A picture containing fireworks, dark, water, flying

Description automatically generated

College of Engineering

School of Aeronautics and Astronautics

AAE 532

Orbital Mechanics

PS 5

Time Elapse of Orbits and 3D Orbits

*Author:*

Tomoki Koike

*Supervisor:*

K. C. Howell

October 9th, 2020 Friday

Purdue University

West Lafayette, Indiana

**Problem 1**: Return to the use of multiple propagators in GMAT. (Demonstrated in a previous GMAT Tip.) The propagator ‘TwoBody’ or ‘EarthPointMass’ is already available (under the name you have selected for previous assignments). Produce the new propagators: ‘EarthMoon’, ‘EarthSun’, ‘EarthMoonSun’. Add the different forces to create the new propagators as described in the GMAT Tips document. Add the Moon’s orbit (Luna) to the output image. Propagate for 60 days.

1. Use a date October 2, 2020 16:00:00. Use the Earth J2000Eq coordinates throughout the simulations. In a Keplerian Coordinate Type, introduce initial conditions such that

Explore the 4 propagators (use different color for each propagated path). Propagate all the trajectories for 60 days. [Sometimes it is convenient to use the ‘Animation’ button on the top bar if you have not already tried it! Watch each simulation evolve.]

Produce a plot with a view approximately down the Moon Orbit Normal with all four spacecraft. Add views on two other dates: October 7, 2020 and October 11, 2020 at the same time of day. Choose another date in October and add a figure.

These simulations all use the relative vector equation of motion for the spacecraft relative to the Earth from Notes Page D2; the perturbations on the right-hand side of the equation vary for each propagator.

Does the model make a difference? Is the two-body model adequate for this particular problem? Why or why not? For the trajectory in this analysis, which relative orbit model would you recommend: two-body, three-body, four-body? Why? Which bodies would you include?

What is the impact of the different epoch dates? Why is there such a difference in the paths?

From the given values of distance of periapsis and apoapsis we can calculate the semi-major axis and eccentricity

Since,

Then,

Set this in the settings for “Sat1” in GMAT. Duplicate this for 3 more satellites.

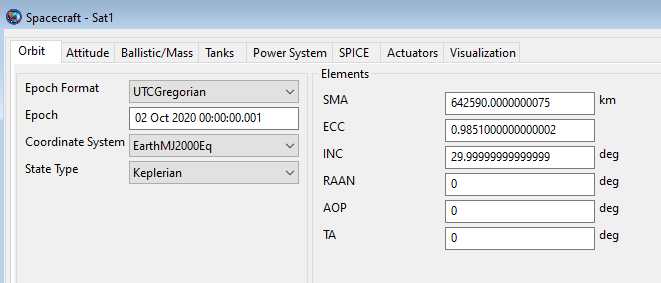


Figure : Sat1 settings

Define 4 missions, “Propagate1”, “Propagate2”, “Propagate3”, and “Propagate4”. Each propagator corresponds to the propagator “TwoBody”, “EarthMoon”, “EarthSun”, and “EarthMoonSun” respectively. For the “Stopping conditions” use the “ElapsedDays” and set that to 60 days.

On the next page you will see the plots for each satellite tracking each propagation defined. Each plot indicates the location of the satellite for the dates October 2, 2020 00:00:00:001, October 7, 2020 00:00:00:001, and October 11, 2020 00:00:00:001 respectively. Additionally, the date October 30, 2020 00:00:00:001 was selected, and the plot is shown as the fourth one.

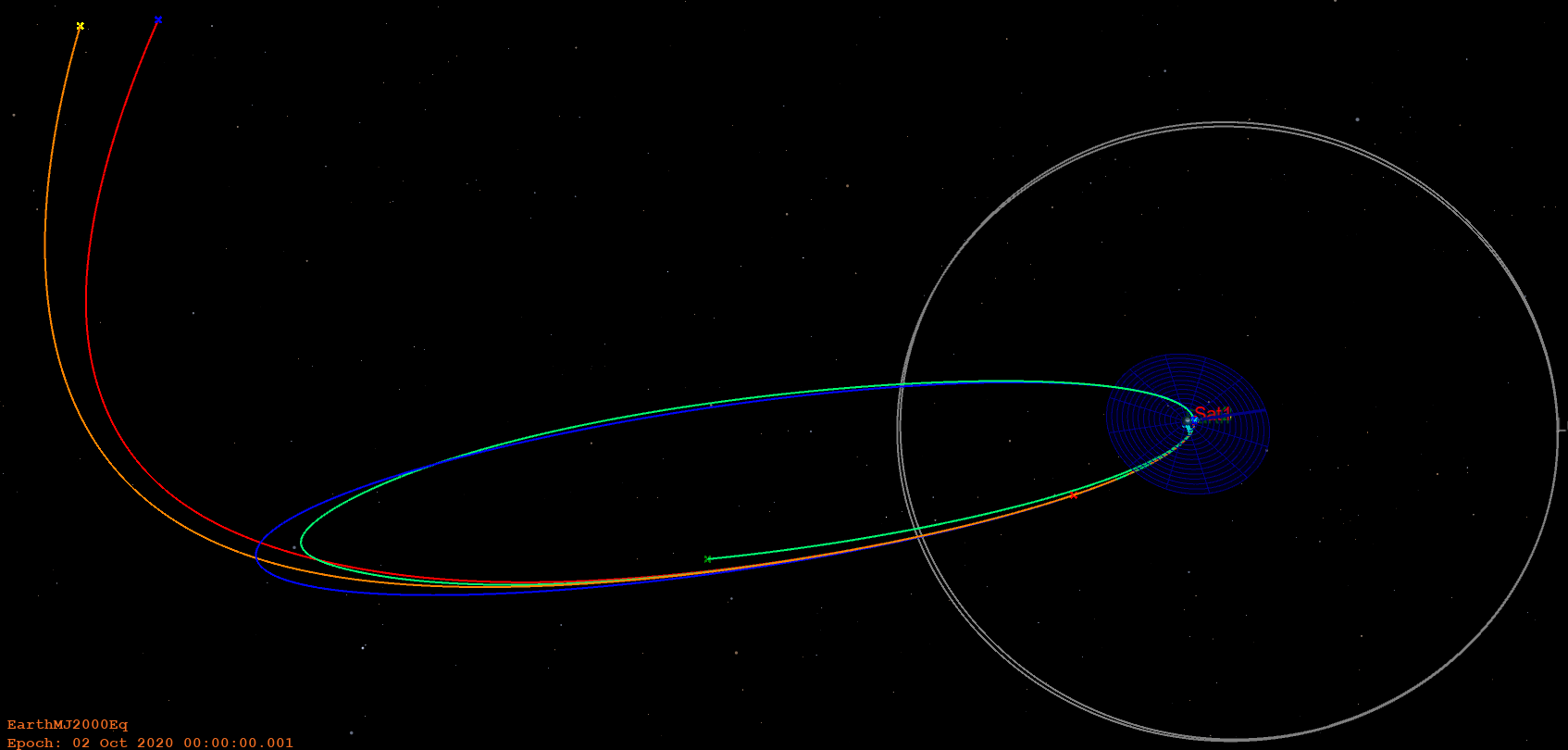


Figure : October 2 orbit

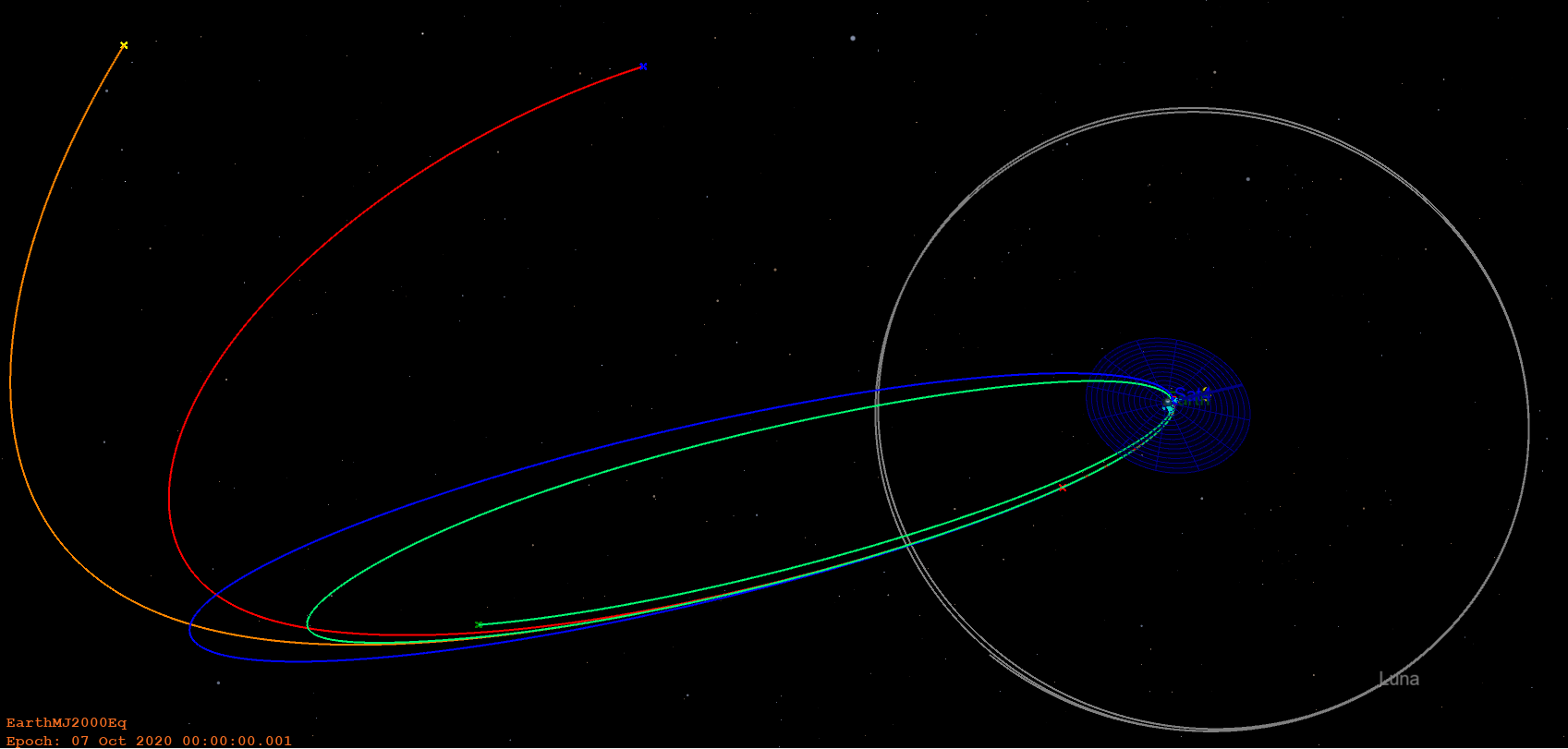


Figure : October 7 orbit

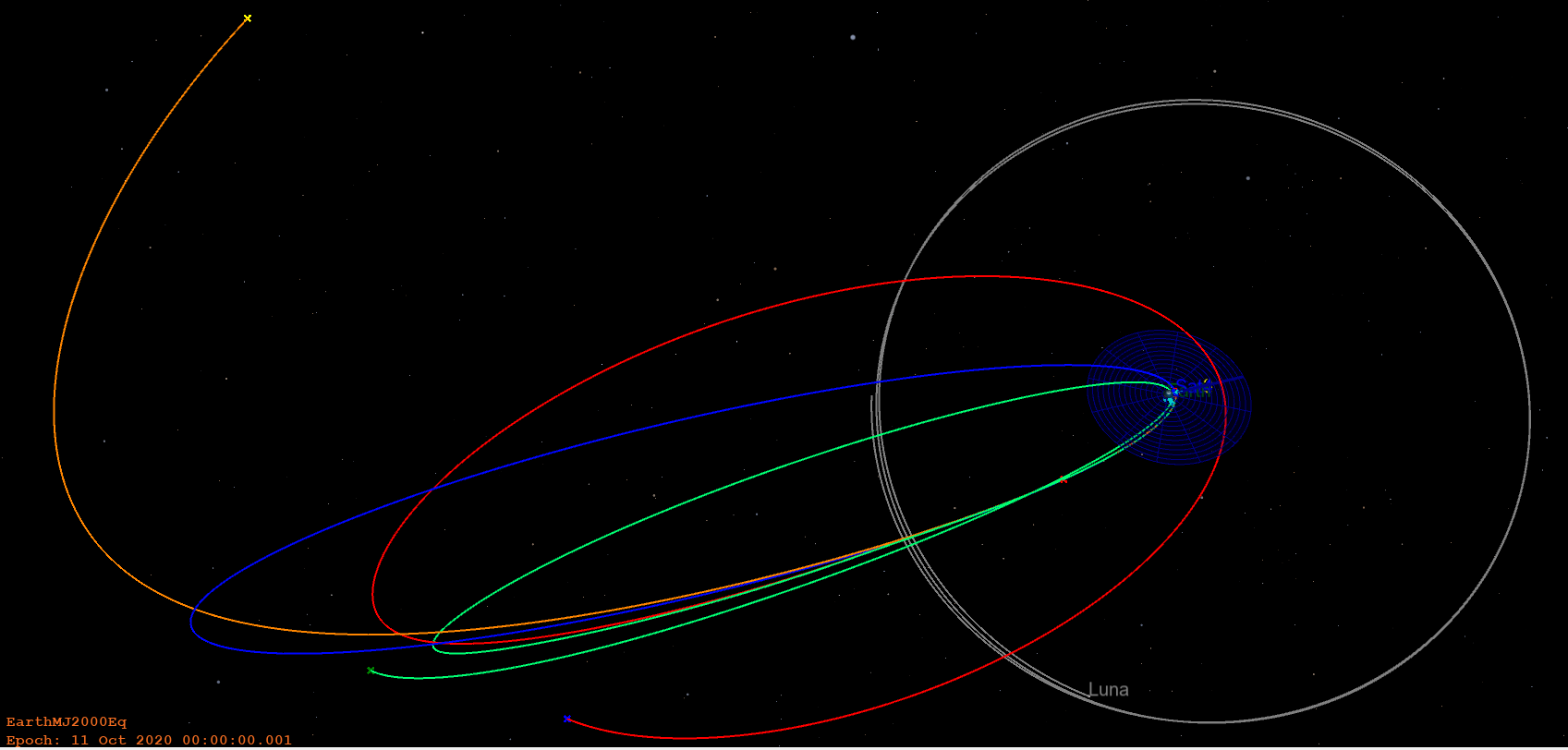


Figure : October 11 orbit

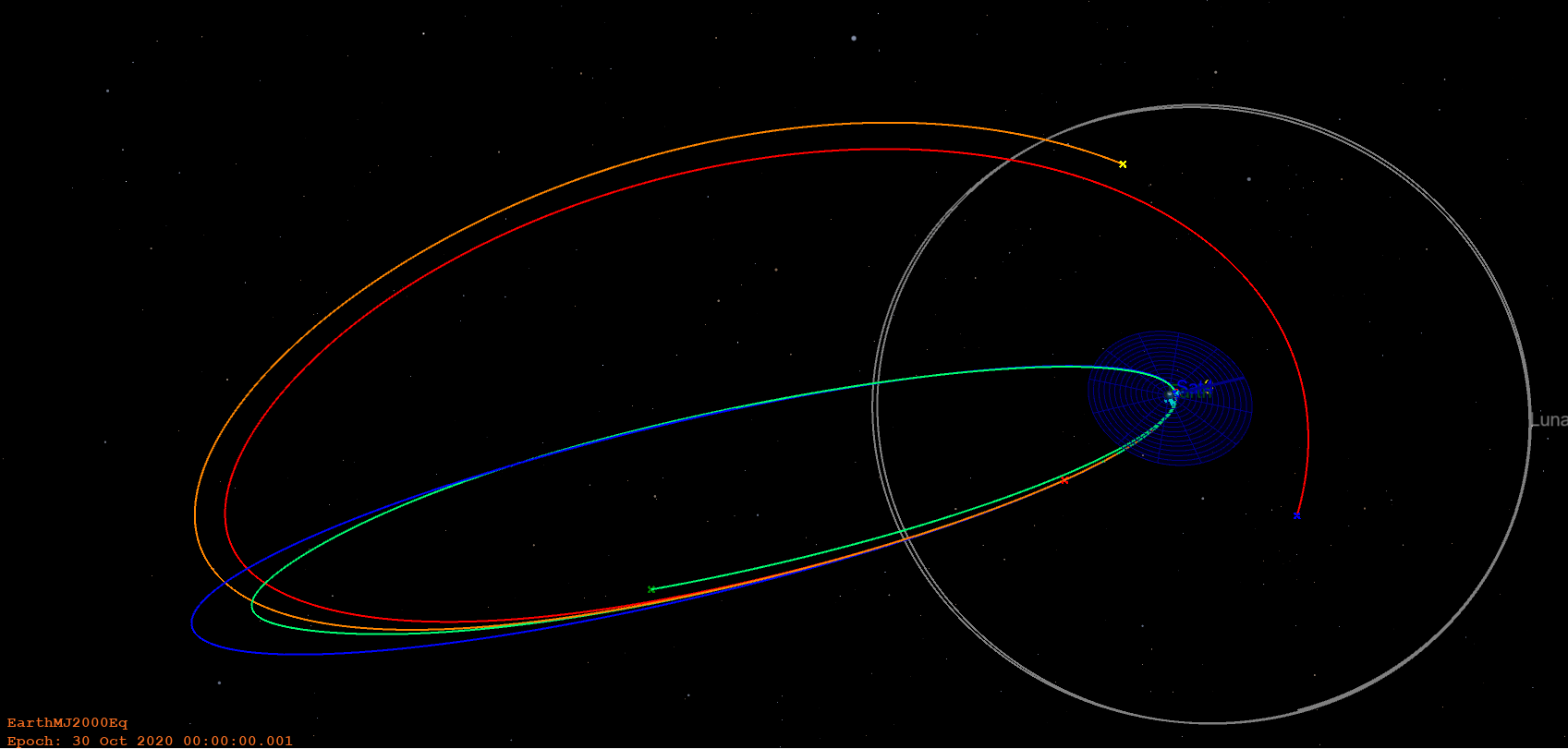


Figure : October 30 orbit

The model does make a difference. The point mass model of the Earth and the three-body problem of the Earth-Moon-spacecraft is almost alike when the Moon is relatively farther away from the spacecraft’s orbit throughout the simulation time; however, when it is close to the spacecraft (figures 3 and 4) the orbits diverge largely. Also, when the Sun is incorporated into the model the orbit largely diverges from the elliptical orbit that the orbital models without the Sun travels. Thus, the two-body model is not adequate for this problem. Although, the three-body is not enough as well from what we have seen when the Moon is close to the spacecraft. That is the orbital behavior changes significantly when we disregard the moon while the Moon is close to the orbit of the spacecraft. Thus, we conclude that we must incorporate both the Moon and Sun in our model which makes the model a four-body model of Earth-Moon-Sun-spacecraft.

From the starting epoch difference the initial relative distance of the Moon to the spacecraft changes and the behavior of the orbit changes significantly on forward of the simulation. We can observe this from the figures 3 and 4. Just with a 4-day difference in the starting point of the simulation causes the “EarthSun” and “EarthMoonSun” propagation to have a large discrepancy.

1. Output some information for each spacecraft at , the end of the propagation. Determine the following information from the GMAT output: . Compare the closest approach altitude for all the spacecraft at the end of the simulation. Are any spacecraft in danger of Earth impact? Which perturbation reduced the ? Does it occur at all starting epochs? (Note that, if the model is not a true conic – as is the case for three of the four propagators – GMAT computes instantaneous values of these quantities. Hint: check the output at the end of the final propagate segment.)

The data from the reports of GMAT is processed using MATLAB and the following tables and graphs are created. are all final values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| October 2, 2020 00:00:00:001 | Earth Point Mass | Earth Moon | Earth Sun | Earth Moon Sun |
|  | 6.4259E+05 | 6.2375E+05 | 1.1981E+06 | 1.0653E+06 |
|  | 0.9851 | 0.9841 | 0.3928 | 0.4497 |
|  | 9.5746E+03 | 9.9413E+03 | 7.2752E+05 | 5.8628E+05 |
|  | -0.3102 | -0.3195 | -0.1663 | -0.1871 |
|  | 8.7040E+04 | 8.8668E+04 | 6.3553E+05 | 5.8205E+05 |
|  | 1.6695E+05 | 6.4202E+05 | 1.6609E+06 | 1.5433E+06 |
|  | 2.0383 | 0.7763 | 0.3838 | 0.3774 |
|  | 154.1002 | 170.0558 | 173.1045 | 182.4081 |
|  | 75.1805 | 79.7525 | 4.4200 | -1.9649 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| October 7, 2020 00:00:00:001 | Earth Point Mass | Earth Moon | Earth Sun | Earth Moon Sun |
|  | 6.4259E+05 | 5.7788E+05 | 1.0950E+06 | 7.5403E+05 |
|  | 0.9851 | 0.9830 | 0.3814 | 0.6218 |
|  | 9.5746E+03 | 9.8391E+03 | 6.7738E+05 | 2.8519E+05 |
|  | -0.3102 | -0.3449 | -0.1820 | -0.2643 |
|  | 8.7040E+04 | 8.8187E+04 | 6.1072E+05 | 4.2937E+05 |
|  | 1.6695E+05 | 9.1054E+05 | 1.4911E+06 | 8.0809E+05 |
|  | 2.0383 | 0.4310 | 0.4130 | 0.6767 |
|  | 154.1002 | 174.5760 | 192.3951 | 226.5451 |
|  | 75.1805 | 77.0140 | -7.4325 | -38.2599 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| October 11, 2020 00:00:00:001 | Earth Point Mass | Earth Moon | Earth Sun | Earth Moon Sun |
|  | 6.4259E+05 | 5.4642E+05 | 1.0218E+06 | 4.8835E+05 |
|  | 0.9851 | 0.9832 | 0.4131 | 0.8699 |
|  | 9.5746E+03 | 9.1542E+03 | 5.9965E+05 | 6.3535E+04 |
|  | -0.3102 | -0.3647 | -0.1950 | -0.4081 |
|  | 8.7040E+04 | 8.5068E+04 | 5.8118E+05 | 2.1761E+05 |
|  | 1.6695E+05 | 1.0734E+06 | 1.3361E+06 | 8.5524E+05 |
|  | 2.0383 | 0.1148 | 0.4545 | 0.3405 |
|  | 154.1002 | 178.9663 | 207.7009 | 171.8384 |
|  | 75.1805 | 46.3637 | -16.8473 | 41.6375 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| October 30, 2020 00:00:00:001 | Earth Point Mass | Earth Moon | Earth Sun | Earth Moon Sun |
|  | 6.4259E+05 | 6.1896E+05 | 7.7179E+05 | 7.0913E+05 |
|  | 0.9851 | 0.9839 | 0.7440 | 0.7720 |
|  | 9.5746E+03 | 9.9520E+03 | 1.9759E+05 | 1.6168E+05 |
|  | -0.3102 | -0.3220 | -0.2582 | -0.2810 |
|  | 8.7040E+04 | 8.8713E+04 | 3.7061E+05 | 3.3793E+05 |
|  | 1.6695E+05 | 6.9249E+05 | 3.0171E+05 | 2.0831E+05 |
|  | 2.0383 | 0.7122 | 1.4580 | 1.8069 |
|  | 154.1002 | 170.8820 | 281.0148 | 60.9116 |
|  | 75.1805 | 79.6375 | -32.5945 | 26.1293 |

The raw data from the report is in the appendix.

Also, from GMAT the altitude data is retrieved and processed in MATLAB. The plot is on the following page.

Chart

Description automatically generated

Chart, line chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generated

From the plots we can tell that the “EarthPointMass” and “EarthMoon” model spacecraft has a moment where the altitude approaches very close to 0. Thus, these 2 models are in danger of colliding into Earth.

From the tables, we can see that the model for “EarthMoon” with an epoch of “11 Oct 2020” has a reduced radius of periapsis. Mostly, the “EarthMoon” model has a radius of periapsis closest to the point mass model.

From the plots we can see that the starting epoch for all models have a local minimum of the altitude. However, still the altitudes are not 0 and 3000~ km. For the Earth point mass and Earth-Moon model there are points that have lower than the starting epoch such as “EarthMoon” with an epoch of “11 Oct 2020”. Thus, not all models have the point of lowest altitude at the starting epoch.

**Problem 2**: A spacecraft is in orbit about Mars and is characterized such that  **and** . The vehicle is currently located such that .

1. Determine the following orbit parameters and spacecraft state information.

From the radius of periapsis and apoapsis

Since, , the semi-major axis becomes

Next, from the periapsis and apoapsis we can calculate the eccentricity

Then, the semi-latus rectum, becomes

When the gravitational parameter, , the specific angular momentum becomes

The period of this orbit is

The constant specific energy of this orbit is

At the current location where the mean anomaly is we can find the eccentric anomaly. A MATLAB function solving the transcendental equation with the Bessel function was used to solve this. The code is shown below.

function E = M2E\_anomaly(M, e, unit, tol)

%{

NAME: M2E\_anomaly

AUTHOR: TOMOKI KOIKE

INPUTS: (1) M: MEAN ANOMALY

(2) e: ECCENTRICITY

(3) unit: GRAVITATIONAL PARAMETER

(4) tol: TOLERANCE

OUTPUTS: (1) E: ECCENTRIC ANOMALY

DESCRIPTION: CALCULATES THE ECCENTRIC ANOMALY FROM THE MEAN ANOMALY

USING THE BESSEL FUNCTIONS EXPLICIT APPROACH.

%}

%Checking for user inputed tolerance

if nargin == 3

%using default value

tol = 10^-8;

elseif nargin > 4

error('Too many inputs.')

elseif nargin < 3

error('Too few inputs.')

end

% Check units

if unit == "deg"

M = deg2rad(M);

end

% Check tolerance

if tol > 0.01

error('Set a tolerance smaller than 0.001')

end

% Bessel function method

del1 = 1; del2 = 1; del3 = 1;

N = 1; ct = 1;

Estore = [];

while ~(del1 < tol && del2 < tol && del3 < tol && ct > 10)

f = 0;

for m = 1:N

f = f + (1 / m) \* besselj(m,m\*e) \* sin(m\*M);

end

Estore = [Estore, M + 2\*f];

if ct > 4

del1 = abs(Estore(ct) - Estore(ct-1));

del2 = abs(Estore(ct-1) - Estore(ct-2));

del3 = abs(Estore(ct-2) - Estore(ct-3));

end

N = N + 1; ct = ct + 1;

end

E = Estore(end);

end

This algorithm computes the eccentric anomaly to be

Then from the eccentric anomaly we can calculate the true anomaly. The following MATLAB function is used to do this computation

function theta\_star = E2T\_anomaly(e, E, unit)

%{

NAME: E2T\_anomaly

AUTHOR: TOMOKI KOIKE

INPUTS: (1) e: ECCENTRICITY

(2) E: ECCENTRIC ANOMALY

(3) unit: DEGREES OR RADIANS

OUTPUTS: (1) theta\_star: TRUE ANOMALY

DESCRIPTION: CALCULATES THE TRUE ANOMALY FROM THE ECCENTRIC ANOMALY.

%}

ee = sqrt((1 + e) / (1 - e));

if unit == "deg"

theta\_star = 2\*atand(ee \* tand(E / 2));

if theta\_star < 0

theta\_star = 360 + theta\_star;

end

else

theta\_star = 2\*atan(ee \* tan(E/ 2));

if theta\_star < 0

theta\_star = 2\*pi + theta\_star;

end

end

end

This gives the following result,

Then,

The position gives us the velocity.

Then, since the orbit is descending the flight path angle is

Finally, from the mean anomaly, we can compute (M in radians and positive)

1. Write and in terms of components in the directions and .

From the true anomaly we can represent the position and velocity vectors as

1. Determine after a time equal to 50% of the period, i.e., . Use and relationships to write in terms of . Prove that produce the same results as .

When the time is equal to 50% of the period,

and is the time when and in part (a).

Using the formula on notes **G7,** we can compute the eccentric anomaly for when half a period has passed from the this point where .

The subscript of “f” is the notation for the final point we are looking for. Now, subtract the first equation from the second one and we obtain

Then,

Since we know that in part (a)

We solve this with the same algorithm we used in part (a) to compute the eccentric anomaly from the mean anomaly, and we get

Now, compute the true anomaly from the eccentric anomaly

To find the final position and velocity vectors for an ellipse we use the following formula from notes H2

Here is a MATLAB function that computes the coefficients and the final position and velocity vectors

function [rvec, vvec, f, g, fdot, gdot] = FandG\_elp(a, mu, dE, dt, r0vec, v0vec)

%{

NAME: rv\_vecTconv\_elp

AUTHOR: TOMOKI KOIKE

INPUTS: (1) a: SEMI MAJOR AXIS

(2) mu: GRAVITATIONAL PARAMETER

(3) dE: DIFFERENCE OF ECCENTRIC ANOMALY

(4) dt: DIFFERENCE OF TIME

(5) r0vec: R-VECTOR AT INITIAL POINT

(6) v0vec: V-VECTOR AT INITIAL POINT

OUTPUTS: (1) rvec: R-VECTOR AT FINAL POINT

(2) vvec: V-VECTOR AT FINAL POINT

(3) f: THE F COEFFICIENT

(4) g: THE G COEFFICIENT

(5) fdot: THE FDOT COEFFICIENT

(6) gdot: THE GDOT COEFFICIENT

DESCRIPTION: CALCULATES THE F AND G FOR POSITION VECTOR AND VELOCITY

VECTOR AS WELL AS THE FINAL VECTORS FOR AN ELLIPSE.

%}

if (dE > 3.14 || -3.14 > dE) || dE > 6.28

error("Enter the units in radius.")

elseif ~(numel(r0vec) >= 2 && numel(v0vec) >= 2) || (numel(r0vec) ~= numel(v0vec))

error("The initial position and velocity vectors should have equal dimensions larger than 2.")

end

% Compute F

n = sqrt(mu / a^3); % mean motion

r0 = norm(r0vec); % magnitude of initial position vector

f = 1 - (a/r0) \* (1 - cos(dE));

% Compute G

g = dt + (sin(dE) - dE) / n;

%Compute rvec

rvec = f \* r0vec + g \* v0vec;

% Compute Fdot

r = norm(rvec); % magnitude of final position vector

fdot = -(n \* a^2 / r / r0) \* sin(dE);

% Compute Gdot

gdot = 1 - (a / r) \* (1 - cos(dE));

% Compute vvec

vvec = fdot \* r0vec + gdot \* v0vec;

end

Using this, we acquire the following results,

Thus,

Now, if we use the true anomaly, we use the following formula on notes H4

Use this MATLAB function,

function [rvec, vvec, f, g, fdot, gdot] = FandG\_conic(p, mu, dTA, r0vec, v0vec, rfmag)

%{

NAME: rv\_vecTconv\_conic

AUTHOR: TOMOKI KOIKE

INPUTS: (1) p: SEMI LATUS RECTUM

(2) mu: GRAVITATIONAL PARAMETER

(3) dTA: DIFFERENCE OF TRUE ANOMALY

(4) r0vec: R-VECTOR AT INITIAL POINT

(5) v0vec: V-VECTOR AT INITIAL POINT

(6) rfmag: THE RADIAL DISTANCE AT THE FINAL POINT

OUTPUTS: (1) rvec: R-VECTOR AT FINAL POINT

(2) vvec: V-VECTOR AT FINAL POINT

(3) f: THE F COEFFICIENT

(4) g: THE G COEFFICIENT

(5) fdot: THE FDOT COEFFICIENT

(6) gdot: THE GDOT COEFFICIENT

DESCRIPTION: CALCULATES THE F AND G FOR POSITION VECTOR AND VELOCITY

VECTOR AS WELL AS THE FINAL VECTORS FOR AN ELLIPSE.

%}

d = abs(dTA / 2 / pi);

if d > 6.28

error("Enter the units in radius.")

elseif ~(numel(r0vec) >= 2 && numel(v0vec) >= 2) || (numel(r0vec) ~= numel(v0vec))

error("The initial position and velocity vectors should have equal dimensions larger than 2.")

end

% Compute F

r0 = norm(r0vec); % magnitude of initial position vector

f = 1 - (rfmag/p) \* (1 - cos(dTA));

% Compute G

g = (rfmag \* r0 / sqrt(mu \* p)) \* sin(dTA);

%Compute rvec

rvec = f \* r0vec + g \* v0vec;

% Compute Fdot

fdot = (dot(r0vec, v0vec) / p / r0) \* (1 - cos(dTA)) - (1/r0)\*sqrt(mu/p)\*sin(dTA);

% Compute Gdot

gdot = 1 - (r0 / p) \* (1 - cos(dTA));

% Compute vvec

vvec = fdot \* r0vec + gdot \* v0vec;

end

Computing the same values using this function we get the exact same values

Thus,

Hence, it is proven that the f, g function for both the elliptical and conical version agree.

1. Plot the orbit with your MATLAB script. By hand, mark on the plot where the spacecraft is currently located by marking ; also sketch the local horizon, , and . Do the same at the second location. Identify the arc from to .

Chart

Description automatically generated

The arc outlined in pink is the arc from to . (The MATLAB code is in the appendix.)

**Problem 3**: Assume that a vehicle is currently moving in Earth orbit. Assume a two-body relative motion of the spacecraft with respect to Earth. The orbit is described with the following characteristics (relative to an Earth equatorial coordinate system) at the initial time :

1. Determine the current state in terms of , ; write in terms of both rotating orbit unit vectors (), unit vectors () as well as inertial unit vectors ().

3-1-3 (body-two) Euler Sequence:

From and , we know

And, the semi latus rectum is

Then, we can find the magnitude of

Since, , the magnitude of the velocity vector is

The specific angular momentum is

The flight path angle is

The period of this orbit is

The eccentric anomaly can be calculated using,

The mean anomaly becomes

The time elapse from periapsis is

Using the radial distance we can express the r vector in the orbital frame

Then, the velocity vector becomes

To compute the frame transformation the following MATLAB function is used

function resvec = orbit\_frame\_transform(theta, i, Omega, vec, frame, unit)

%{

NAME: orbit\_frame\_transform

AUTHOR: TOMOKI KOIKE

INPUTS: (1) theta: ARGUMENT OF LATITUDE

(2) i: INCILNATION

(3) Omega: RIGHT ASCENSION OF ASCENDING NODE

(4) vec: VECTOR TO TRANSFORM

(5) frame: THE STARTING FRAME (ORBITAL OR INERTIAL)

(6) unit: DEGREE OR RADIANS

OUTPUTS: (1) resvec: RESULTING VECTOR STRUCTURE

DESCRIPTION: TRANSFORMS THE ORBITAL VECTOR (POSITION OR VELOCITY)

USING THE ORBITAL ANGLES.

%}

if unit == "radian"

theta = rad2deg(theta);

i = rad2deg(i);

Omega = rad2deg(Omega);

end

% Direction cosine matrices

% r,theta,h --> qx,qy,qz

C\_oq = [cosd(theta), sind(theta), 0; -sind(theta), cosd(theta), 0; 0,0,1];

% qx,qy,qz --> nx,ny,nz

C\_qn = [1,0,0;0,cosd(i),sind(i);0,-sind(i),cosd(i)];

% nx,ny,nz --> x,y,z

C\_ni = [cosd(Omega),sind(Omega),0;-sind(Omega),cosd(Omega),0;0,0,1];

% For reverse

C\_in = C\_ni'; C\_nq = C\_qn'; C\_qo = C\_oq';

% Transform

if frame == "orbital"

% Orbital to q-frame

resvec.q = vec \* C\_oq;

% q-frame to n-frame

resvec.n = resvec.q \* C\_qn;

% n-frame to inertial

resvec.i = resvec.n \* C\_ni;

elseif frame == "inertial"

% Inertial to n-frame

resvec.n = vec \* C\_in;

% n-frame to q-frame

resvec.q = resvec.n \* C\_nq;

% q-frame to orbital frame

resvec.o = resvec.q \* C\_qo;

else

error("Enter a frame of either 'orbital' or 'inertial'.");

end

end

From (1) and (2),

Hence, to transform the vectors from the orbital frame (“o”) to the n-frame (“n”) is

Thus, in the n-frame the position vector and velocity vectors become

Then to transform it to the inertial frame (“i”), from (3)

Compute this using MATLAB and we get,

1. Confirm the general results in GMAT with the conic propagator. Plot the GMAT image viewing down onto the orbit plane.

The GMAT results give us ,

Propagate Command: Propagate1

Spacecraft : Sat1

Coordinate System: EarthMJ2000Eq

Time System Gregorian Modified Julian

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

UTC Epoch: 13 Oct 2020 17:56:23.819 29136.2474979045

TAI Epoch: 13 Oct 2020 17:57:00.819 29136.2479261452

TT Epoch: 13 Oct 2020 17:57:33.003 29136.2482986452

TDB Epoch: 13 Oct 2020 17:57:33.001 29136.2482986262

Cartesian State Keplerian State

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

X = 13353.666846499 km SMA = 127562.72599965 km

Y = -158511.40493158 km ECC = 0.5999999999984

Z = -81970.967996886 km INC = 34.000000000000 deg

VX = 0.8810382912704 km/sec RAAN = 45.000000000000 deg

VY = 0.7412445371591 km/sec AOP = 29.999999999962 deg

VZ = -0.0666745675888 km/sec TA = 204.99999999877 deg

MA = 253.30062046024 deg

EA = 227.82397421787 deg

Spherical State Other Orbit Data

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

RMAG = 178950.90250115 km Mean Motion = 1.385744618e-05 deg/sec

RA = -85.184533212648 deg Orbit Energy = -1.5623703490827 km^2/s^2

DEC = -27.262251997392 deg C3 = -3.1247406981653 km^2/s^2

VMAG = 1.1533071717959 km/s Semilatus Rectum = 81640.144640011 km

AZI = 111.15056655935 deg Angular Momentum = 180393.45247994 km^2/s

VFPA = 119.06592694950 deg Beta Angle = -21.012833536281 deg

RAV = 40.074930888723 deg Periapsis Altitude = 44646.954100056 km

DECV = -3.3142102788421 deg VelPeriapsis = 3.5353872196127 km/s

VelApoapsis = 0.8838468049075 km/s

Orbit Period = 453415.81894988 s

Planetodetic Properties

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

LST = 275.14019961493 deg

MHA = 291.85203753226 deg

Latitude = -27.258513764093 deg

Longitude = -16.711837917326 deg

Altitude = 172577.24511534 km

Spacecraft Properties

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

Cd = 2.200000

Drag area = 15.00000 m^2

Cr = 1.800000

Reflective (SRP) area = 1.000000 m^2

Dry mass = 850.00000000000 kg

Total mass = 850.00000000000 kg

SPADDragScaleFactor = 1.000000

SPADSRPScaleFactor = 1.000000

========================================================================

The values in red verifies our answers. (The FPA is (90– VFPA) on the report above.)

A close up of a light

Description automatically generated

1. Use Kepler’s equation and determine the values of in exactly 3 days, i.e., time . For this value of , what are the corresponding values of , . Confirm the result in GMAT.

3 days is a time elapse of 259,200 seconds.

The mean anomaly becomes

Using the MATLAB function in problem 2 (on page 13-14) we can find the eccentric anomaly at this second point to be

The true anomaly at this time is

Thus, knowing that 3 days is less than one full orbit

Using the f and g function we can compute the position and velocity vectors after this time.

Using the MATLAB function we used in problem 2 (on page 18) we get the following results

Finally the flight path angle is

Propagate Command: Propagate1

Spacecraft : Sat1

Coordinate System: EarthMJ2000Eq

Time System Gregorian Modified Julian

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

UTC Epoch: 11 Oct 2020 11:59:28.000 29133.9996296296

TAI Epoch: 11 Oct 2020 12:00:05.000 29134.0000578704

TT Epoch: 11 Oct 2020 12:00:37.184 29134.0004303704

TDB Epoch: 11 Oct 2020 12:00:37.182 29134.0004303513

Cartesian State Keplerian State

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

X = -119251.69147723 km SMA = 127562.72599965 km

Y = -125671.20223367 km ECC = 0.5999999999984

Z = -3061.7827422874 km INC = 34.000000000000 deg

VX = 0.2022456483717 km/sec RAAN = 45.000000000000 deg

VY = -1.0409621417831 km/sec AOP = 29.999999999962 deg

VZ = -0.5929473913807 km/sec TA = 151.81082438204 deg

MA = 99.098468928579 deg

EA = 126.67164814234 deg

Spherical State Other Orbit Data

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

RMAG = 173273.17018107 km Mean Motion = 1.385744618e-05 deg/sec

RA = -133.49860225159 deg Orbit Energy = -1.5623703490826 km^2/s^2

DEC = -1.0124842709371 deg C3 = -3.1247406981653 km^2/s^2

VMAG = 1.2149453040594 km/s Semilatus Rectum = 81640.144640011 km

AZI = 123.98673315762 deg Angular Momentum = 180393.45247994 km^2/s

VFPA = 58.970842503039 deg Beta Angle = -21.401642701711 deg

RAV = -79.005139108244 deg Periapsis Altitude = 44646.954100056 km

DECV = -29.212134042839 deg VelPeriapsis = 3.5353872196127 km/s

VelApoapsis = 0.8838468049075 km/s

Orbit Period = 453415.81894988 s

Planetodetic Properties

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

LST = 226.76913615495 deg

MHA = 200.40385314776 deg

Latitude = -1.0911644074554 deg

Longitude = 26.365283007198 deg

Altitude = 166895.04162097 km

Spacecraft Properties

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

Cd = 2.200000

Drag area = 15.00000 m^2

Cr = 1.800000

Reflective (SRP) area = 1.000000 m^2

Dry mass = 850.00000000000 kg

Total mass = 850.00000000000 kg

SPADDragScaleFactor = 1.000000

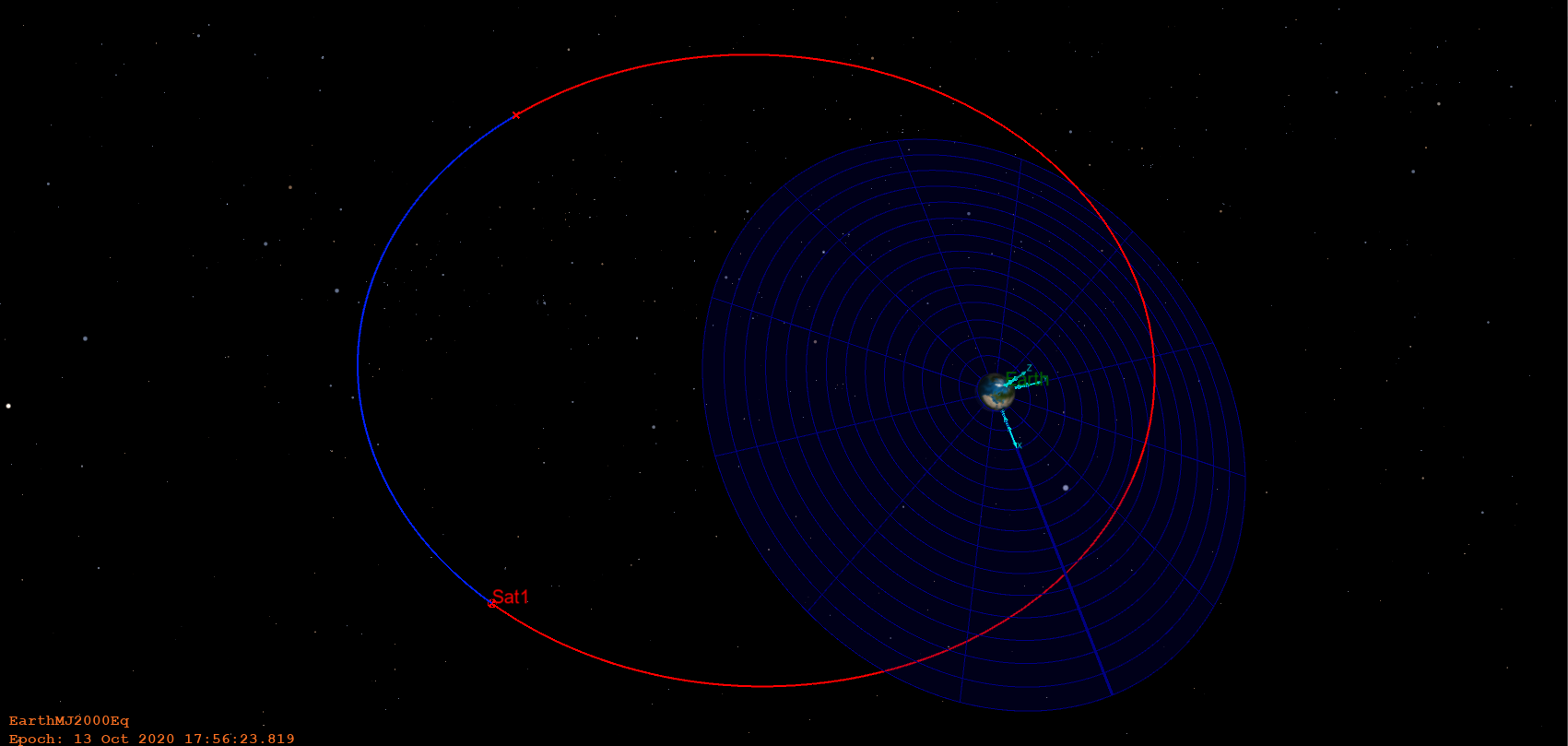
SPADSRPScaleFactor = 1.000000

========================================================================

The values in red match with our values. Thus, our results are verified. (The FPA is (VFPA -90) on the report above.)

1. Plot the orbit in MATLAB or GMAT. Mark at the two times; mark the usual quantities (vectors, local horizon, , ) and highlight the arc between the two times.

The plot from GMAT is the following.



**Problem 4**: A vehicle is moving in some Earth orbit; assume a two-body model. At a certain time, the following information is given

1. Determine . Are you sure it is an ellipse? Why? What quantity do you check to assess the type of conic?

The magnitude of the position and the velocity are

We know that

Then,

and

Also, the unit r-vector is

The other unit vector can be found from the two unit vectors with the cross product

We check if this is an elliptical orbit

Thus, this is an elliptical orbit.

Now, the semi major axis is

The eccentricity becomes

The radius of periapsis and apoapsis are

The semi latus rectum is

Since these three parameters are larger than the radius of the Earth, we can say that there is no risk of collision.

Now using the rotational transformation of frames

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Plugging in the values that we have, we know that

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Thus, the inclination can be found initially

Then, from

Thus, the common angle value is

Similarly

The common angle of the two is

To figure out if this point is in ascending or descending, we do the following

It is smaller than 0, thus the position is descending in orbit.

The true anomaly becomes

Since the orbit is descending,

Then the argument of periapsis becomes

The flight path angle is

The eccentric anomaly can be found using the following formula (solved by the MATLAB function below)

function E = T2E\_anomaly(e, theta\_star, unit)

%{

NAME: T2E\_anomaly

AUTHOR: TOMOKI KOIKE

INPUTS: (1) e: ECCENTRICITY

(2) theta\_star: TRUE ANOMALY

(3) unit: DEGREES OR RADIANS

OUTPUTS: (1) E: ECCENTRIC ANOMALY

DESCRIPTION: CALCULATES THE ECCENTRIC ANOMALY FROM THE TRUE ANOMALY.

%}

ee = sqrt((1 - e) / (1 + e));

if unit == "deg"

E = 2\*atand(ee \* tand(theta\_star / 2));

else

E = 2\*atan(ee \* tan(theta\_star / 2));

end

end

Then the mean anomaly becomes

And,

As discussed in the calculation process, we have concluded that the orbit was an ellipse by proving that the velocity and the point was smaller than the circular velocity multiplied by the square root of two. Also, assessing the eccentricity tells that the conic type of the orbit is an ellipse.

1. Sketch the orbit in the orbit plane: add .



1. Sketch the orbit in 3D (or a section of the orbit) to mark the following quantities: , AN (Ascending Node), DN (Descending Node), direction of motion. Is periapsis above or below the fundamental plane? How do you know? What is at the AN? DN?

Direction of motion



AN

DN

The periapsis is below the fundamental plane. We can tell that from the signs of .

At AN , and at DN .

Appendix

MATLAB CODE

%% AAE 532 HW 5 Problem 1

% Tomoki Koike

close all; clear all; clc;

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps5';

set(groot, 'defaulttextinterpreter','latex');

set(groot, 'defaultAxesTickLabelInterpreter','latex');

set(groot, 'defaultLegendInterpreter','latex');

format longG;

% Load the GMAT data

warning('off','all');

reportDir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\gmat\ps5\reports';

for i = 0:3

for j = 1:4

if j == 1

T(i+1).earthpointmass = readtable( ...

fullfile(reportDir,"ps5\_p1\_"+num2str(i)+num2str(j)+".txt"));

elseif j == 2

T(i+1).earthmoon = readtable( ...

fullfile(reportDir,"ps5\_p1\_"+num2str(i)+num2str(j)+".txt"));

elseif j == 3

T(i+1).earthsun = readtable( ...

fullfile(reportDir,"ps5\_p1\_"+num2str(i)+num2str(j)+".txt"));

else

T(i+1).earthmoonsun = readtable( ...

fullfile(reportDir,"ps5\_p1\_"+num2str(i)+num2str(j)+".txt"));

end

end

end

warning("on");

% Process the data to filter out the unnecessary values

for i = 1:4

for j = 1:4

if j == 1

t = T(i).earthpointmass.Sat1\_Earth\_RMAG;

x1 = T(i).earthpointmass.Sat1\_Earth\_RMAG(1);

x2 = T(i).earthpointmass.Sat1\_Earth\_RMAG(end);

XX = find(t ~= x1 & t ~= x2);

elseif j == 2

t = T(i).earthmoon.Sat2\_Earth\_RMAG;

y1 = T(i).earthmoon.Sat2\_Earth\_RMAG(1);

y2 = T(i).earthmoon.Sat2\_Earth\_RMAG(end);

YY = find(t ~= y1 & t ~= y2);

elseif j == 3

t = T(i).earthsun.Sat3\_Earth\_RMAG;

z1 = T(i).earthsun.Sat3\_Earth\_RMAG(1);

z2 = T(i).earthsun.Sat3\_Earth\_RMAG(end);

ZZ = find(t ~= z1 & t ~= z2);

else

t = T(i).earthmoonsun.Sat4\_Earth\_RMAG;

w1 = T(i).earthmoonsun.Sat4\_Earth\_RMAG(1);

w2 = T(i).earthmoonsun.Sat4\_Earth\_RMAG(end);

WW = find(t ~= w1 & t ~= w2);

end

end

T(i).earthpointmass = T(i).earthpointmass([XX(1)-1, XX', XX(end)+1], :);

T(i).earthmoon = T(i).earthmoon([YY(1)-1, YY', YY(end)+1], :);

T(i).earthsun = T(i).earthsun([ZZ(1)-1, ZZ', ZZ(end)+1], :);

T(i).earthmoonsun = T(i).earthmoonsun([WW(1)-1, WW', WW(end)+1], :);

% Compute the FPA

d1 = T(i).earthpointmass.Sat1\_Earth\_HMAG;

d2 = T(i).earthpointmass.Sat1\_Earth\_RMAG;

d3 = T(i).earthpointmass.Sat1\_EarthMJ2000Eq\_VMAG;

T(i).earthpointmass.Sat1\_EarthMJ2000Eq\_FPA = acosd(d1 ./ d2 ./ d3);

dd = find(T(i).earthpointmass.Sat1\_Earth\_TA > 180);

T(i).earthpointmass.Sat1\_EarthMJ2000Eq\_FPA(dd) = -T(i).earthpointmass.Sat1\_EarthMJ2000Eq\_FPA(dd);

d1 = T(i).earthmoon.Sat2\_Earth\_HMAG;

d2 = T(i).earthmoon.Sat2\_Earth\_RMAG;

d3 = T(i).earthmoon.Sat2\_EarthMJ2000Eq\_VMAG;

T(i).earthmoon.Sat2\_EarthMJ2000Eq\_FPA = acosd(d1 ./ d2 ./ d3);

dd = find(T(i).earthmoon.Sat2\_Earth\_TA > 180);

T(i).earthmoon.Sat2\_EarthMJ2000Eq\_FPA(dd) = -T(i).earthmoon.Sat2\_EarthMJ2000Eq\_FPA(dd);

d1 = T(i).earthsun.Sat3\_Earth\_HMAG;

d2 = T(i).earthsun.Sat3\_Earth\_RMAG;

d3 = T(i).earthsun.Sat3\_EarthMJ2000Eq\_VMAG;

T(i).earthsun.Sat3\_EarthMJ2000Eq\_FPA = acosd(d1 ./ d2 ./ d3);

dd = find(T(i).earthsun.Sat3\_Earth\_TA > 180);

T(i).earthsun.Sat3\_EarthMJ2000Eq\_FPA(dd) = -T(i).earthsun.Sat3\_EarthMJ2000Eq\_FPA(dd);

d1 = T(i).earthmoonsun.Sat4\_Earth\_HMAG;

d2 = T(i).earthmoonsun.Sat4\_Earth\_RMAG;

d3 = T(i).earthmoonsun.Sat4\_EarthMJ2000Eq\_VMAG;

T(i).earthmoonsun.Sat4\_EarthMJ2000Eq\_FPA = acosd(d1 ./ d2 ./ d3);

dd = find(T(i).earthmoonsun.Sat4\_Earth\_TA > 180);

T(i).earthmoonsun.Sat4\_EarthMJ2000Eq\_FPA(dd) = -T(i).earthmoonsun.Sat4\_EarthMJ2000Eq\_FPA(dd);

end

% a = []; e = []; rp = []; E = []; h = [];

% for i = 1:4

% t1 = T(i).earthpointmass;

% t2 = T(i).earthmoon;

% t3 = T(i).earthsun;

% t4 = T(i).earthmoonsun;

% for j = 1:4

% if j == 1

% a(i,j) = mean(t1.Sat1\_Earth\_SMA);

% e(i,j) = mean(t1.Sat1\_Earth\_ECC);

% rp(i,j) = mean(t1.Sat1\_Earth\_RadPer);

% En(i,j) = mean(t1.Sat1\_Earth\_Energy);

% h(i,j) = mean(t1.Sat1\_Earth\_HMAG);

% elseif j == 2

% a(i,j) = mean(t2.Sat2\_Earth\_SMA);

% e(i,j) = mean(t2.Sat2\_Earth\_ECC);

% rp(i,j) = mean(t2.Sat2\_Earth\_RadPer);

% En(i,j) = mean(t2.Sat2\_Earth\_Energy);

% h(i,j) = mean(t2.Sat2\_Earth\_HMAG);

% elseif j == 3

% a(i,j) = mean(t3.Sat3\_Earth\_SMA);

% e(i,j) = mean(t3.Sat3\_Earth\_ECC);

% rp(i,j) = mean(t3.Sat3\_Earth\_RadPer);

% En(i,j) = mean(t3.Sat3\_Earth\_Energy);

% h(i,j) = mean(t3.Sat3\_Earth\_HMAG);

% else

% a(i,j) = mean(t4.Sat4\_Earth\_SMA);

% e(i,j) = mean(t4.Sat4\_Earth\_ECC);

% rp(i,j) = mean(t4.Sat4\_Earth\_RadPer);

% En(i,j) = mean(t4.Sat4\_Earth\_Energy);

% h(i,j) = mean(t4.Sat4\_Earth\_HMAG);

% end

% end

% end

% Get the last values for each column

a = []; e = []; rp = []; E = []; h = []; rf = []; vf = []; TA\_f = []; FPA\_f = [];

for i = 1:4

t1 = T(i).earthpointmass;

t2 = T(i).earthmoon;

t3 = T(i).earthsun;

t4 = T(i).earthmoonsun;

for j = 1:4

if j == 1

a(i,j) = t1.Sat1\_Earth\_SMA(end);

e(i,j) = t1.Sat1\_Earth\_ECC(end);

rp(i,j) = t1.Sat1\_Earth\_RadPer(end);

En(i,j) = t1.Sat1\_Earth\_Energy(end);

h(i,j) = t1.Sat1\_Earth\_HMAG(end);

rf(i,j) = t1.Sat1\_Earth\_RMAG(end);

vf(i,j) = t1.Sat1\_EarthMJ2000Eq\_VMAG(end);

TA\_f(i,j) = t1.Sat1\_Earth\_TA(end);

FPA\_f(i,j) = t1.Sat1\_EarthMJ2000Eq\_FPA(end);

elseif j == 2

a(i,j) = t2.Sat2\_Earth\_SMA(end);

e(i,j) = t2.Sat2\_Earth\_ECC(end);

rp(i,j) = t2.Sat2\_Earth\_RadPer(end);

En(i,j) = t2.Sat2\_Earth\_Energy(end);

h(i,j) = t2.Sat2\_Earth\_HMAG(end);

rf(i,j) = t2.Sat2\_Earth\_RMAG(end);

vf(i,j) = t2.Sat2\_EarthMJ2000Eq\_VMAG(end);

TA\_f(i,j) = t2.Sat2\_Earth\_TA(end);

FPA\_f(i,j) = t2.Sat2\_EarthMJ2000Eq\_FPA(end);

elseif j == 3

a(i,j) = t3.Sat3\_Earth\_SMA(end);

e(i,j) = t3.Sat3\_Earth\_ECC(end);

rp(i,j) = t3.Sat3\_Earth\_RadPer(end);

En(i,j) = t3.Sat3\_Earth\_Energy(end);

h(i,j) = t3.Sat3\_Earth\_HMAG(end);

rf(i,j) = t3.Sat3\_Earth\_RMAG(end);

vf(i,j) = t3.Sat3\_EarthMJ2000Eq\_VMAG(end);

TA\_f(i,j) = t3.Sat3\_Earth\_TA(end);

FPA\_f(i,j) = t3.Sat3\_EarthMJ2000Eq\_FPA(end);

else

a(i,j) = t4.Sat4\_Earth\_SMA(end);

e(i,j) = t4.Sat4\_Earth\_ECC(end);

rp(i,j) = t4.Sat4\_Earth\_RadPer(end);

En(i,j) = t4.Sat4\_Earth\_Energy(end);

h(i,j) = t4.Sat4\_Earth\_HMAG(end);

rf(i,j) = t4.Sat4\_Earth\_RMAG(end);

vf(i,j) = t4.Sat4\_EarthMJ2000Eq\_VMAG(end);

TA\_f(i,j) = t4.Sat4\_Earth\_TA(end);

FPA\_f(i,j) = t4.Sat4\_EarthMJ2000Eq\_FPA(end);

end

end

end

% Get array for data

arr1 = [a(1,:);e(1,:);rp(1,:);En(1,:);h(1,:);rf(1,:);vf(1,:);TA\_f(1,:);FPA\_f(1,:)];

arr2 = [a(2,:);e(2,:);rp(2,:);En(2,:);h(2,:);rf(2,:);vf(2,:);TA\_f(2,:);FPA\_f(2,:)];

arr3 = [a(3,:);e(3,:);rp(3,:);En(3,:);h(3,:);rf(3,:);vf(3,:);TA\_f(3,:);FPA\_f(3,:)];

arr4 = [a(4,:);e(4,:);rp(4,:);En(4,:);h(4,:);rf(4,:);vf(4,:);TA\_f(4,:);FPA\_f(4,:)];

% Convert arrays to table

M1 = array2table(arr1);

M2 = array2table(arr2);

M3 = array2table(arr3);

M4 = array2table(arr4);

% Save table data as excel file

warning('off','MATLAB:xlswrite:AddSheet'); %optional

writetable(M1,fullfile(fdir, 'p1\_data.xlsx'),'Sheet',1);

writetable(M2,fullfile(fdir, 'p1\_data.xlsx'),'Sheet',2);

writetable(M3,fullfile(fdir, 'p1\_data.xlsx'),'Sheet',3);

writetable(M4,fullfile(fdir, 'p1\_data.xlsx'),'Sheet',4);

% Plotting

for i = 1:4

t1 = T(i).earthpointmass;

t2 = T(i).earthmoon;

t3 = T(i).earthsun;

t4 = T(i).earthmoonsun;

fig = figure("Renderer","painters", "Position",[10 10 900 1000]);

subplot(2,2,1)

plot(t1.Sat1\_Earth\_Altitude)

title('EarthPointMass')

grid on; grid minor; box on;

subplot(2,2,2)

plot(t2.Sat2\_Earth\_Altitude)

title('EarthMoon')

grid on; grid minor; box on;

subplot(2,2,3)

plot(t3.Sat3\_Earth\_Altitude)

title('EarthSun')

grid on; grid minor; box on;

subplot(2,2,4)

plot(t4.Sat4\_Earth\_Altitude)

title('EarthMoonSun')

grid on; grid minor; box on;

% Give common xlabel and ylabel to your figure

han=axes(fig,'visible','off');

han.XLabel.Visible='on';

han.YLabel.Visible='on';

xlabel(han,'sample points in the order of epoch');

ylabel(han,'altitude [km]');

ch\_str = ["02 Oct 2020", "07 Oct 2020", "11 Oct 2020", "30 Oct 2020"];

title\_str = 'Altitude: 60 Day History From ' + ch\_str(i) + ' - T. Koike';

sgtitle(title\_str)

file\_str = 'alt' + ch\_str(i) + '.png';

saveas(fig, fullfile(fdir, file\_str));

end

%% AAE 532 HW 5 Problem 2

% Tomoki Koike

close all; clear all; clc;

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps5';

set(groot, 'defaulttextinterpreter','latex');

set(groot, 'defaultAxesTickLabelInterpreter','latex');

set(groot, 'defaultLegendInterpreter','latex');

format longG;

% Set constants

planet\_consts = setup\_planetary\_constants(); % Function that sets up all the constants in the table

sun = planet\_consts.sun; % structure of sun

earth = planet\_consts.earth; % structure of earth

mars = planet\_consts.mars; % structure of mars

G = 6.6743015e-20; % Gravitational constant [km^3/kg/s^2]

% (a) Calculate parameters

rp = 1.5\*mars.mer; % radius of periapsis

ra = 6.5\*mars.mer; % radius of apoapsis

M = -90; % mean anomaly

a = 0.5\*(rp + ra); % semi major axis

e = (ra - rp) / (ra + rp); % eccentricity

p = a \* (1 - e^2); % semi latus rectum

mu = mars.gp; % gravitational parameter

h = sqrt(mu \* p); % specific angular momentum

IP = 2 \* pi \* sqrt(a^3 / mu); % period

IP\_days = IP / 60 / 60 / 24; % period in days

En = -mu / 2 / a; % specific energy

E = rad2deg(M2E\_anomaly(M, e, "deg")) % eccentric anomaly

E0 = E;

E = 360 + E

TA = E2T\_anomaly(e,E,"deg") % true anomaly

r = p / (1 + e \* cosd(TA)) % position

v = vis\_viva(r, a, mu) % velocity

gamma = -acosd(h / r / v) % flight path angle

del\_t = deg2rad(M + 360) \* sqrt(a^3 / mu) % time elapse

del\_t\_day = del\_t / 60 / 60 / 24

% (b)

r0vec = r \* [cosd(TA), sind(TA)];

v0vec = v \* [cosd(TA + 90 + abs(gamma)), sind(TA + 90 + abs(gamma))];

% (c)

Mf = 90;

Ef = rad2deg(M2E\_anomaly(Mf, e, "deg"));

TAf = E2T\_anomaly(e, Ef, "deg");

rfmag = p / (1 + e \* cosd(TAf));

[rfvec, vfvec, f, g, fdot, gdot] = FandG\_elp(a, mu, deg2rad(Ef-E0), 0.5\*IP, r0vec, v0vec);

[rfvec2, vfvec2, f2, g2, fdot2, gdot2] = FandG\_conic(p, mu, deg2rad(TAf-(TA-360)),r0vec,v0vec,rfmag);

% (d)

Th = -pi:0.01:pi;

R = p ./ (1 + e \* cos(Th)); X = R .\* cos(Th); Y = R .\* sin(Th);

fig = figure("Renderer","painters", "Position",[10 10 900 700]);

plot(X, Y, '-b')

title('Elliptical Orbit of $e=0.625$ Around Mars - T. Koike')

hold on; grid on; grid minor; box on; axis equal;

plot(r\*cosd(TA), r\*sind(TA), '.r', 'MarkerSize', 18)

plot(rfmag\*cosd(TAf), rfmag\*sind(TAf), '.r', 'MarkerSize', 18)

% Mars

plot(0, 0, '.m', 'MarkerSize', 30)

% Axes

nanikore = -ra:rp; korenani = -a\*sqrt(1-e^2):a\*sqrt(1-e^2);

plot(nanikore, zeros(size(nanikore)), '--k')

plot(-a\*e\*ones(size(korenani)), korenani, '--k')

hold off

xlabel('$\hat{e}$ [km]')

ylabel('$\hat{p}$ [km]')

saveas(fig, fullfile(fdir, 'p2\_orbit.png'));

% AAE 532 HW 5 Problem 3

% Tomoki Koike

close all; clear all; clc;

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps5';

set(groot, 'defaulttextinterpreter','latex');

set(groot, 'defaultAxesTickLabelInterpreter','latex');

set(groot, 'defaultLegendInterpreter','latex');

format longG;

% Set constants

planet\_consts = setup\_planetary\_constants(); % Function that sets up all the constants in the table

earth = planet\_consts.earth; % structure of earth

G = 6.6743015e-20; % Gravitational constant [km^3/kg/s^2]

Re = earth.mer;

a = 20\*Re % semi major axis

RA = 45; % right ascension

e = 0.6; % eccentricity

refA = 235; % anomaly from reference line

AP = 30; % argument of periapsis

i = 34; % inclination

mu = earth.gp

C\_pq = [cosd(refA), sind(refA), 0; -sind(refA), cosd(refA), 0; 0,0,1];

C\_qn = [1,0,0;0,cosd(i),sind(i);0,-sind(i),cosd(i)];

C\_ni = [cosd(RA),sind(RA),0;-sind(RA),cosd(RA),0;0,0,1];

% i: inertial frame, n: n-frame, q: q-frame, p: orbital frame (polar)

TA = refA - AP % true anomaly

p = a \* (1 - e^2); % semi latus rectum

r1 = p / (1 + e \* cosd(TA));

v1 = vis\_viva(r1, a, mu);

h = sqrt(mu \* p); % specific angular momentum

FPA = -acosd(h / r1 / v1); % flight path angle

IP = 2 \* pi \* sqrt(a^3 / mu); % period

IP\_day = IP / 60 / 60 / 24;

E = T2E\_anomaly(e, TA, "deg"); % eccentric anomaly

M = rad2deg(deg2rad(E) - e\*sind(E)); % mean anomaly

dt\_per = deg2rad(M) \* sqrt(a^3 / mu); % time elapse from periapsis

dt\_per\_day = dt\_per / 60 / 60 / 24;

r1vec\_p = r1 \* [1,0,0]; % position vector in orbital frame

v1vec\_p = v1 \* [sind(FPA), cosd(FPA), 0]; % velocity vector in orbital frame

r1vec\_n = r1vec\_p \* C\_pq \* C\_qn;

v1vec\_n = v1vec\_p \* C\_pq \* C\_qn;

r1vec\_i = r1vec\_n \* C\_ni;

v1vec\_i = v1vec\_n \* C\_ni;

% syms theta i Omega

% assume([theta, i, Omega], 'real');

% Cm1 = [cos(theta), sin(theta), 0; -sin(theta), cos(theta), 0; 0,0,1];

% Cm2 = [1,0,0;0,cos(i),sin(i);0,-sin(i),cos(i)];

% Cm3 = [cos(Omega),sin(Omega),0;-sin(Omega),cos(Omega),0;0,0,1];

% Cm1\*Cm2

% (c)

dt2 = 3\*24\*60\*60;

M2 = mod((rad2deg(sqrt(mu / a^3) \* dt2)+M),360)

E2 = rad2deg(M2E\_anomaly(M2,e,"deg"))

TA2 = E2T\_anomaly(e,E2,"deg")

dTA12 = TA2 - (TA-360)

dE12 = E2 - (E-360)

[rvec2, vvec2, f2, g2, fdot2, gdot2] = FandG\_elp(a,mu,deg2rad(dE12),dt2,r1vec\_i,v1vec\_i)

FPA2 = acosd(h / norm(rvec2) / norm(vvec2))

% AAE 532 HW 5 Problem 3

% Tomoki Koike

close all; clear all; clc;

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps5';

set(groot, 'defaulttextinterpreter','latex');

set(groot, 'defaultAxesTickLabelInterpreter','latex');

set(groot, 'defaultLegendInterpreter','latex');

format shortG;

% Set constants

planet\_consts = setup\_planetary\_constants(); % Function that sets up all the constants in the table

earth = planet\_consts.earth; % structure of earth

G = 6.6743015e-20; % Gravitational constant [km^3/kg/s^2]

% Setup

Re = earth.mer;

rvec\_i = Re\*[0.15, -1.44, -0.65]; % the position vector in inertial frame

vvec\_i = [6.62, 2.7, -1.56]; % the velocity vector in inertial frame

mu = earth.gp; % gravitational parameter

r = norm(rvec\_i) % magnitude of the position vector

v = norm(vvec\_i) % magnitude of the velocity vector

hvec\_i = cross(rvec\_i, vvec\_i) % the h vector perpendicular to the position and velocity vectors

hhat\_i = hvec\_i / norm(hvec\_i) % the unit h vector in inertial frame

h = norm(hvec\_i) % magnitude of the h vector

rhat\_i = rvec\_i / r % the unit r vector in inertial frame

thetahat\_i = cross(hhat\_i, rhat\_i) % the unit vector of theta direction

check = v < sqrt(2 \* mu / r) % check to see if this orbit is an ellipse

a = (-mu / 2) / (v^2/2 - mu/r) % semi major axis

e = sqrt(1 - h^2 / mu / a) % the eccentricity

rp = a\*(1 - e) % radius of periapsisc

ra = a\*(1 + e) % radius of apoapsis

p = a\*(1 - e^2) % semi latus rectum

% Check that there are no collisions with the Earth

check = rp > Re

check = ra > Re

thetastar = acos\_dbval(1/e \* (p/r - 1), "deg") % double valued true anomaly

i = acos\_dbval(hhat\_i(3), "deg")

i = i(0 <= i & i <= 180) % inclination

Omega1 = asin\_dbval(hhat\_i(1) / sind(i), "deg")

Omega2 = -acos\_dbval(hhat\_i(2) / sind(i), "deg")

Omega = intersect(round(Omega1,5), round(Omega2,5))

theta1 = asin\_dbval(rhat\_i(3) / sind(i), "deg")

theta2 = acos\_dbval(thetahat\_i(3) / sind(i), "deg")

theta = intersect(round(theta1,5), round(theta2,5))

rdot = dot(vvec\_i, rhat\_i)

thetastar = thetastar(thetastar < 0)

omega = theta - thetastar

gamma = -acosd(h /r / v)

E = T2E\_anomaly(e, thetastar, "deg")

M = rad2deg(deg2rad(E) - e\*sind(E))

dtp = deg2rad(M+360)\*sqrt(a^3 / mu)

dtp\_day = dtp / 60 / 60/ 24

ang = [0:0.01:2\*pi];

R = p./(1 + e\*cos(ang));

X = R .\* cos(ang);

Y = R .\* sin(ang);

plot(X,Y)

axis equal

function res = asin\_dbval(x, unit)

if unit == "deg"

ang1 = asind(x);

if (0<=ang1 && ang1<=180)

ang2 = 180 - ang1;

elseif -90<=ang1 && ang1<0

ang2 = -ang1 - 180;

else

ang2 = 540 - ang1;

end

else

ang1 = asin(x);

if (0<=ang1 && ang1<=pi)

ang2 = pi - ang1;

elseif -pi/2<=ang1 && ang1<0

ang2 = -ang1 - pi;

else

ang2 = 3\*pi - ang1;

end

end

res = [ang1, ang2];

end

function res = acos\_dbval(x, unit)

if unit == "deg"

ang1 = acosd(x);

if (0<=ang1 && ang1<=180)

ang2 = -ang1;

else

ang2 = 360 - ang1;

end

else

ang1 = asin(x);

if (0<=ang1 && ang1<pi)

ang2 = -ang1;

else

ang2 = 2\*pi - ang1;

end

end

res = [ang1, ang2];

end