# ASEN 5050 – Homework #4

Edward Meletyan

***[Matlab code attached at the end]***

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | | (1) | |
|  | |  | | (2) | |
|  | |  | | (3) | |
|  | |  | | (4) | |
|  | |  | | (5) | |
|  | |  | | (6) | |
|  | |  | |  | |
| **Transfer Option** |  | |  | |  |
| **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.

1. Convert ECI position to an ECEF position then to geocentric latitude, longitude, and latitude using the following procedure:
   1. Rotate the ECI position vector to ECEF coordinate frame using ECI2ECEF.m and input. This function yields the ECEF position vector:
   2. Use the ECEF2LATLON.m function to convert the ECEF position vector to geodetic latitude, longitude, and altitude.
   3. Convert the geodetic latitude to geocentric latitude using:
   4. Yields the following results:
2. Convert the given geodetic latitude, longitude, and elevation to an ECI position vector using the following procedure:
   1. 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:
   2. Now use ECEF2ECI.m function with and 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:
3. Compute the azimuth, elevation, and range using the following procedure:
   1. 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:
   2. 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:
   3. The azimuth, elevation, and range of the satellite relative to Boulder can be found using the following equations ( are the x, y, and z components of the vector between the tracking station and the satellite):

%% 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);

%% 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