ASEN 5050 – Homework #4

Edward Meletyan

[Matlab code attached at the end]

1. Hohmann transfer using the following process:
   a. Determine semi-major axis of transfer orbit using Equation (1).
   b. Calculate velocities using Equation (2) where r is either periapse or apoapse depending on Transfer Option.
      i. Calculate velocity in the initial orbit at point 'a'.
      ii. Calculate velocity required to enter transfer orbit at point 'a'.
      iii. Calculate velocity to enter final orbit at point 'b'.
      iv. Calculate velocity in the transfer orbit at point 'b'.
   c. Total change in velocity and transfer time can be found using Equation (3) – (6).

$$a_{trans} = \frac{r_i + r_f}{2} \tag{1}$$

$$v = \sqrt{\frac{2\mu}{r} - \frac{\mu}{a}} \tag{2}$$

$$\Delta v_a = v_{trans_a} - v_i \tag{3}$$

$$\Delta v_b = v_f - v_{trans_b} \tag{4}$$

$$\Delta v_{total} = |\Delta v_a| + |\Delta v_b| \tag{5}$$

$$\tau_{trans} = \pi\sqrt{\frac{a_{trans}^3}{\mu}} \tag{6}$$

Transfer Option	$\boldsymbol{\Delta v_a}$ (km/s)	$\boldsymbol{\Delta v_b}$ (km/s)	$\boldsymbol{\Delta v_{total}}$ (km/s)
P → A	0.864	0.373	1.236
P → P	0.341	0.920	1.261
A → A	0.955	0.290	1.245
A → P	0.435	0.827	1.262

Transfer option "P → A" requires the least Δv.

2. Convert ECI position to an ECEF position then to geocentric latitude, longitude, and latitude using the following procedure:
   a. Rotate the ECI position vector to ECEF coordinate frame using ECI2ECEF.m and input $\theta_{GST} = 82.75^o$. This function yields the ECEF position vector:
   $$\vec{r}_{ECEF} = \begin{bmatrix} 6176.6 \\ -766.8 \\ 2834 \end{bmatrix} \frac{km}{s}$$
   b. Use the ECEF2LATLON.m function to convert the ECEF position vector to geodetic latitude, longitude, and altitude.
   c. Convert the geodetic latitude to geocentric latitude using:
   $$\tan(\phi_{gc}) = (1 - e_{Earth}^2)\tan(\phi_{gd}) = (1 - 0.8182^2)\tan(24.617^o)$$
   d. Yields the following results:

$$Geocentric\ latitude, \phi_{gc} = 8.6126^o$$

$$Longitude, \lambda = -7.0768^o$$

$$Altitude, h = 464.375\ km$$

3. Convert the given geodetic latitude, longitude, and elevation to an ECI position vector using the following procedure:
   a. Use the LATLON2ECEF.m function to compute the ECEF position vector from the give latitude, longitude, and altitude information. This function outputs the following ECEF position vector:

$$\vec{r}_{ECEF} = \begin{bmatrix} -1278.7 \\ -4716.2 \\ 4101.7 \end{bmatrix} km$$

   b. Now use ECEF2ECI.m function with θ_{GST} and $\vec{r}_{ECEF}$ as inputs to rotate the ECEF vector to the ECI frame using an R3 DCM and a negative Greenwich Sidereal Time. This function yields the following ECI position vector:

$$\boxed{\vec{r}_{ECI} = \begin{bmatrix} 4883.0 \\ -185.0 \\ 4101.7 \end{bmatrix} km}$$

4. Compute the azimuth, elevation, and range using the following procedure:
   a. Calculate the position of the tracking station using the LATLON2ECEF.m function using the given latitude, longitude, and elevation of Boulder, CO. This function yields the following ECEF location of the tracking station:

$$\vec{r}_{Boulder_{ECEF}} = \begin{bmatrix} -1278.7 \\ -4716.2 \\ 4101.7 \end{bmatrix} km$$

   b. Calculate the ECEF vector between the tracking station and the satellite in ECEF coordinates by taking the difference of the station position and the satellite position:

$$\vec{r}_{sat_{Boulder}} = \vec{r}_{sat_{ECEF}} - \vec{r}_{Boulder_{ECEF}} = \begin{bmatrix} -402.29 \\ -456.82 \\ 303.32 \end{bmatrix} km$$

   c. The azimuth, elevation, and range of the satellite relative to Boulder can be found using the following equations (r_x, r_y, r_z are the x, y, and z components of the vector between the tracking station and the satellite):

$$Azimuth = \operatorname{atan}\left(\frac{r_y}{r_x}\right) = -131.37^o$$

$$Elevation = \operatorname{atan}\left(\frac{r_z}{\sqrt{r_x^2 + r_y^2}}\right) = 26.49^o$$

$$Range = \sqrt{r_x^2 + r_y^2 + r_z^2} = 680.09\ km$$

```matlab
%% Ed Meletyan
%  ASEN 5050
%  Homework #4

%% Problem 1----------------------------------------------------------
clc; clear;

% Given

% Periapsis/Apoapsis altitude of initial orbit
hp1 = 250;   % km
ha1 = 600;   % km
% Periapsis/Apoapsis altitude of final orbit
hp2 = 2000;  % km
ha2 = 5000;  % km

% Earth radius
R_E = 6378.137;  % km
% Earth gravitational parameter
MU  = 398600.4418;

% Analysis

% Calculate the semi-major axes of the initial and final orbits
rp1 = hp1 + R_E;
ra1 = ha1 + R_E;
a1  = (rp1 + ra1) / 2;
rp2 = hp2 + R_E;
ra2 = ha2 + R_E;
a2  = (rp2 + ra2) / 2;

% Cases corresponding to Transfer Option
switch 4
    case 1
        % P - A
        rInitial   = rp1;
        rFinal     = ra2;
    case 2
        % P - P
        rInitial   = rp1;
        rFinal     = rp2;
    case 3
        % A - A
        rInitial   = ra1;
        rFinal     = ra2;
    case 4
        % A - P
        rInitial   = ra1;
        rFinal     = rp2;
end

% Calculate Hohmann Transfer
[aTrans, tauTrans, dva, dvb, dv] = ...
    HOHMANNTRANSFERELLIPTIC(rInitial, rFinal, a1, a2, MU);
```

```matlab
%% Problem 2-------------------------------------------------------------
clc; clear;

% Given

% Greenwich Sidereal Time
theta_gst = rad2deg(82.75);
% Eccentricity of Earth
e_E       = 0.81819221456;
% Position in ECI
r_eci     = [-5634; -2645; 2834];

% Analysis

% Convert ECI to ECEF frame
r_ecef = ECI2ECEF(r_eci, theta_gst);

% Determine lat/lon/alt of ECEF vector
[phi_gd, lambda, h] = ECEF2LATLON(r_ecef);
% Compute Geocentric latitude
phi_gc = atan((1 - e_E^2) * tan(phi_gd));


%% Problem 3-------------------------------------------------------------
clc; clear;

% Given

% Greenwhich Sidereal Time
theta_gstBoulder = deg2rad(103);
% Geodetic Latitude of Boulder
phiBoulder       = deg2rad(40.01);
% Longitude of Boulder
lambdaBoulder    = deg2rad(254.83);
% Altitude of Boulder
hBoulder         = 1.615;  % km

% Analysis

% Find ECEF position of Boulder
ecef_Boulder = LATLON2ECEF(phiBoulder, lambdaBoulder, hBoulder);
% Rotate to ECI
eci_Boulder = ECEF2ECI(ecef_Boulder, theta_gstBoulder);


%% Problem 4-------------------------------------------------------------
clc; clear;

% Given

% Satellite position in ECEF
r_sat            = [-1681, -5173, 4405]';  % km
% Geodetic Latitude of Boulder
phiBoulder       = deg2rad(40.01);
```

```
% Longitude of Boulder
lambdaBoulder    = deg2rad(254.83);
% Altitude of Boulder
hBoulder         = 1.615;  % km

% Analysis

% Calculate azimuth, elevation, and range
r_topo = ECEF2TOPO(r_sat, phiBoulder, lambdaBoulder, hBoulder);

----------------------------------------------------------

function [aTrans, tauTrans, dva, dvb, dv] = ...
    HOHMANNTRANSFERELLIPTIC(rInitial, rFinal, aInitial, aFinal, MU)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Computes velocities required to achieve a two-burn Hohmann transfer
%%% between two ellipitcal orbits. Assume:
%%%     - Burns only occur at 0 deg or 180 deg
%%%     - Semi-major axes must be aligned
%%%
%%% Input:  rInitial    -   Initial orbit radius
%%%           rFinal    -   Final orbit radius
%%%         aInitial    -   Semi-major axis of starting orbit
%%%           aFinal    -   Semi-major axis of destination orbit
%%%               MU    -   Gravitational parameter km^3/s^2
%%%
%%% Output:   aTrans    -   Semi-major axis of transfer orbit
%%%         tauTrans    -   Transfer time
%%%              dva    -   Burn 1 to get out of initial orbit
%%%              dvb    -   Burn 2 to achieve final orbit
%%%               dv    -   Total change in velocity
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Transfer orbit semi-major axis
aTrans   = (rInitial + rFinal) / 2;

% Tangential velocity before 1st burn
vInitial = sqrt(2 * MU / rInitial - MU / aInitial);

% Velocity required to enter transfer orbit
vTransA  = sqrt(2 * MU / rInitial - MU / aTrans);

% Velocity required to enter destination orbit
vFinal   = sqrt(2 * MU / rFinal - MU / aFinal);

% Velocity in transfer orbit before 2nd burn
vTransB  = sqrt(2 * MU / rFinal - MU / aTrans);

% 1st Burn
dva      = vTransA - vInitial;

% 2nd Burn
dvb      = vFinal - vTransB;
```

```
% Total velocity change
dv       = abs(dva) + abs(dvb);

% Transfer time
tauTrans = pi * sqrt(aTrans^3 / MU);
end

-----------------------------------------------------------

function pos_ecef = ECI2ECEF(pos_eci, theta_gst)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Rotate inertial coordinates to Earth-centered Earth-fixed
%%%
%%% Input:   pos_eci    -   Position in ECI coordinates
%%%        theta_gst    -   Greenwich Sidereal Time (rad)
%%%
%%% Output: pos_ecef    -   Position in ECEF coordinates
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pre-assign sin/cos functions
st = sin(theta_gst);
ct = cos(theta_gst);

% Rotation about z-axis
R3 = [ ct st 0;
      -st ct 0;
       0  0  1];

% Rotate ECI vector by Greenwich Sidereal Time
pos_ecef = R3 * pos_eci;

end

-----------------------------------------------------------

function [phi, lambda, h] = ECEF2LATLON(pos_ecef)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Compute the geodetic latitude, longitude, and altitude from ECEF
%%% position.
%%%
%%% Input:  pos_ecef    -   Position in ECEF coordinates
%%%
%%% Output:      phi    -   Geodetic latitude (rad)
%%%           lambda    -   Longitude (rad)
%%%                h    -   Altitude
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define Earth constants
R_E = 6378.137;         % Radius (km)
e_E = 0.081819221456;   % Eccentricity
```

```
% Decompse position vector
rx = pos_ecef(1);
ry = pos_ecef(2);
rz = pos_ecef(3);

% Equatorial component
rd = sqrt(rx^2 + ry^2);

% Calculate longitude
sa = ry / rd;
ca = rx / rd;
lambda = atan(sa / ca);

% Iterate to solve for geodetic latitude
tol = 1e-8;

% First guess
phi_init = atan(rz / rd);
phi      = phi_init + 1;

while abs(phi - phi_init) >= tol
    phi_init = phi;
    C        = R_E / sqrt(1 - e_E^2 * sin(phi_init)^2);

    % Output final geodetic latitude
    phi      = atan((rz + C * e_E^2 * sin(phi_init)) / rd);
end

% Calculate altitude
if phi <= 1
    h = rz / sin(phi) - C * (1 - e_E^2);
else
    h =  rd / cos(phi) - C;
end

end


----------------------------------------------------------

function pos_ecef = LATLON2ECEF(phi_gd, lambda, h)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Compute ECEF position on Earth from lat, lon, and altitude
%%%
%%% Input:    phi_gd    -   Geodetic latitude (rad)
%%%           lambda    -   Longitude (rad)
%%%                h    -   Altitude
%%%
%%% Output: pos_ecef    -   Position in ECEF coordinates
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pre-assign sin/cos functions
```

```
r  = h + 6378.1363;
cp = cos(phi_gd);
sp = sin(phi_gd);
cl = cos(lambda);
sl = sin(lambda);

% Compute ECEF position
ri = r * cp * cl;
rj = r * cp * sl;
rk = r * sp;

pos_ecef = [ri; rj; rk];

end

-----------------------------------------------------------

function pos_eci = ECEF2ECI(pos_ecef, theta_gst)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Rotate Earth-centered Earth-fixed to inertial coordinates
%%%
%%% Input:  pos_ecef    -   Position in ECEF coordinates
%%%        theta_gst    -   Greenwich Sidereal Time (rad)
%%%
%%% Output:  pos_eci    -   Position in ECI coordinates
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pre-assign sin/cos functions
st = sin(-theta_gst);
ct = cos(-theta_gst);

% Rotation about z-axis
R3 = [ ct st 0;
      -st ct 0;
       0  0  1];

% Rotate ECI vector
pos_eci = R3 * pos_ecef;

end

-----------------------------------------------------------

function pos_topo = ECEF2TOPO(pos_ecef, phi, lambda, h)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Compute Azimuth, Elevation, and Range from ECEF positions of tracking
%%% station and satellite.
%%%
%%% Input:  pos_ecef    -   Position in ECEF of the satellite
%%%              phi    -   Geodetic latitude (rad) of tracking station
%%%           lambda    -   Longitude (rad) of tracking station
%%%                h    -   Altitude of tracking station
```

```
%%%
%%% Output: pos_topo    -   [Azimuth; Elevation; Range] vector
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Find difference between tracking station and satellite in ECEF
r = pos_ecef - LATLON2ECEF(phi, lambda, h);

% Decompose vector for convenience
rx = r(1);
ry = r(2);
rz = r(3);

% Calculate azimuth, range, elevation relative to tracking station
azimuth     = atan2(ry, rx);
elevation   = atan2(rz, sqrt(rx^2 + ry^2));
range       = norm(r);

% Concatenate into single output vector
pos_topo = [azimuth; elevation; range];
end
```