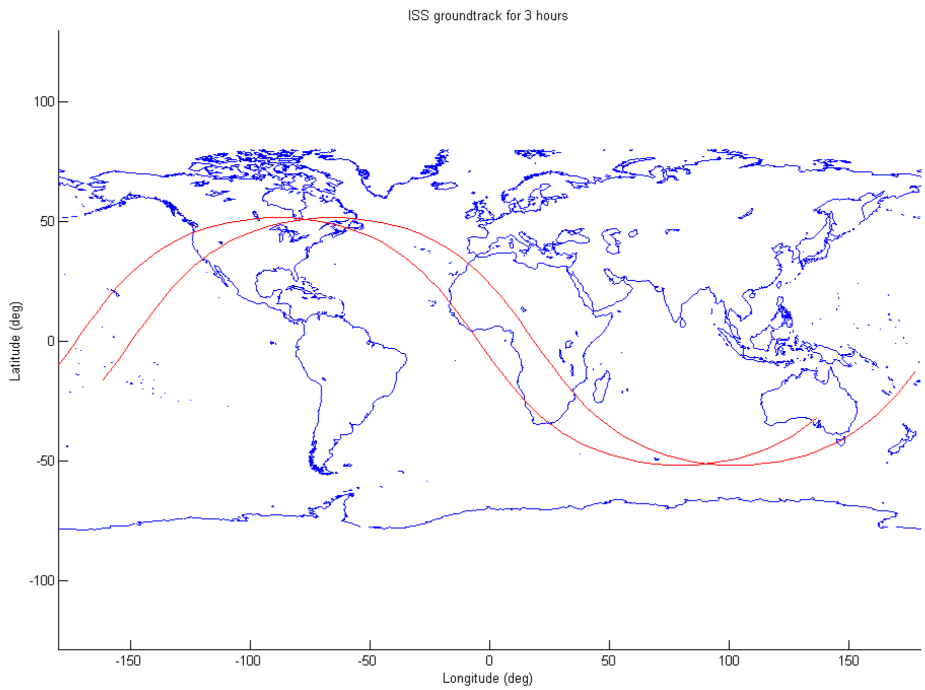
HW6 Problem 1

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

% Oct 10, 2014
hr = .86929405* 24;
minute = (hr - floor(hr))*60;
second = (minute - floor(minute))*60;
MDT_offset = -6;
% The epoch is 10-10-2014 floor(hr):floor(min):sec GMT
fprintf(['Epoch is 10-10-2014 ' num2str(floor(hr)) ':' ...
    num2str(floor(minute)) ':' num2str((second)) ' GMT\n']);
fprintf(['        = 10-10-2014 ' num2str(floor(hr)+MDT_offset) ':' ...
    num2str(floor(minute)) ':' num2str((second)) ' MDT\n']);
fprintf('\t(used www.epochconverter.com to find day from day-of-year\n');
time_to_init = 19-(hr+MDT_offset);
fprintf(['Must propagate ' num2str(time_to_init) ' hours (' ...
    num2str(time_to_init/24) ' days) to reach \n\t10-10-2014 19:00:00 MDT\n']);

load('Map.dat');

i = 051.6467 * pi/180;
RAAN = 246.1653 * pi/180;
e = 0.0002545;
w = 242.3213 * pi/180;
M = 215.0173 * pi/180;
n = 15.50348149909277 * 2*pi /24/3600; % rad/s
a = (Earth.mu/n/n)^(1/3);

% Get Greenwich Mean Siderial Time for the beginning of the scenario
JD = computeJD(2014, 10, 11, 1, 0, 0);

dt = 60; % sec
prop_time = 3*3600; %sec
time_vec = 0:dt:prop_time;
num_pts = length(time_vec);
geocen_lat_vec= zeros(length(time_vec),1);
lat_vec = zeros(length(time_vec),1);
lon_vec = zeros(length(time_vec),1);
r_ecef_store = zeros(3,length(time_vec));
cnt = 0;

% First, propagate to the init time.
M = M + n*time_to_init*3600;
while M > 2*pi %Assumes dt < Period of orbit...
    M = M - 2*pi;
end

% Now propagate for the prescribed time.
```

```
for t = time_vec
    cnt = cnt + 1;
    if t > 0
        M = M + n*dt;
        % Increment date to find GMT angle
        JD = JD + dt/86400;
    end
    if M > 2*pi %Assumes dt < Period of orbit...
        M = M - 2*pi;
    end

    % Get true anom
    E = M2E(M,e);
    f = E2f(E,e);

    % Cartesian ECI state
    [r,v] = OE2cart(a,e,i,RAAN,w,f,Earth.mu);

    % Find the Greenwich solar angle
    greenwich_time = computeLocalSiderealTime(JD,0,Earth.spin_rate);

    % Coords in ECEF
    r_ecef = eci2ecef(r, greenwich_time);
    % Save it for Problem 2
    r_ecef_store(:,cnt) = r_ecef;

    % Resultant lat(geocentric), lon, altitude
    [lat, lon, alt] = ECEF2latlonalt(r_ecef);

    % Record the result for the groundtrack plot
    geocen_lat_vec(cnt) = lat;
    lat_vec(cnt) = atan2(tan(lat),(1-Earth.flattening*Earth.flattening)); %FIXME!!
    lon_vec(cnt) = lon;
end

discontinuity_idx = [];
for ii = 2:length(lon_vec)
    if abs(lon_vec(ii) - lon_vec(ii-1)) > pi/2
        discontinuity_idx = [discontinuity_idx ii];
    end
end

for ii = fliplr(discontinuity_idx);
    lon_vec = [lon_vec(1:ii-1); NaN; lon_vec(ii:end)];
    lat_vec = [lat_vec(1:ii-1); NaN; lat_vec(ii:end)];
    geocen_lat_vec = [geocen_lat_vec(1:ii-1); NaN; geocen_lat_vec(ii:end)];
end

figure('OuterPosition', [0 50 hw_pub.figWidth hw_pub.figHeight])
hold on
plot(Map(:,1), Map(:,2))
plot(lon_vec*180/pi, lat_vec*180/pi, 'r')
% plot(lon_vec*180/pi, geocen_lat_vec*180/pi, 'g--')
title('ISS groundtrack for 3 hours')
```

```
xlim([-180,180])
xlabel('Longitude (deg)')
ylim([-90, 90])
ylabel('Latitude (deg)')
axis equal
```

```
Epoch is 10-10-2014 20:51:47.0059 GMT
       = 10-10-2014 14:51:47.0059 MDT
 (used www.epochconverter.com to find day from day-of-year
Must propagate 4.1369 hours (0.17237 days) to reach
 10-10-2014 19:00:00 MDT
```

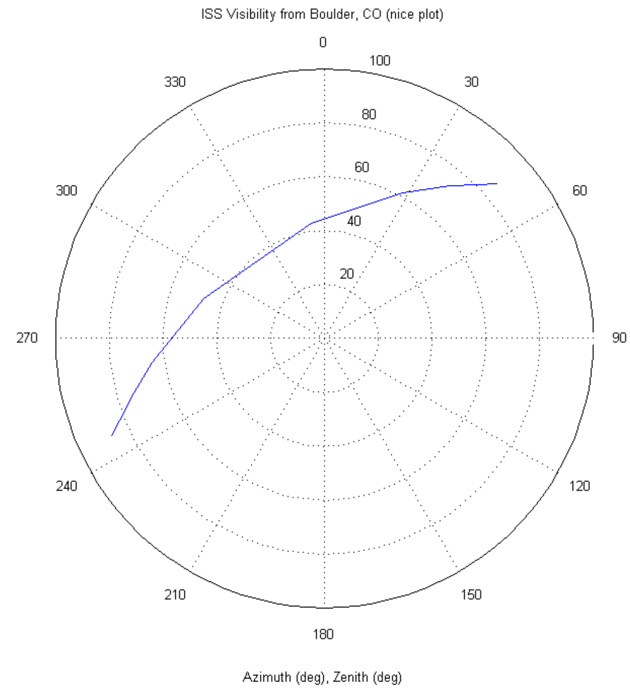


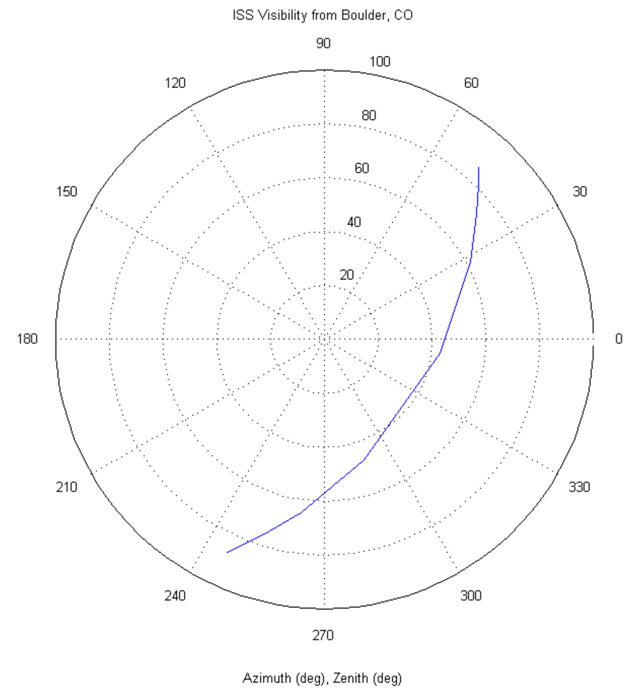
Published with MATLAB® R2013b

HW6 Problem 2

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

lat = 40.01 * pi/180;
lon = 254.83 * pi/180;
alt = 1.615;
topo_store = zeros(3,length(time_vec));

% Store off all the topographic coord vectors from the r_ecef info in the
% last problem
for ii = 1:length(r_ecef_store(1,:))
    topo_store(:,ii) = ecef2topo( r_ecef_store(:,ii), lat, lon, alt);
end

% Get the first pass indices. Elevation is the second element of each
% vector. Look for them being > 0.
first_idx = find(topo_store(2,:) > 0,1);
last_idx =  find(topo_store(2,first_idx:end) <= 0,1)+first_idx-2;

% Warning, this part assumes dt = 1 minute, and that the first pass happens
% in the first hour of the scenario.
first_time = (first_idx - 1); % minutes
last_time = (last_idx - 1); % minutes
fprintf(['Pass begins: 10-11-2014 0' num2str(1) ':' ...
    num2str(first_time) ':00 GMT\n']);
fprintf(['Pass ends:   10-11-2014 0' num2str(1) ':' ...
    num2str(last_time) ':00 GMT\n']);
fprintf(['Pass duration is approx ' num2str(last_time-first_time) ...
    ' minutes\n'])
fprintf(['max elevation is ' ...
    num2str(max(topo_store(2,first_idx:last_idx)*180/pi)) ...
    ' degrees\n'])

% A plot with North being up on the plot, clockwise angles
figure('OuterPosition', [0 50 hw_pub.figWidth hw_pub.figHeight])
plotVisZenith(topo_store(1,first_idx:last_idx), ...
    topo_store(2,first_idx:last_idx))
title('ISS Visibility from Boulder, CO (nice plot)')

% regular ol' polar plot
figure('OuterPosition', [0 50 hw_pub.figWidth hw_pub.figHeight])
polar(topo_store(1,first_idx:last_idx), ...
    90-topo_store(2,first_idx:last_idx)*180/pi)
xlabel('Azimuth (deg), Zenith (deg)')
title('ISS Visibility from Boulder, CO')
```

```
Pass begins: 10-11-2014 01:16:00 GMT

Pass ends:   10-11-2014 01:25:00 GMT
Pass duration is approx 9 minutes
max elevation is 46.8494 degrees
```





Published with MATLAB® R2013b

CelestialConstants

Table of Contents

Description 1
Earth 1
Venus 1
Mars 1
Jupiter 1
Celestial units 1
Physical constants 2

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
Earth.a = 149598023; %km
Earth.spin_rate = 7.2921158553e-05; %rad/s
Earth.flattening = 1/298.25722; %WGS-84

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

Venus

```
Venus.a = 108208601; %km
```

Mars

```
Mars.a = 227939186; %km
```

Jupiter

```
Jupiter.a = 778298361; %km
```

Celestial units

```
au2km = 149597870.7;
```

Physical constants

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

Published with MATLAB® R2013b

```
function JD = computeJD(yr,mo,day,hr,mn,sec, is_leap_sec_day)
%computeJD return Julian date for UT1
% ONLY VALID BETWEEN 1900-2100!!!!
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

sec_denom = 60;
if nargin < 6
    fprintf('ERROR - not enough arguments for computeJD.\n')
    JD = -1;
    return;
elseif nargin == 7
    if is_leap_sec_day == 1
        sec_denom = 61;
    end
end
JD = 367*yr -floor(7/4*(yr+floor((mo+9)/12)))...
    + floor(275*mo/9) + day + 1721013.5 ...
    + ((sec/sec_denom + mn)/60 + hr)/24;
```

Published with MATLAB® R2013b

```
function E = M2E( M, e )
%M2E Mean anom (M) to eccentric anom (E)
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

tol = 1e-5;

if (M < 0 && M > -pi) || M > pi
    E_1 = M - e;
else
    E_1 = M + e;
end

E = E_1 + tol + 1;
while abs(E_1-E) > tol
    E = E_1;
    E_1 = E - (E - e*sin(E) - M)/(1 - e*cos(E));
end
```

Published with MATLAB® R2013b

```
function f = E2f( E, e )
%E2f Eccentric anomaly (E) to true anomaly (f)
% Only valid for e < 1
% units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Vallado eqn 2-10
f = acos((cos(E) - e)/(1-e*cos(E)));
if E > pi
    f = 2*pi - f;
end
```

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 LST = computeLocalSiderealTime(JD, lon, spin_rate)
%computeLocalSiderealTime return local sidereal time
% units in radians
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Couldn't get 3-45 and 3-46 to work :(
T_UT1 = ((JD)-2451545)/36525;

% GMST0 = 100.4606184 ...
%     + 36000.77005361 * T_UT1 ...
%     + 0.00038793 * T_UT1 * T_UT1 ...
%     - 2.6e-8 * T_UT1 * T_UT1 * T_UT1;
% GMST0 = GMST0 * pi/180;

GMST = 67310.54841 ...
    + (876600*3600+8640184.812866) * T_UT1 ...
    + 0.093104 * T_UT1 * T_UT1 ...
    - 6.2e-6 * T_UT1 * T_UT1 * T_UT1;
% GMST = GMST * pi/3600/12;
GMST = GMST-86400*floor(GMST/(86400));
GMST = GMST * pi/3600/12;

% GMST = GMST-2*pi*floor(GMST/(2*pi));
% GMST = GMST-360*floor(GMST/(360));

% GMST = GMST0 + spin_rate*(JD - floor(JD));

LST = GMST + lon;
```

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

```matlab
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 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 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 plotVisZenith(az, el)
%plotVisZenith Make a nice az/ze plot!
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% To go from Matlab's orientation to a clockwise position with north being
% up, need to reflect about 45 degree line (bisecting first and 3rd
% quadrants)

% Reflect about 45 degrees
fixed_az = pi/2 - az;

% Do the plot
polar(fixed_az, 90 - el * 180/pi)

% Grab the handle for the text. Set up the vectors of current, temp, and
% final labels
text = findall(gca, 'type', 'text');
text_angles = 0:30:330;
tmp_text = 'abcdefghijkl';
new_text_angles = [90:-30:0 330:-30:120];
% Set them to a temporary var to avoid redundantly and wrongly setting some
% labels
for ang = text_angles
    hText = findall(text, 'string', num2str(ang));
    set(hText, 'string', tmp_text(find(text_angles == ang,1)))
end
% Set them to the real thing
for ang = tmp_text
    hText = findall(text, 'string', num2str(ang));
    set(hText, 'string', num2str(new_text_angles(find(tmp_text == ang,1))))
end
xlabel('Azimuth (deg), Zenith (deg)')
```

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