College of Engineering
School of Aeronautics and Astronautics

AAE 532
Orbital Mechanics

PS 7
Orbital 3D Maneuvers and Transfers

Author:
Tomoki Koike

Supervisor:
K. C. Howell

October 30th, 2020 Friday
Purdue University
West Lafayette, Indiana

Problem 1: As part of an interplanetary mission, a spacecraft is in the following orbit around Mars (relative to a Mars centered equatorial J2000 coordinate frame):

$$a = 6R_{♂}, \qquad \Omega = 45°$$

$$e = 0.5, \qquad \theta^* = -165°$$

$$\omega = 30°, \qquad i = 30°$$

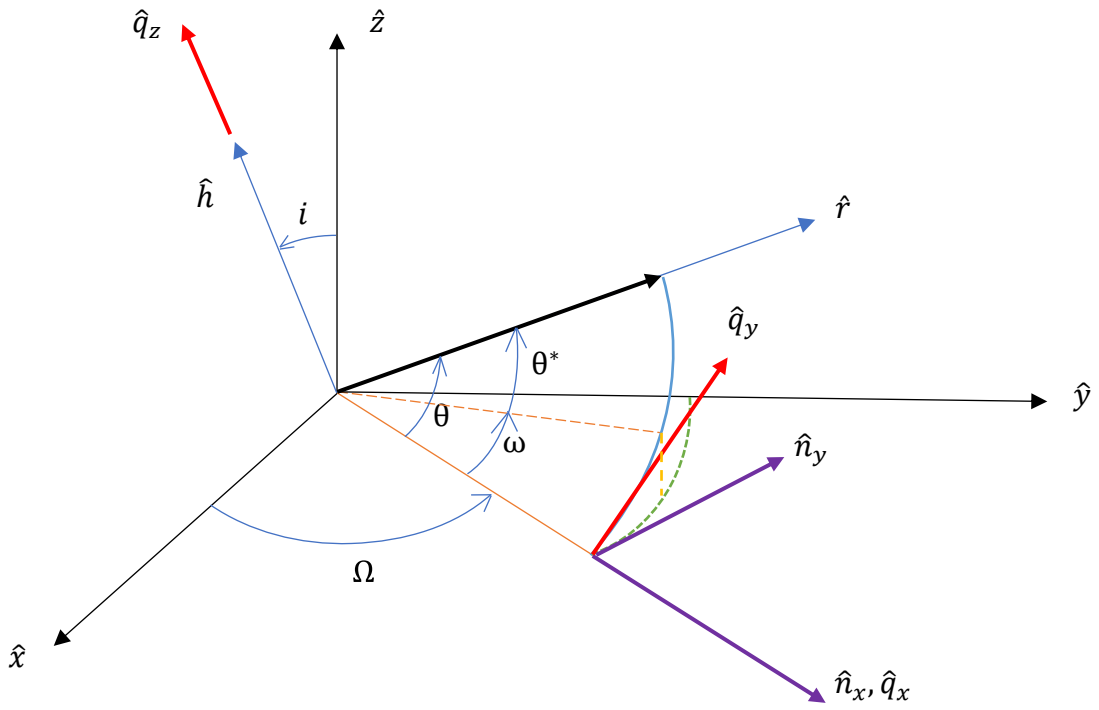
When $\theta^* = 150°$, the following maneuver is implemented:

$$\Delta\bar{v} = 0.1\hat{x} + 0.25\hat{y} + 0.35\hat{z} \quad km/s$$

(a) Transform $\Delta\bar{v}$ into $\hat{r}, \hat{\theta}, \hat{h}$ and *VNB* components corresponding to the original orbit. How much of the $\Delta\bar{v}$ is out-of-plane? What is the value as a % of the total $|\Delta\bar{v}|$? (Define this out-of-plane component as $\Delta\bar{v}_h$.)
Define $\Delta\bar{v}_{r\theta}$ as the projection of $\Delta\bar{v}$ in the orbital plane. Determine $\Delta\bar{v}_{r\theta}$, β, ϕ.
Define $\Delta\bar{v}_{BV}$; is it equal to $\Delta\bar{v}_{r\theta}$? Determine α between the velocity vector in the original orbit and $\Delta\bar{v}_{BV}$.

All the calculations for this problem are done by MATLAB. The code is in the appendix.

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



From this diagram we can deduce the rotational transformation from the inertial to the orbital frame. Which is,

$$(\hat{r} \quad \hat{\theta} \quad \hat{h}) = (\hat{q}_x \quad \hat{q}_y \quad \hat{q}_z)\begin{pmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$(\hat{q}_x \quad \hat{q}_y \quad \hat{q}_z) = (\hat{n}_x \quad \hat{n}_y \quad \hat{n}_z)\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & -s_i \\ 0 & s_i & c_i \end{pmatrix}$$

$$(\hat{n}_x \quad \hat{n}_y \quad \hat{n}_z) = (\hat{x} \quad \hat{y} \quad \hat{z})\begin{pmatrix} c_\Omega & -s_\Omega & 0 \\ s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\therefore (\hat{r} \quad \hat{\theta} \quad \hat{h}) = (\hat{x} \quad \hat{y} \quad \hat{z})\begin{pmatrix} c_\Omega & -s_\Omega & 0 \\ s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & -s_i \\ 0 & s_i & c_i \end{pmatrix}\begin{pmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Since we know all the angles

$$\Omega = 45°$$

$$i = 30°$$

$$\theta = \theta^* + \omega = 150° + 30° = 180°$$

we can transform the coordinates to the orbital frame.

Using a MATLAB function, we are able to conduct the transformation. (*The function code is on page 12.)

$$(0.1 \quad 0.25 \quad 0.35)\begin{pmatrix} c_\Omega & -s_\Omega & 0 \\ s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & -s_i \\ 0 & s_i & c_i \end{pmatrix}\begin{pmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$= (-0.24749 \quad -0.26686 \quad 0.25008)$$

Thus,

$$\Delta\bar{v} = -0.24749\hat{r} - 0.26686\hat{\theta} + 0.25008\hat{h} \ \ km/s\,.$$

$$|\Delta\bar{v}| = 0.44159 \ \ km/s\,.$$

Now from the $\Delta\bar{v}$ value we can see that the $\hat{h}$ component is a positive direction and has a magnitude of $0.25008\ km/s$. Then find the % of the magnitude out-of-plane compared to the total magnitude of the vector.

Which makes it,

$$56.631\%\ .$$

Then, the projection of $\Delta\bar{v}$ onto the orbital plane is

$$\Delta\bar{v}_{r\theta} = -0.24749\hat{r} - 0.26686\hat{\theta}\ \ km/s\ .$$

To find the coordinates in the *VNB* frame, we have to find the velocity and flight path angle at the given true anomaly.

$$p = a(1-e^2) = \ 15287\ km\ .$$

$$r = \frac{p}{1+ecos(\theta^*)} = 26961\ km\ .$$

$$v = \sqrt{\mu\left(\frac{2}{r}-\frac{1}{a}\right)} = 1.0372\ \ km/s \qquad \because \mu = 42828\ \ km^3/s^2$$

$$\gamma = arccos\left(\frac{\sqrt{\mu p}}{rv}\right) = 23.794°\ .$$

Thus, we know that

$$\bar{v}_1 = v_1\left(sin\gamma_1\hat{r}_1 + cos\gamma_1\hat{\theta}_1\right) = 0.41846\hat{r} + 0.94904\hat{\theta}\ \ km/s\ .$$

(*the # 1 denotes that it is prior to the maneuver).

$$\alpha = arccos\left(\frac{\bar{v}_1\cdot\Delta\bar{v}_{r\theta}}{|\bar{v}_1||\Delta\bar{v}_{r\theta}|}\right) = \pm 160.95°$$

From notes JS_3Dex 2, we know the relationship

$$\Delta\bar{v} = \Delta v\left(cos\beta sin\phi\hat{r} + cos\beta cos\phi\hat{\theta} + sin\beta\hat{h}\right)\ .$$

Therefore, comparing it with

$$\Delta\bar{v} = -0.24749\hat{r} - 0.26686\hat{\theta} + 0.25008\hat{h}\ \ km/s\ .$$

we can tell

$$\beta = arcsin\left(\frac{0.25008}{0.44159}\right) = 34.493°\ \text{ or }\ .$$

and

$$\phi_1 = arccos\left(\frac{-0.26686}{0.44159cos\beta}\right) = \pm 136.16°.$$

$$\phi_2 = arcsin\left(\frac{-0.24749}{0.44159cos\beta}\right) = \pm 136.16°.$$

$$\phi = \ -137.16°$$

Then, we can verify that

$$\phi = \alpha + \gamma \Rightarrow \alpha = -160.95°.$$

Then using the formula provided in notes JS_3Dex 3, we can compute the velocity vector in the *VNB* coordinate

$$\Delta\bar{v} = \Delta v\left(cos\beta cos\alpha\hat{V} + cos\beta sin\alpha\hat{B} + sin\beta\hat{N}\right)$$

$$\Delta\bar{v} = -0.34402\hat{V} - 0.11879\hat{B} + 0.25008\hat{N}\ \ km/s\ .$$

Then the projection of the maneuver velocity in the *VNB* frame onto the orbital plane becomes

$$\Delta\bar{v}_{BV} = -0.34402\hat{V} - 0.11879\hat{B}\ km/s\ .$$

(b) To apply the maneuver, all positions, velocities, and $\Delta\bar{v}$'s must be written in terms of the same set of unit vectors, such as the inertial unit vectors $\hat{x}, \hat{y}, \hat{z}$. Determine the new $\bar{r}^+, \bar{v}^+$ immediately after the maneuver.

From part (a), we know the position vector immediately before the maneuver and $\bar{r}^+$ is equivalent to it.

$$\bar{r}_1 = \bar{r}^- = 26961\hat{r}_1$$

We can convert this to the inertial frame using the transposed direction cosines matrices with a reversed sequence from part (a)

$$(\hat{q}_x \quad \hat{q}_y \quad \hat{q}_z) = (\hat{r} \quad \hat{\theta} \quad \hat{h})\begin{pmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$(\hat{n}_x \quad \hat{n}_y \quad \hat{n}_z) = (\hat{q}_x \quad \hat{q}_y \quad \hat{q}_z)\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & s_i \\ 0 & -s_i & c_i \end{pmatrix}$$

$$(\hat{x} \quad \hat{y} \quad \hat{z}) = (\hat{n}_x \quad \hat{n}_y \quad \hat{n}_z)\begin{pmatrix} c_\Omega & s_\Omega & 0 \\ -s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$(26961 \quad 0 \quad 0)\begin{pmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & s_i \\ 0 & -s_i & c_i \end{pmatrix}\begin{pmatrix} c_\Omega & s_\Omega & 0 \\ -s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix} = (-19064 \quad -19064 \quad 0)$$

$$\bar{r}_1 = \bar{r}^- = -19064\hat{x} - 19064\hat{y}\ \ km\ .$$

Thus,

$$\bar{r}_N = \bar{r}^+ = -19064\hat{x} - 19064\hat{y}\ \ km\ .$$

Similarly, the velocity vector before the maneuver can be expressed in the inertial frame

$$\bar{v}_1 = 0.41846\hat{r} + 0.94904\hat{\theta}\ \ km/s$$

$$(0.41846 \quad 0.94904 \quad 0)\begin{pmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & s_i \\ 0 & -s_i & c_i \end{pmatrix}\begin{pmatrix} c_\Omega & s_\Omega & 0 \\ -s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$= (0.28527 \quad -0.87706 \quad -0.47452)$$

$$\bar{v}_1 = \bar{v}^- = 0.28527\hat{x} - 0.87706\hat{y} - 0.47452\hat{z}\ \ km/s\ .$$

The velocity is then,

$$\bar{v}^+ = \bar{v}^- + \Delta\bar{v} = 0.38527\hat{x} - 0.62706\hat{y} - 0.12452\hat{z}\ \ km/s\ .$$

(c) Determine the orbital elements of the new orbit, i.e. $a^+, e^+, i^+, \Omega^+, \omega^+, \theta^{*+}$.

The information we know about the new orbit are

$$\bar{r}_N = \bar{r}^+ = -19064\hat{x} - 19064\hat{y}\ \ km\ .$$

$$\bar{v}^+ = 0.38527\hat{x} - 0.62706\hat{y} - 0.12452\hat{z}\ \ km/s\ .$$

Then,

$$r_2 = r^+ = 26961km, \quad v_2 = v_N = v^+ = 0.74642\ \ km/s\ .$$

$$\hat{r}_2 = -0.70711\hat{x} - 0.70711\hat{y}\ .$$

From, the equation of specific energy

$$\frac{-\mu}{2a} = \frac{v_2^2}{2} - \frac{\mu}{r_2} \Rightarrow a^+ = 16347\ \ km\ .$$

The specific angular momentum vector is

$$\bar{h} = \bar{r}_2 \times \bar{v}_2 \Rightarrow h = |\bar{h}| = \sqrt{\mu a(1-e^2)} \Rightarrow e^+ = 0.67224\ .$$

$$\hat{h} = \frac{\bar{h}}{|\bar{h}|} = 0.12118\hat{x} - 0.12118\hat{y} + 0.98521\hat{z}\ .$$

$$\hat{\theta}_2 = \hat{h} \times \hat{r}_2 = 0.69665\hat{x} - 0.69665\hat{y} - 0.17138\hat{z}\ .$$

Now using the rotational transformation of frames

	$\hat{r}$	$\hat{\theta}$	$\hat{h}$
$\hat{x}$	$c_\Omega c_\theta - s_\Omega c_i s_\theta$	$-c_\Omega s_\theta - s_\Omega c_i s_\theta$	$s_\Omega s_i$
$\hat{y}$	$s_\Omega c_\theta + c_\Omega c_i s_\theta$	$-s_\Omega s_\theta + c_\Omega c_i s_\theta$	$-c_\Omega s_i$
$\hat{z}$	$s_i s_\theta$	$s_i c_\theta$	c_i

Plugging in the values that we have, we know that

	$\hat{r}$	$\hat{\theta}$	$\hat{h}$
$\hat{x}$	-0.70711	0.69665	0.12118
$\hat{y}$	-0.70711	-0.69665	-0.12118
$\hat{z}$	0	-0.17138	0.98521

Thus, the inclination can be found initially

$$i = arccos(0.98521) = \pm 9.8680°$$

$$\because 0° \leq i \leq 180°$$

$$i^+ = 9.8680°$$

Then, from

$$s_\Omega s_i = 0.12118$$

$$\Omega_1 = 45°, 135°$$

$$-c_\Omega s_i = -0.12118$$

$$\Omega_2 = \pm 45$$

Thus, the common angle value is

$$\Omega^+ = 45°.$$

Similarly

$$s_i s_\theta = -0$$

$$\theta_1 = 0°, 180°$$

$$s_i c_\theta = -0.17138$$

$$\theta_2 = \pm 180°$$

The common angle of the two is

$$\theta^{+} = 180°$$

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

$$\dot{r} = \bar{v} \cdot \hat{r} = (0.38527\hat{x} - 0.62706\hat{y} - 0.12452\hat{z}) \cdot (-0.70711\hat{x} - 0.70711\hat{y})$$

$$\dot{r} = 0.17097 > 0$$

It is smaller than 0, thus in the new orbit at this position it is ascending.

The true anomaly becomes

$$\theta^{*+} = \pm arccos\left(\frac{1}{e}\left(\frac{p}{r} - 1\right)\right) = \pm 173.32°$$

Since the orbit is descending,

$$\theta^{*+} = 173.32°$$

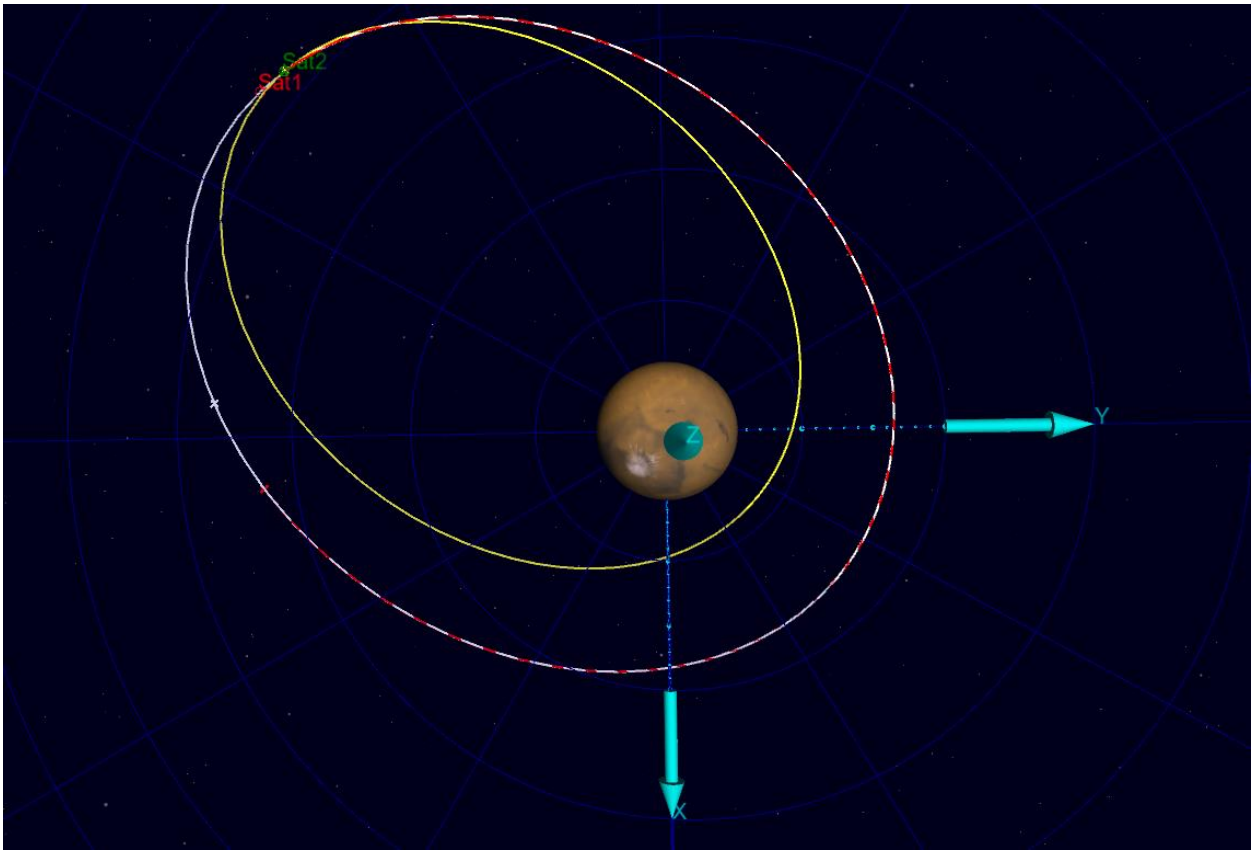
Then the argument of periapsis becomes

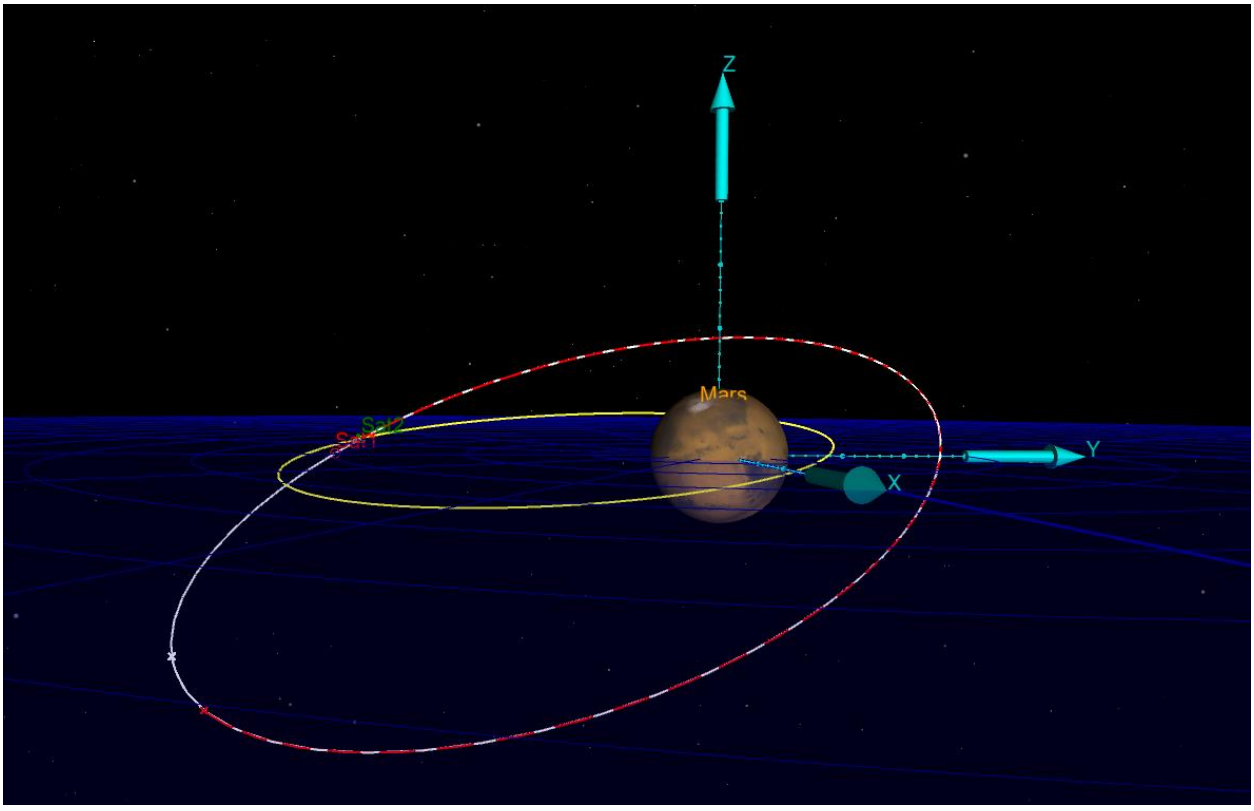
$$\omega^{+} = \theta^{+} - \theta^{*+} = 180° - (173.32°) = 6.6802°$$

(d) Insert the maneuver into GMAT; which GMAT components are the VNB directions? Use the scroll to verify your results.
Plot the new and the old orbits in MATLAB in terms of $\hat{x}, \hat{y}, \hat{z}$ coordinates. (You already have a MATLAB script that plots the orbit in the orbit plane in terms of $\hat{e}, \hat{p}, \hat{h}$; how can you use the transformation matrix to produce the $\hat{x}, \hat{y}, \hat{z}$ coordinates.) Add the following elements to the Matlab plot: Mars equator, Line of Nodes, Orbital Angular Momentum Vector, and Periapsis Vector. Also note the maneuver location.

The GMAT plots the following orbits.

1. The white orbit is the whole orbit of the old orbit
2. The red orbit indicates the trajectory from start point to maneuver point for the old orbit
3. The third orbit is after the impulsive maneuver (cannot see color because it is overlapped by the fourth orbit)
4. The whole orbit of the new orbit in yellow


Sat2
Sat1
Y
X


Z
Mars
Sat2
Sat1
Y
X

The mission summary is the following

```
******  Changes made to the mission will not be reflected ******
******  in the data displayed until the mission is rerun  ******

       Maneuver Command: Maneuver1
       Spacecraft       : Sat1
       Coordinate System: EarthMJ2000Eq

       Time System   Gregorian                    Modified Julian
       ------------------------------------------------------------------------
       UTC Epoch:    01 Nov 2020 05:35:20.210      29154.7328727999
       TAI Epoch:    01 Nov 2020 05:35:57.210      29154.7333010407
       TT  Epoch:    01 Nov 2020 05:36:29.394      29154.7336735407
       TDB Epoch:    01 Nov 2020 05:36:29.392      29154.7336735236

       Cartesian State                      Keplerian State
       ---------------------------          --------------------------------
       X  =   67187690.657896 km            SMA  =  -7179.5258477719 km
       Y  =   18549294.045176 km            ECC  =   4182.4788677641
       Z  =   5746419.3449228 km            INC  =   172.00557086337 deg
       VX =   7.3400676935410 km/sec        RAAN =   159.48727704143 deg
       VY =  -1.2713048886952 km/sec        AOP  =   79.198665800173 deg
       VZ =   0.1940027010593 km/sec        TA   =   64.588436006625 deg
                                            MA   =   504048.27984445 deg
                                            HA   =   85.318215805139 deg

       Spherical State                      Other Orbit Data
       ---------------------------          --------------------------------
       RMAG =   69937711.006318 km          Mean Motion        =   1.037827663e-03 deg/sec
       RA   =   15.433856412879 deg         Orbit Energy       =   27.759524093343 km^2/s^2
       DEC  =   4.7130104820707 deg         C3                 =   55.519048186685 km^2/s^2
       VMAG =   7.4518753957513 km/s        Semilatus Rectum   =   125592368075.48 km
       AZI  =  -96.464736106486 deg         Angular Momentum   =   223743543.73684 km^2/s
       VFPA =   25.423936328097 deg         Beta Angle         =   7.7398693757396 deg
       RAV  =  -9.8261863254201 deg         Periapsis Altitude =   30014657.476725 km
       DECV =   1.4918113109716 deg         VelPeriapsis       =   7.4528922526499 km/s

       Planetodetic Properties              Hyperbolic Parameters
       ---------------------------          --------------------------------
       LST       =   15.703333335696 deg    BdotT          =  -29746518.577520 km
       MHA       =   124.80706103425 deg    BdotR          =  -4103447.9889583 km
       Latitude  =   4.8228061571434 deg    B Vector Angle =  -172.14577958818 deg
       Longitude =  -109.10372769856 deg    B Vector Mag   =   30028214.280586 km
       Altitude  =   69931333.020919 km     DLA            =   1.4916300021686 deg
                                            RLA            =  -9.8275009890494 deg


       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


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

       Spacecraft       : Sat2
       Coordinate System: EarthMJ2000Eq

       Time System   Gregorian                    Modified Julian
       ------------------------------------------------------------------------
       UTC Epoch:    30 Oct 2020 11:59:28.000      29152.9996296296
       TAI Epoch:    30 Oct 2020 12:00:05.000      29153.0000578704
       TT  Epoch:    30 Oct 2020 12:00:37.184      29153.0004303704
```

```
TDB Epoch:    30 Oct 2020 12:00:37.182      29153.0004303530

Cartesian State                          Keplerian State
---------------------------              --------------------------------
X  =   66102437.951333 km                SMA  =  -6848.7852415578 km
Y  =   18665949.773599 km                ECC  =   4661.6073876064
Z  =   5672215.3375788 km                INC  =   173.74080564272 deg
VX =   7.4632995368109 km/sec            RAAN =   146.92477969338 deg
VY =  -1.5551851277323 km/sec            AOP  =   68.568397524816 deg
VZ =   0.3037969960156 km/sec            TA   =   62.418132315899 deg
                                         MA   =   510973.66198796 deg
                                         HA   =   80.456331905921 deg

Spherical State                          Other Orbit Data
---------------------------              --------------------------------
RMAG =   68921143.424179 km              Mean Motion        =   1.113906057e-03 deg/sec
RA   =   15.768572934975 deg             Orbit Energy       =   29.100083258658 km^2/s^2
DEC  =   4.7207864871118 deg             C3                 =   58.200166517316 km^2/s^2
VMAG =   7.6296614192551 km/s            Semilatus Rectum   =   148828092279.41 km
AZI  =  -94.114611785104 deg             Angular Momentum   =   243563017.08218 km^2/s
VFPA =   27.592760724738 deg             Beta Angle         =   8.1156713319580 deg
RAV  =  -11.770724719755 deg             Periapsis Altitude =   31913120.956634 km
DECV =   2.2820001485364 deg             VelPeriapsis       =   7.6305400774945 km/s

Planetodetic Properties                  Hyperbolic Parameters
---------------------------              --------------------------------
LST       =   16.038041809192 deg        BdotT          =  -31761269.737228 km
MHA       =   219.13115319097 deg        BdotR          =  -3242435.2902541 km
Latitude  =   4.8305636382462 deg        B Vector Angle =  -174.17099886810 deg
Longitude =   156.90688861822 deg        B Vector Mag   =   31926347.143581 km
Altitude  =   68914765.439265 km         DLA            =   2.2818581497827 deg
                                         RLA            =  -11.772116544458 deg


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



Maneuver Summary
-----------------
Impulsive Burn:     ImpulsiveBurn1
Spacecraft:         Sat1
Origin:             Mars
Axes:               VNB
Delta V Vector:
   Element 1:   -0.3440200000000 km/s
   Element 2:   -0.1187900000000 km/s
   Element 3:    0.2500800000000 km/s

No mass depletion
```

The values in red indicate the maneuver velocities in the *VNB* coordinates.

Now we will plot the 3d orbit using MATLAB. To transform the orbital frame to $\hat{e}, \hat{p}, \hat{h}$ it requires this process

Since we know that

$$(\hat{q}_x \quad \hat{q}_y \quad \hat{q}_z) = (\hat{r} \quad \hat{\theta} \quad \hat{h}) \begin{pmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$(\hat{n}_x \quad \hat{n}_y \quad \hat{n}_z) = (\hat{q}_x \quad \hat{q}_y \quad \hat{q}_z) \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & s_i \\ 0 & -s_i & c_i \end{pmatrix}$$

$$(\hat{x} \quad \hat{y} \quad \hat{z}) = (\hat{n}_x \quad \hat{n}_y \quad \hat{n}_z) \begin{pmatrix} c_\Omega & s_\Omega & 0 \\ -s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\therefore (\hat{x} \quad \hat{y} \quad \hat{z}) = (\hat{r} \quad \hat{\theta} \quad \hat{h}) \begin{pmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_i & s_i \\ 0 & -s_i & c_i \end{pmatrix} \begin{pmatrix} c_\Omega & s_\Omega & 0 \\ -s_\Omega & c_\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The caveat is that for each position vector in the orbital frame the argument of latitude, θ changes and we have to compute a different Direction Cosine Matrix. The following MAT LAB functions make the computation possible.

```
function C = DCMake(theta, i, Omega)
%{
       NAME:    DCMake
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) theta:  ARGUMENT OF LATITUDE (CAN BE VECTOR/MATRIX)
                (2) i:      INCILNATION
                (3) Omega:  RIGHT ASCENSION OF ASCENDING NODE
       OUTPUTS: (1) C: THE RESULTANT DIRECTION COSINE MATRIX
       DESCRIPTION: CREATES THE DIRECTION COSINE MATRIX FOR 3D ORBIT FROM
                    ORBITAL TO INERTIAL
 %}

    % 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];

    C.o2q = C_oq; C.q2n = C_qn; C.n2i = C_ni;
end
```

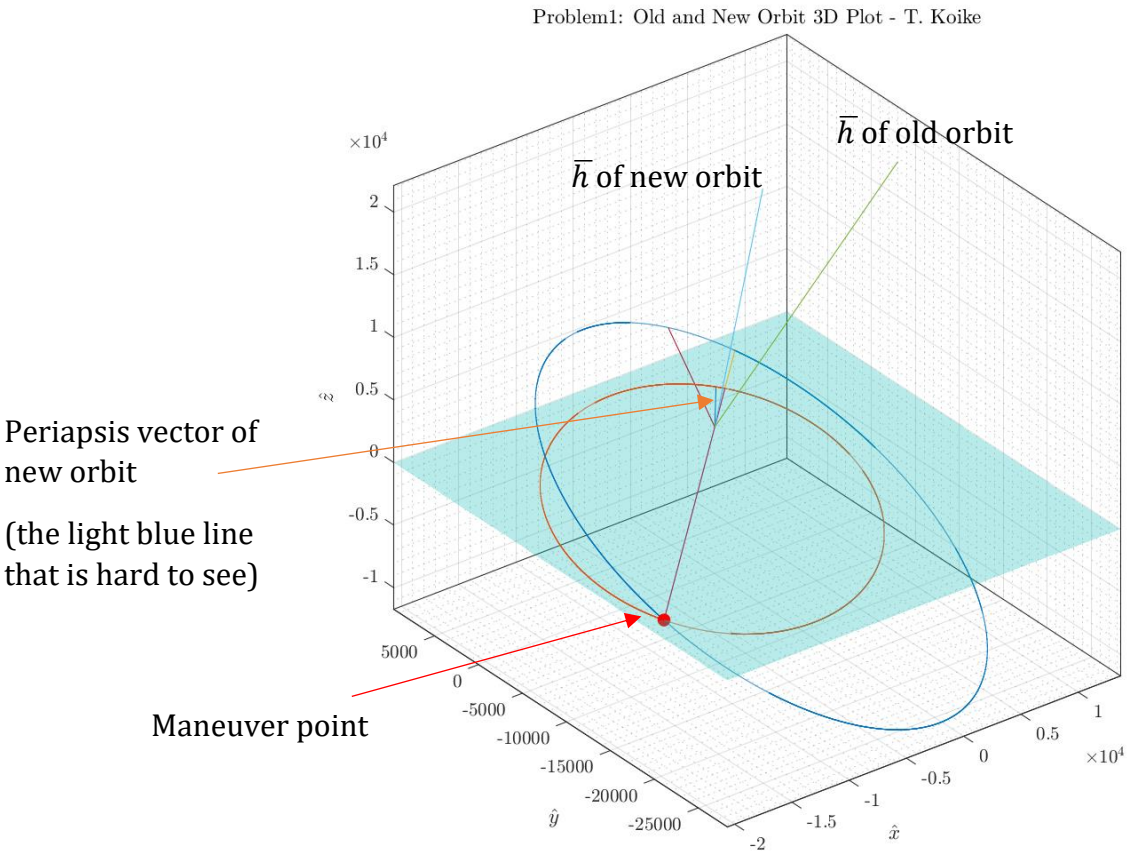
```
function resvec = orbit_frame_transform(theta, i, Omega, vec, frame, unit)
%{
       NAME:    orbit_frame_transform
       AUTHOR:  TOMOKI KOIKE
```

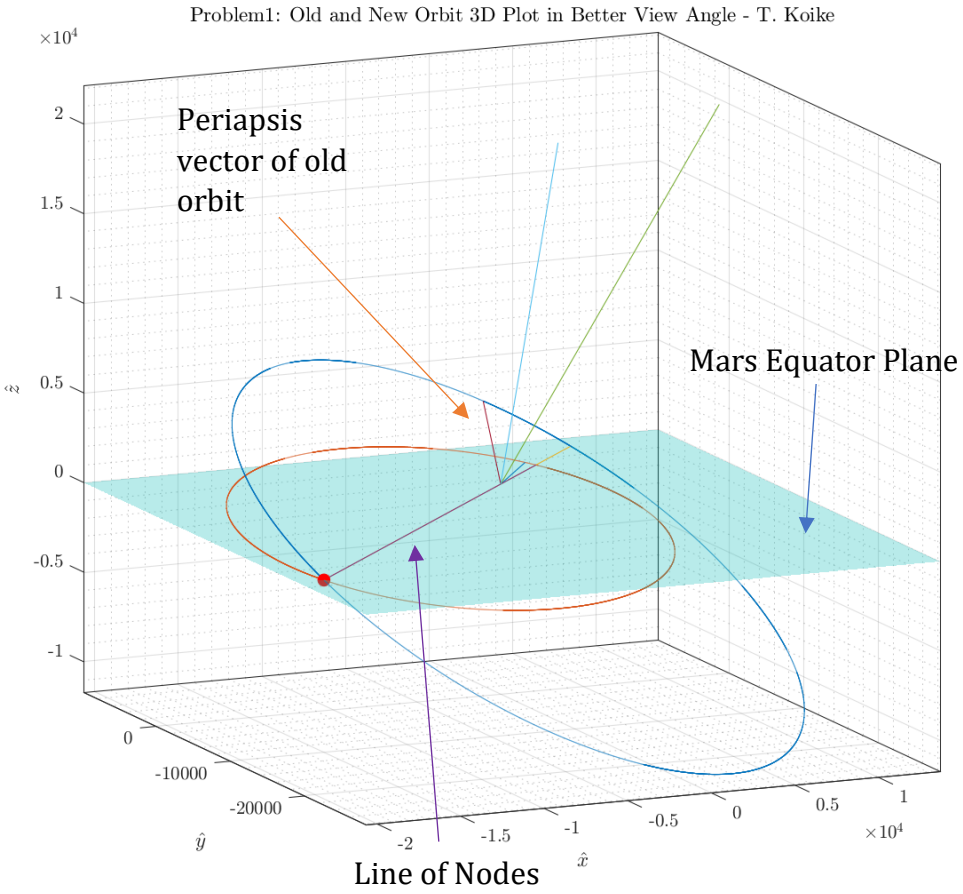
```
       INPUTS:  (1) theta:  ARGUMENT OF LATITUDE (CAN BE VECTOR/MATRIX)
                (2) i:      INCILNATION
                (3) Omega:  RIGHT ASCENSION OF ASCENDING NODE
                (4) vec:    VECTOR TO TRANSFORM
                (5) frame:  THE STARTING FRAME (ORBITAL OR INERTIAL)
                (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

    % Size of the vector
    % if the vec variable is a matrix we need to loop through
    SZ = size(vec);
    sz = SZ(1);

    % Transform
    if frame == "orbital"
       for m = 1:sz
           % Create DCM
           C = DCMake(theta(m), i, Omega);
           % Orbital to q-frame
           resvec.q(m,:) = vec(m,:) * C.o2q;
           % q-frame to n-frame
           resvec.n(m,:) = resvec.q(m,:) * C.q2n;
           % n-frame to inertial
           resvec.i(m,:) = resvec.n(m,:) * C.n2i;
       end
    elseif frame == "inertial"
        for m = 1:sz
           % Create DCM
           C = DCMake(theta(m), i, Omega);
           % Inertial to n-frame
           resvec.n(m,:) = vec(m,:) * C.n2i.';
           % n-frame to q-frame
           resvec.q(m,:) = resvec.n(m,:) * C.q2n.';
           % q-frame to orbital frame
           resvec.o(m,:) = resvec.q(m,:) * C.o2q.';
        end
    else
        error("Enter a frame of either 'orbital' or 'inertial'.");
    end
end
```

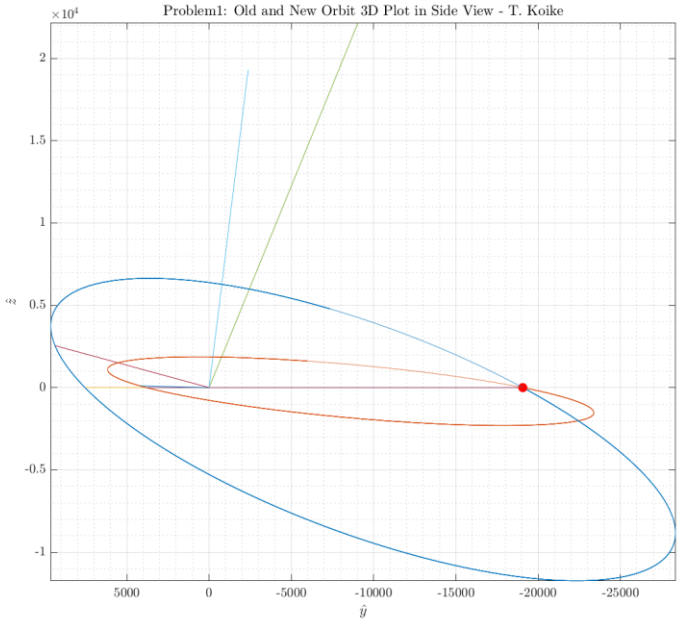
The following function code finds the AN and DN points in the orbit

```
function res = find_AN_DN(posMat)
   %{
       NAME:    find_AN_DN
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) posMat: X, Y, Z POSITIONS
       OUTPUTS: (1) res: STRUCTURE WITH THE AN AND DN POINTS
       DESCRIPTION: FINDS THE ASCENDING NODE AND DESCENDING NODE FOR GIVEN
       DATA POINTS
 %}

    idx = [];
    for i = 1:length(posMat(:,3))-1
        mae = posMat(i, 3);
        ato = posMat(i+1, 3);
        if mae * ato <= 0
            if abs(mae) < abs(ato)
                kore = i;
            else
                kore = i+1;
            end
            idx = [idx, kore];
        end
    end
    idx = unique(idx);
    preidx = idx(1) - 1; postidx = idx(1) + 1;
    pre = posMat(preidx, 3); post = posMat(postidx, 3);
    if (pre < 0) && (post > 0)
        res.AN.values = posMat(idx(1), :);
        res.DN.values = posMat(idx(2), :);
        res.AN.idx = idx(1);
        res.DN.idx = idx(2);
    else
        res.AN.values = posMat(idx(2), :);
        res.DN.values = posMat(idx(1), :);
        res.AN.idx = idx(2);
        res.DN.idx = idx(1);
    end
end
```

Using these functions and some MATLAB code we get the following plots (the execution code is in the appendix).

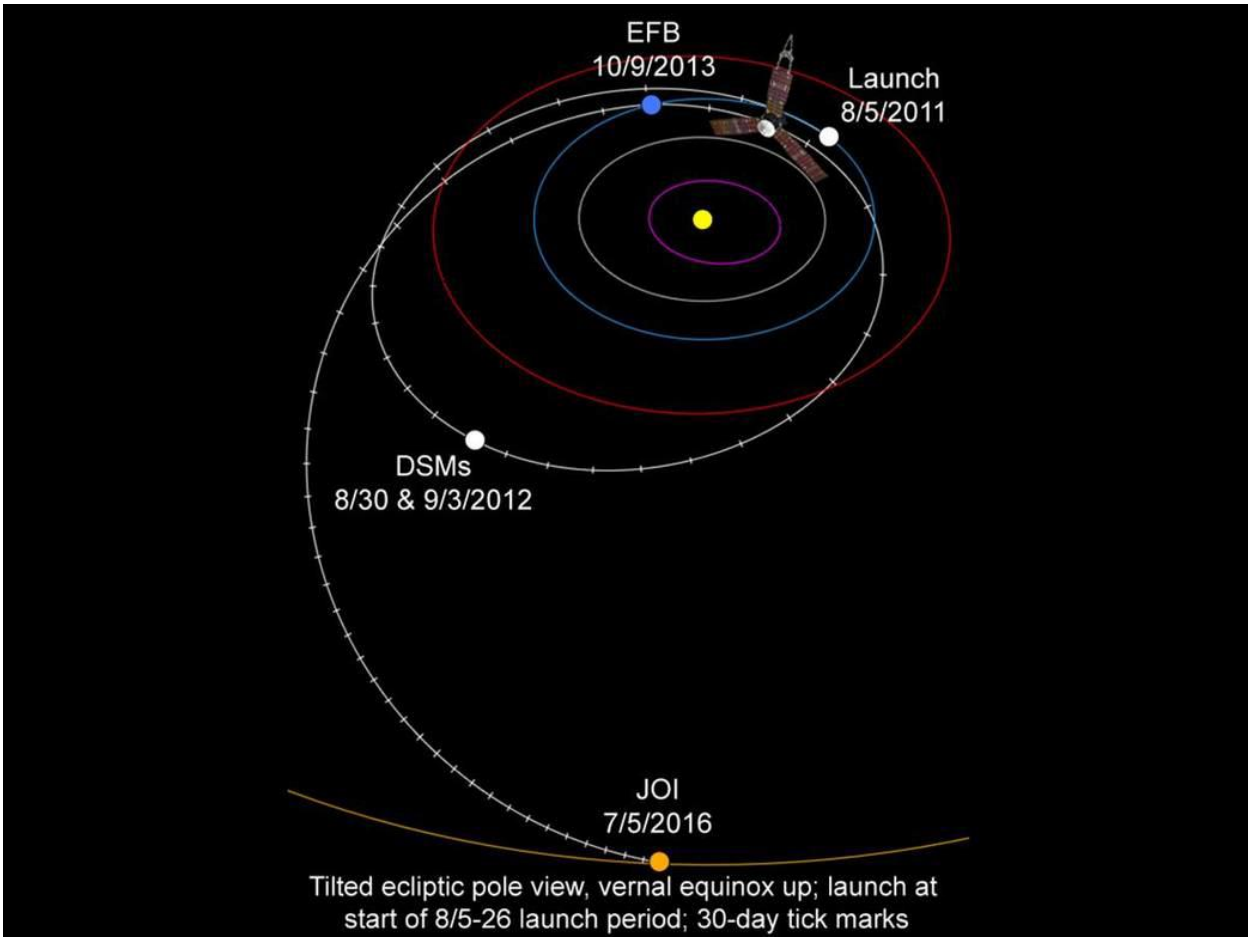



Problem1: Old and New Orbit 3D Plot in Better View Angle - T. Koike
Periapsis vector of old orbit
Mars Equator Plane
Line of Nodes
×10^4
2
1.5
1
0.5
0
-0.5
-1
ẑ
0
-10000
-20000
ŷ
-2
-1.5
-1
-0.5
0
0.5
1
×10^4
x̂


Problem1: Old and New Orbit 3D Plot in Side View - T. Koike
×10^4
2
1.5
1
0.5
0
-0.5
-1
ẑ
5000
0
-5000
-10000
-15000
-20000
-25000
ŷ

Problem 2: The Juno spacecraft launched August 5, 2011 and arrived in the Jovian system July 5, 2016; it remains in operation till July 2021. For preliminary analysis, obtain some information for such a mission. Ignore local gravity fields.

(a) Clearly, the actual transfer to Jupiter is not a Hohmann. But determine the corresponding Julian Dates and the actual TOF in days and years. (Use noon to represent a given Julian date.)

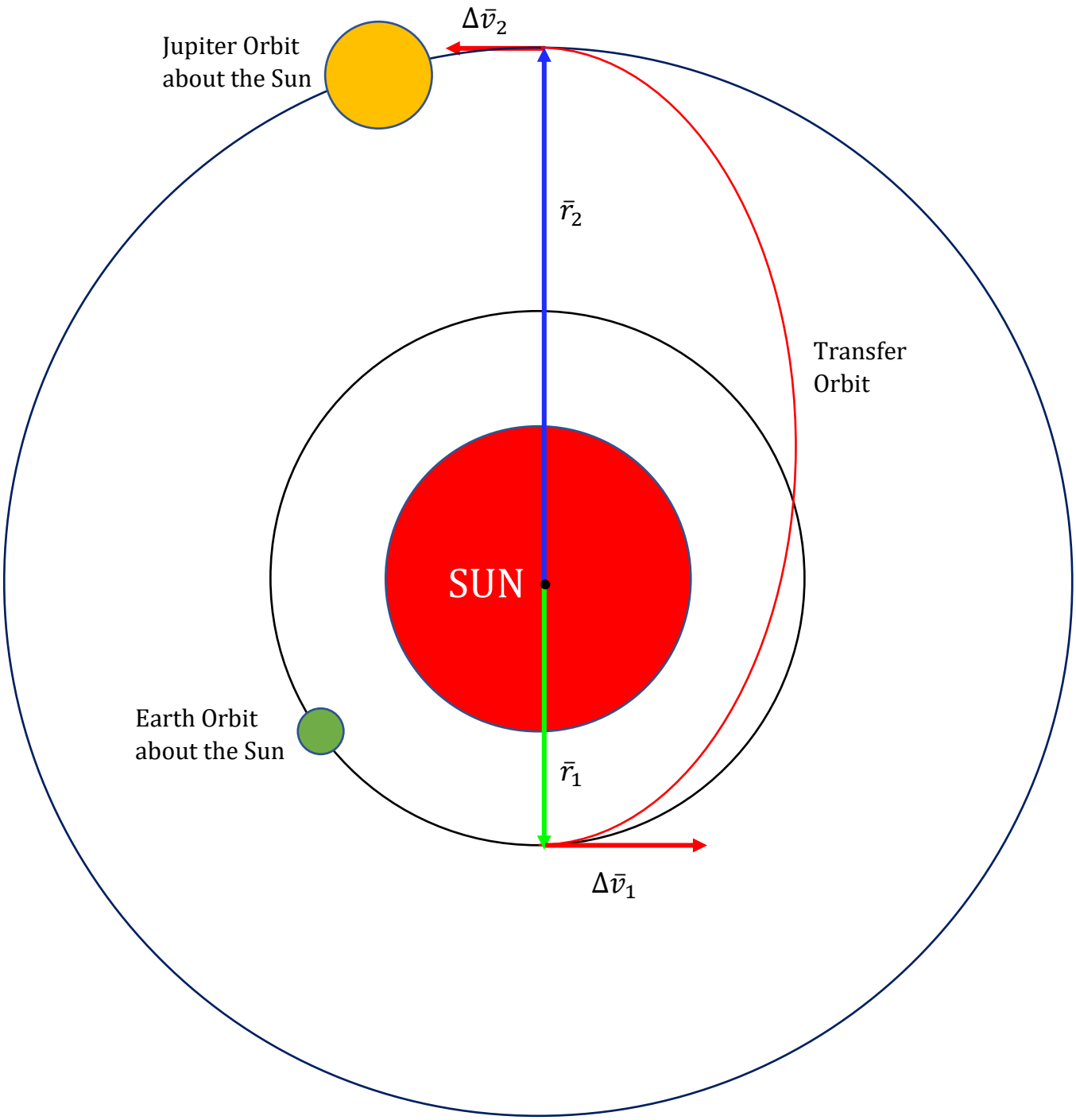


UTC Date	Julian Date
2011-Aug-05 12:00:00	2455779.0000000 UT
2016-Jul-05 12:00:00	2457575.0000000 UT
TOF	
Days	**Years (common year 365 days)**
1797	4.9233

Calculated using the "Time Conversion Tool" of NASA.

(b) Begin by examining at transfer from $\oplus$ to Jupiter. Let the Earth and Jupiter planetary orbits be assumed as coplanar and circular.

Hohmann Transfer from Earth to Jupiter:



(c) compute the total $|\Delta\bar{v}|$ and TOF (time of flight in years) for a Hohmann transfer from Earth to Jupiter. Ignore local fields. [Do not forget vector diagrams! Each planar $\Delta\bar{v}$ still requires $|\Delta\bar{v}|$, α.]
Comment on the required $|\Delta\bar{v}|$ — is this value large/small? Easily delivered by a launch vehicle? How does the TOF for the Hohmann transfer compare to the actual TOF?

Since we are assuming that the orbits are coplanar and circular, we can say that the radii are equal to the semi major axis of Earth and Jupiter about the Sun. Additionally we are disregarding the local fields of the Earth and Jupiter.

$$\mu = \mu_\odot = 132712440017.99 \ \ \mathrm{km}^3/\mathrm{s}^2$$

$$e_\oplus, e_{♃} = 0$$

$$r_1 = 149597898 \ \ km$$

$$r_2 = 778279959 \ \ km \ .$$

The velocity before entering the transfer orbit and after departing the transfer orbit are

$$-\frac{\mu}{2r_1} = \frac{v_1^2}{2} - \frac{\mu}{r_1} \Rightarrow v_1 = \sqrt{\frac{\mu}{r_1}} \ = \ 29.7847 \ \ km/s \ .$$

Similarly,

$$v_2 = \sqrt{\frac{\mu}{r_2}} = 13.0583 \ \ km/s \ .$$

The transfer orbit becomes,

$$a_T = 0.5(r_1 + r_2) = 4.6394e + 8 \ \ km$$

$$r_p = r_1, \ \ r_a = r_2 \ \ \Rightarrow \ \ e = \frac{r_a - r_p}{r_a + r_p} = \ 0.6775 \ .$$

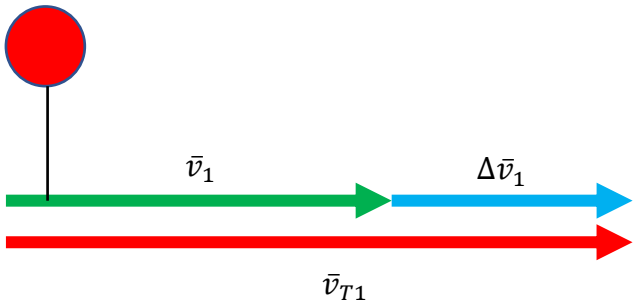
Then,

$$\frac{v_{T1}^2}{2} = \frac{\mu}{r_1} - \frac{\mu}{2a_T} \ \ \Rightarrow \ \ v_{T1} = \ 38.5772 \ \ km/s \ .$$

Similarly,

$$\frac{v_{T2}^2}{2} = \frac{\mu}{r_2} - \frac{\mu}{2a_T} \Rightarrow v_{T2} = 7.4152 \ \ km/s \ .$$

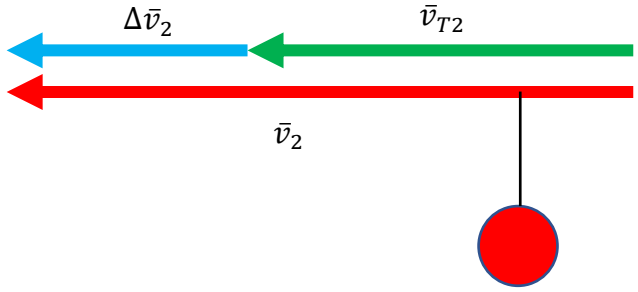
Thus, for the first maneuver point



$$\Delta\bar{v}_1 = \bar{v}_{T1} - \bar{v}_1 \;\Rightarrow\; \Delta v_1 = 8.7925\ km/s\ .$$

$$\alpha = 0°$$

At the second maneuver point



$$\Delta\bar{v}_2 = \bar{v}_2 - \bar{v}_{T2} \;\Rightarrow\; \Delta v_2 = 5.6432\ km/s\ .$$

$$\alpha = 0°$$

Thus, the total delta V is going to be

$$\Delta v_{tot} = \Delta v_1 + \Delta v_2 = 14.4357\ km/s\ .$$

The TOF is half the period of the transfer ellipse, and that is

$$0.5 * \mathcal{P} = \pi\sqrt{\frac{a_T^3}{\mu}} = 8.6176e+7\ s$$

which is

$$TOF = 2.7326\ years\ .$$

This total delta V value is very large and is NOT easily delivered by a launch vehicle. However, the *TOF* is 55.50% of the actual *TOF* which is significantly shorter.

(d) Assuming the Hohmann transfer path is employed, what is the required phase angle at Earth departure for arrival at Jupiter? For missions to Jupiter, what is the synodic period?

From notes J5, we can compute the phase angle, ϕ as

$$n_{♃} \times \text{TOF} = (180° - \phi)\frac{\pi}{180°} \quad \text{where} \quad n_{♃} = \sqrt{\frac{\mu}{r_2^3}} = 1.6778e-8 \ s^{-1} \ .$$

Thus,

$$\phi = 97.156° \ .$$

The mean motion of the Earth is

$$n_{\oplus} = \sqrt{\frac{\mu}{r_1^3}} = 1.991e-7 \ s^{-1} \ .$$

Then the synodic period becomes

$$\mathcal{SP} = \frac{2\pi}{n_{\oplus} - n_{♃}} = 3.4462e+7 \ s \ .$$

Which is

$$\mathcal{SP} = 1.0928 \ years \ .$$

(e) Jupiter's orbit is actually eccentric. Assume arrival at Jupiter perihelion and re-compute the Hohmann TOF and required $|\Delta\bar{v}|$. Compare these results to the Hohmann in (b). Does perihelion arrival same time or $|\Delta\bar{v}|$?

If we assume that Jupiter's orbit is an ellipse, we know that

$$a_{♃} = r_2 = 778279959 \ km$$

$$e_{♃} = 0.04853590$$

$$r_{p♃} = a_{♃}(1 - e_{♃}) = 7.4051e+8 \ km \ .$$

$$r_2 = r_{p♃}$$

The rest are all the same

$$e_{\oplus} = 0$$

$$\mu = \mu_{\odot} = 132712440017.99 \ \ \mathrm{km}^3/\mathrm{s}^2$$

$$r_1 = 149597898 \ \ km$$

The velocity before entering the transfer orbit and after departing the transfer orbit are

$$-\frac{\mu}{2r_1} = \frac{v_1^2}{2} - \frac{\mu}{r_1} \Rightarrow v_1 = \sqrt{\frac{\mu}{r_1}} \ = \ 29.7847 \ \ km/s \ .$$

Similarly,

$$v_2 = \sqrt{\frac{\mu}{r_2}} = 13.387 \ \ km/s \ .$$

The transfer orbit becomes,

$$a_T = 0.5(r_1 + r_2) = 4.4505e + 8 \ \ km$$

$$r_p = r_1, \ \ r_a = r_2 \ \ \Rightarrow \ \ e = \frac{r_a - r_p}{r_a + r_p} = \ 0.66386 \ .$$

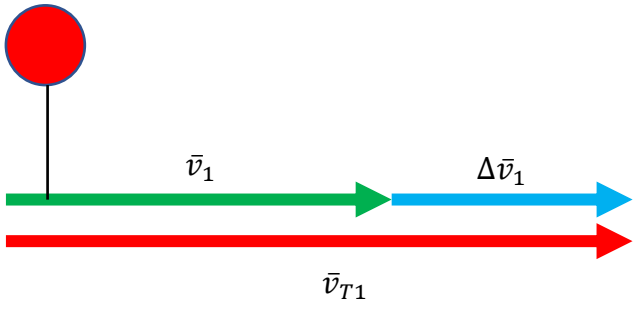
Then,

$$\frac{v_{T1}^2}{2} = \frac{\mu}{r_1} - \frac{\mu}{2a_T} \ \ \Rightarrow \ \ v_{T1} = \ 38.420 \ \ km/s \ .$$

Similarly,

$$\frac{v_{T2}^2}{2} = \frac{\mu}{r_2} - \frac{\mu}{2a_T} \Rightarrow v_{T2} = 7.7616 \ \ km/s \ .$$

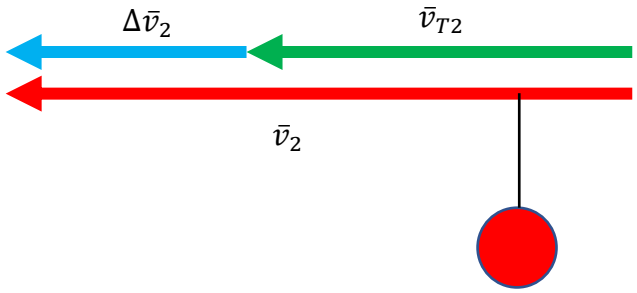
Thus, for the first maneuver point



$$\Delta\bar{v}_1 = \bar{v}_{T1} - \bar{v}_1 \ \ \Rightarrow \ \ \Delta v_1 \ = \ 8.6348 \ \ km/s \ .$$

$$\alpha = \ 0°$$

At the second maneuver point



$$\Delta\bar{v}_2 = \bar{v}_2 - \bar{v}_{T2} \;\Rightarrow\; \Delta v_2 = 5.6257 \;\; km/s \;.$$

$$\alpha = 0°$$

Thus, the total delta V is going to be

$$\Delta v_{tot} = \Delta v_1 + \Delta v_2 = 14.261 \;\; km/s \;.$$

The TOF is half the period of the transfer ellipse, and that is

$$0.5 * \mathcal{P} = \pi\sqrt{\frac{a_T^3}{\mu}} = 8.0967e+7 \;\; s$$

which is

$$TOF = 2.5675 \;\; years \;.$$

We can see that when we introduce the eccentricity of Jupiter, we have a smaller total delta V value and TOF. The perihelion time arrival is improved.

(f) Discuss: Do you think that arrival at perihelion affects the synodic period? Does it matter? [Return to the JPL small body database from PS1. View Jupiter's orbit. It may assist in the discussion.]

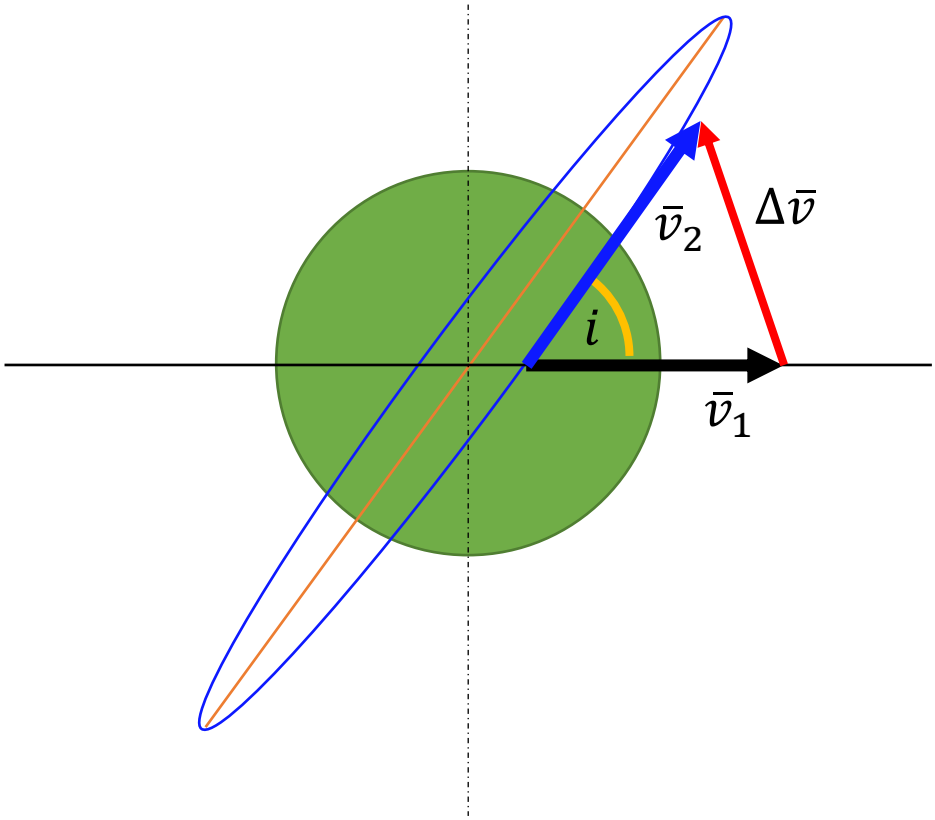
Even though we introduced the eccentricity of the Jupiter orbit, the assumption of no local gravity fields holds true. The synodic period, as we can observe in our calculations, is a function of the mean motion of the Earth and Jupiter about the gravity field of the Sun. Thus, changing the eccentricity of Jupiter does not change the synodic period.

Since the eccentricity of Jupiter is close to 0, with the coplanar assumption, the affect of it is trivial. However, we can examine that the total delta V and the *TOF* values are reduced. Hence, it is reasonable to consider the eccentricity of Jupiter regardless of its near-circular value.

Problem 3: Consider transfers about the Earth that include a plane change. Assume that a vehicle is in a circular Earth equatorial orbit of altitude 200 km and the vehicle is to be shifted to an orbit with the same altitude but an inclination of 57° (consistent with the Kodiak launch site in Alaska).

(a) Consider a single maneuver to accomplish the plane change. Include the vector diagram and determine the appropriate values of $|\Delta\bar{v}|$, α, β.

The diagram of the maneuver is the following.



From the old orbit we know that

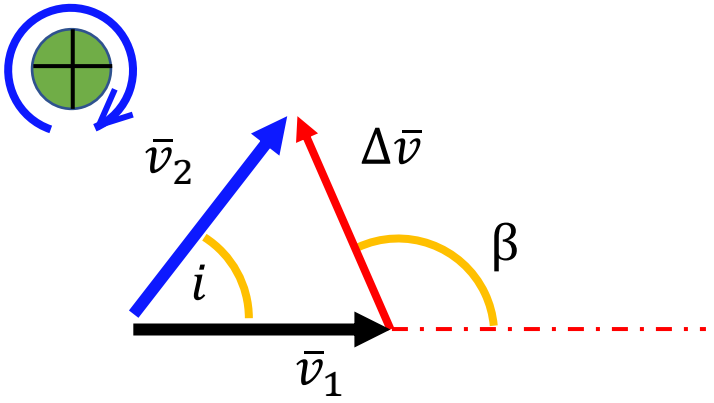
$$r_1 = 200 + R_\oplus = 6578.1 \ km$$

$$v_1 = \sqrt{\frac{\mu_\oplus}{r_1}} = 7.7843 \ km/s \ .$$

Since the new orbit has the same altitude

$$r_2 = 6578.1 \ km$$

$$v_2 = 7.7843 \ km/s \ .$$



From this vector diagram we can see that

$$\bar{v}_1 = v_1\hat{\theta}_1$$

$$\beta = 180° - \frac{180° - 57°}{2} = 118.5°$$

Since the projection of $\bar{v}_2$ on the original orbit plane has 0 angle with $\bar{v}_1$, we know that

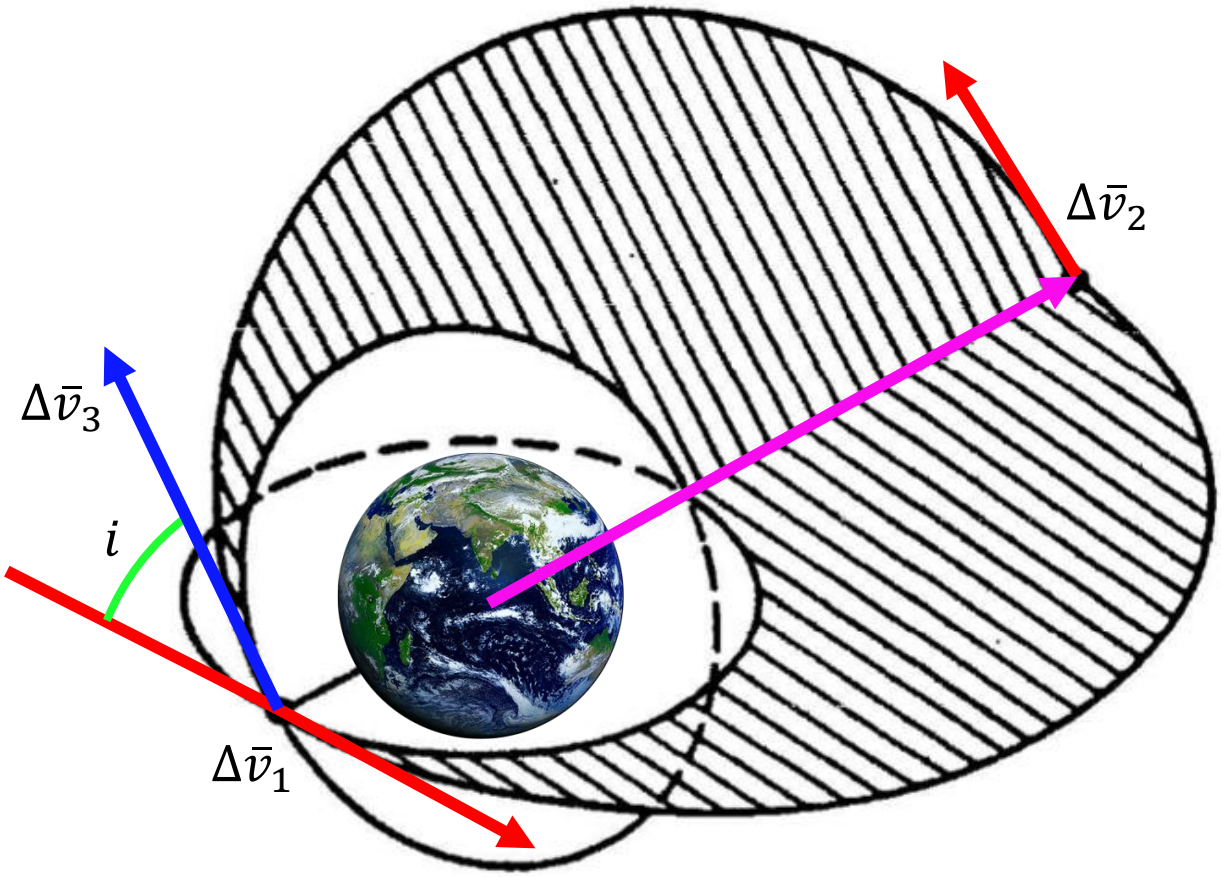
$$\alpha = 0° \ .$$

And from trigonometric properties

$$|\Delta\bar{v}|^2 = v_1^2 + v_2^2 - 2v_1v_2cosi \ \Rightarrow \ |\Delta\bar{v}| = 7.4287 \ km/s \ .$$

(b) Use a bi-elliptic transfer and an intermediate radius of $55R_{\oplus}$. The cost includes all the required maneuvers; include vector diagrams and appropriate values of $|\Delta\bar{v}|$, α, β for each maneuver. Of course, the plane change occurs at the second maneuver.

The pink line in the diagram below indicates the intermediate radius. This diagram is to only vision our transfer and the inclination should be the opposite direction.



First let us define the transfer orbit

$$a_T = 0.5 * (r_1 + r_i) = 1.7869e + 5km \qquad \because r_i = 55R_{\oplus}$$

$$r_{pT} = r_1 = 6578.1 \ km$$

$$e = 1 - \frac{r_{pT}}{a_T} = 0.96319\,.$$

Then the velocities for the two elliptic transfers are going to be,

First ellipse:

$$v_{T11} = \sqrt{\mu_\oplus \left(\frac{2}{r_1} - \frac{1}{a_T}\right)} = 10.907 \ km/s$$

$$v_{T12} = \sqrt{\mu_\oplus \left(\frac{2}{r_i} - \frac{1}{a_T}\right)} = 0.20452 \ km/s$$

Second ellipse:

$$v_{T21} = \sqrt{\mu_\oplus \left(\frac{2}{r_i} - \frac{1}{a_T}\right)} = 0.20452 \ km/s$$

$$v_{T22} = \sqrt{\mu_\oplus \left(\frac{2}{r_1} - \frac{1}{a_T}\right)} = 10.907 \ km/s$$

From the old orbit and new orbit we know that

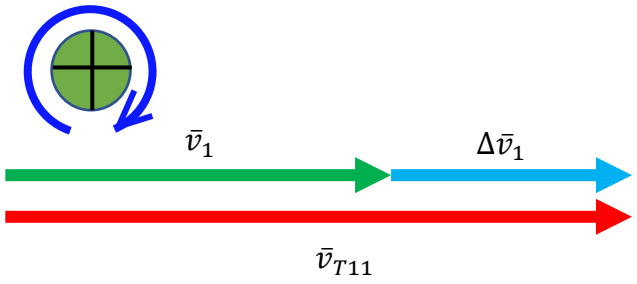
$$r_1 = 200 + R_\oplus = 6578.1 \ km$$

$$v_1 = \sqrt{\frac{\mu_\oplus}{r_1}} = 7.7843 \ km/s \ .$$

$$r_2 = 6578.1 \ km$$

$$v_2 = 7.7843 \ km/s \ .$$

First maneuver:

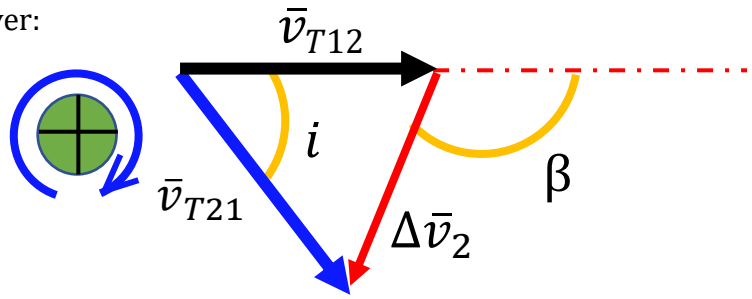


$$\Delta\bar{v}_1 = \bar{v}_{T11} - \bar{v}_1 \ \Rightarrow \ |\Delta\bar{v}_1| = 10.907 - 7.7843 = 3.1226 km/s.$$

$$\alpha_1 = 0°$$

$$\beta_1 = 0°$$

Second Maneuver:



From this vector diagram we can see that

$$\beta_2 = -\left(180° - \frac{180° - 57°}{2}\right) = -118.5°$$

Since the projection of $\bar{v}_2$ on the original orbit plane has 0 angle with $\bar{v}_1$, we know that

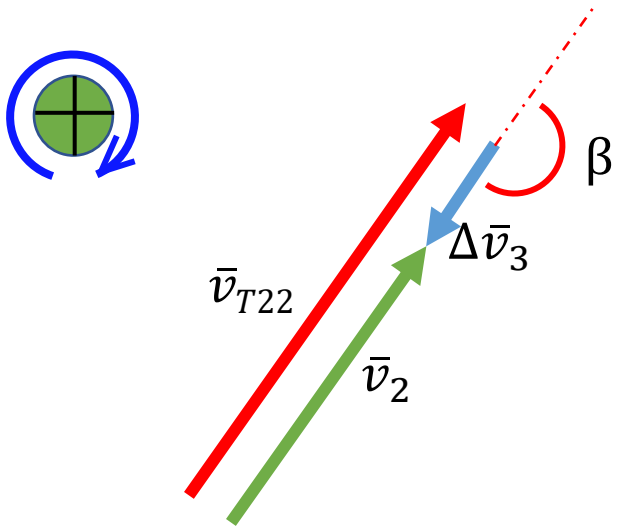
$$\alpha_2 = 0°.$$

And from trigonometric properties

$$|\Delta\bar{v}_2| = v_{T12}^2 + v_{T21}^2 - 2v_{T12}v_{T21}cosi \quad \Rightarrow \quad |\Delta\bar{v}_2| = 0.19518 \ \ km/s \ .$$

$$\Delta\bar{v}_2 = |\Delta\bar{v}_2|\left(cos\beta\hat{V} + sin\beta\hat{N}\right) = -0.093132\hat{V} - 0.17153\hat{N}$$

Third Maneuver:



$$\Delta\bar{v}_3 = \bar{v}_2 - \bar{v}_{T22} \quad \Rightarrow \quad |\Delta\bar{v}_3| = |7.7843 - 10.907| = 3.1226 \ \ km/s \ .$$

$$\alpha_3 = 0°$$

$$\beta_3 = 180°$$

(c) What is the total cost and the TOF for the bi-elliptic option? What is the $|\Delta\bar{v}|$ cost savings? Time penalty?

The total delta V is

$$|\Delta\bar{v}|_{TOT} = |\Delta\bar{v}_1| + |\Delta\bar{v}_2| + |\Delta\bar{v}_3| = 6.4403 \ km/s \ .$$

The TOF is

$$TOF = 2\pi\sqrt{\frac{a_T^3}{\mu}} = 7.5172e+5 \ s \ = \ 8.7004 \ days \ .$$

The cost saving of delta V is

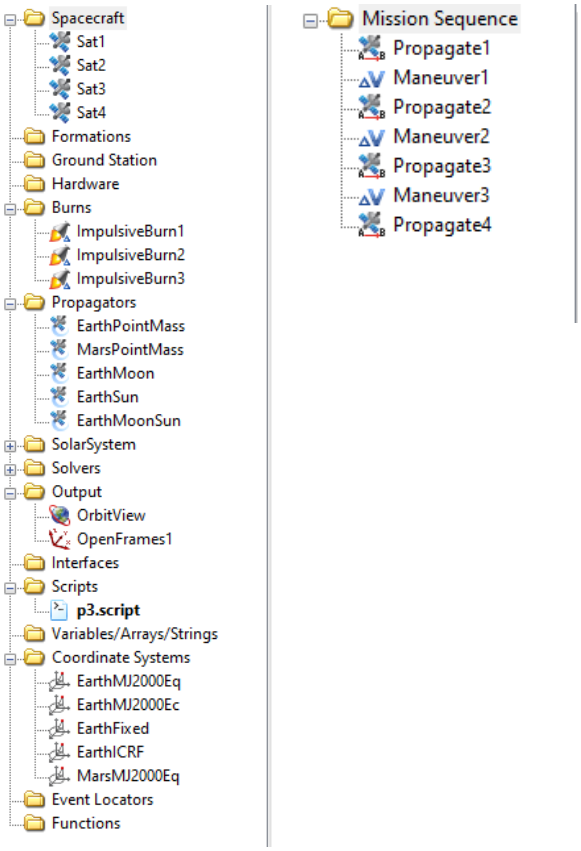
$$7.4287 \ - \ 6.4403 \ = \ 0.98836 \ km/s \ .$$

Since for the first single maneuver discussed in part (a) the maneuver can be done anywhere in the circular orbit, the time penalty for using the bi-elliptic option is the entire TOF. Thus,

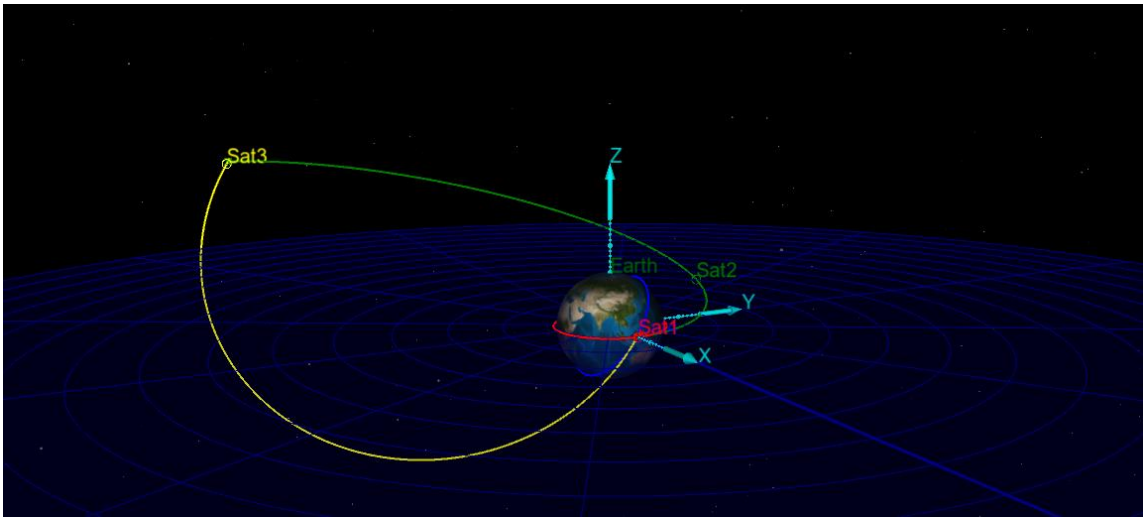
$$Time \ Penalty \ = \ TOF \ = \ 8.7004 \ days \ .$$

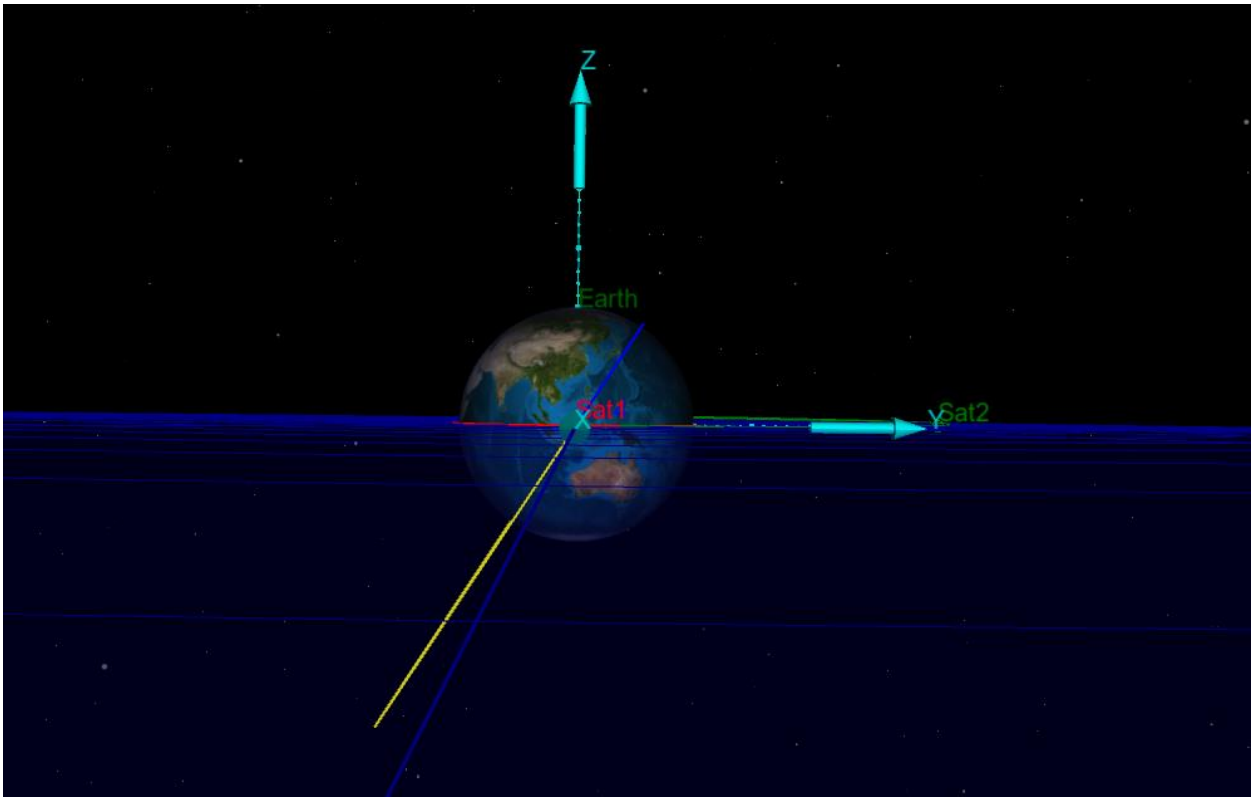
(d) Reproduce the bi-elliptic option in GMAT and confirm your results. Add the Moon's orbit to the 3D window. How close is the intermediate orbit to the lunar radius? Is this close enough to introduce a concern about lunar gravity?

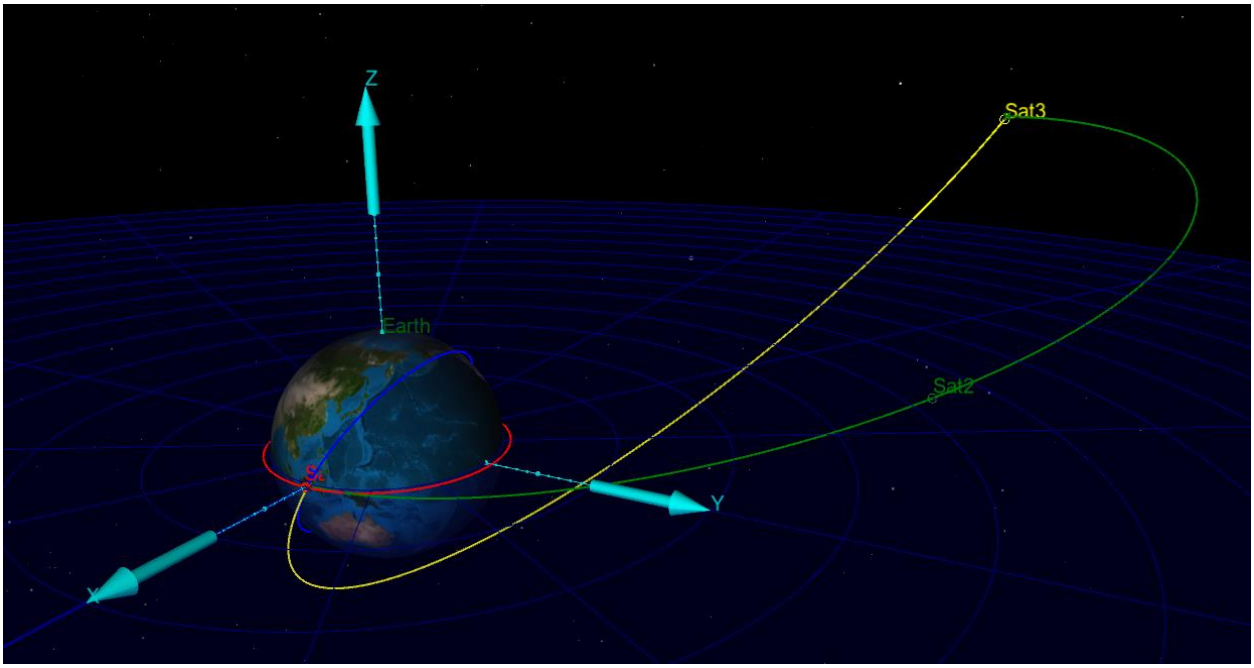
In GMAT, set the propagators and burns



The GMAT plots are the following,




Z
Earth
Sat1
X
Y
Sat2


Z
Sat3
Earth
Sat2
Y

The distance from the Earth to the Moon is approximately 384400 km. Thus, the distance from the Moon to the satellite is

$$384400 - 350800 = 33603\ km$$

We consider the equation of motion for the Earth-spacecraft-Moon linear (the three bodies lined up) model relative to the Earth when the satellite is at the apoapsis of the transfer ellipse.

$$\ddot{\bar{r}}_{\oplus s/c} + G\frac{(m_\oplus + m_{s/c})}{r_{\oplus s/c}^3}\bar{r}_{\oplus s/c} = Gm_\leftmoon\left(\frac{\bar{r}_{s/c\leftmoon}}{r_{s/c\leftmoon}^3} - \frac{\bar{r}_{\oplus\leftmoon}}{r_{\oplus\leftmoon}^3}\right)$$

$$dominant\ term := G\frac{(m_\oplus + m_{s/c})}{r_{\oplus s/c}^3}\bar{r}_{\oplus s/c}^3 \approx G\frac{(m_\oplus)}{r_{\oplus s/c}^3}\bar{r}_{\oplus s/c} = 3.2391e-6\ \ km/s^2$$

$$direct\ perturbing\ term := Gm_\leftmoon\left(\frac{\bar{r}_{s/c\leftmoon}}{r_{s/c\leftmoon}^3}\right) = 4.3421e-6\ \ km/s$$

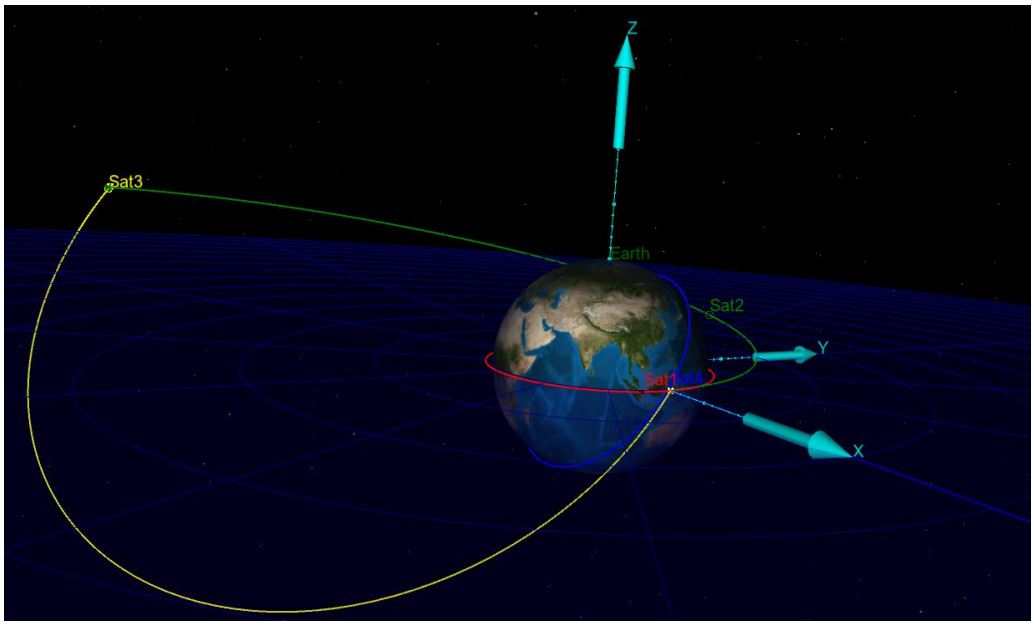
$$indirect\ pertubing\ term := Gm_\leftmoon\left(\frac{\bar{r}_{\oplus\leftmoon}}{r_{\oplus\leftmoon}^3}\right) = 3.3180e-8\ \ km/s$$

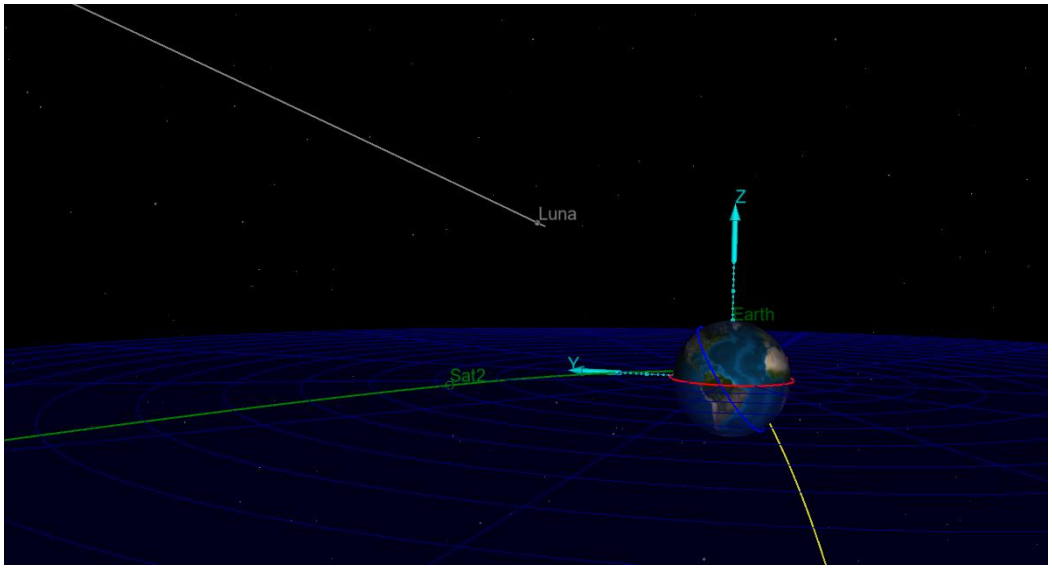
$$net\ perturbing\ term := Gm_\leftmoon\left(\frac{\bar{r}_{s/c\leftmoon}}{r_{s/c\leftmoon}^3} - \frac{\bar{r}_{\oplus\leftmoon}}{r_{\oplus\leftmoon}^3}\right) = 4.3089e-6\ \ km/s$$

Since we can see that the net perturbing term is larger than the dominant term when the satellite approaches the apoapsis of the transfer ellipse, we can say that the lunar gravity is not ignorable. Thus, the satellite trajectory has its path close enough to the Moon to introduce lunar gravity.

(e) After thinking about the potential impact of lunar gravity in (d), add the Earth+Moon propagator. That is, use the same three maneuvers but propagate forward with the E+M propagator. Is the result impacted significantly by the lunar gravity (for an intermediate radius of $55R_{\oplus}$)? Does the date that you use to initiate the propagation matter? Why?

Changing the propagator to "EarthPointMass" to "Earth+Moon" for the transfer ellipse in GMAT, we see the following plots

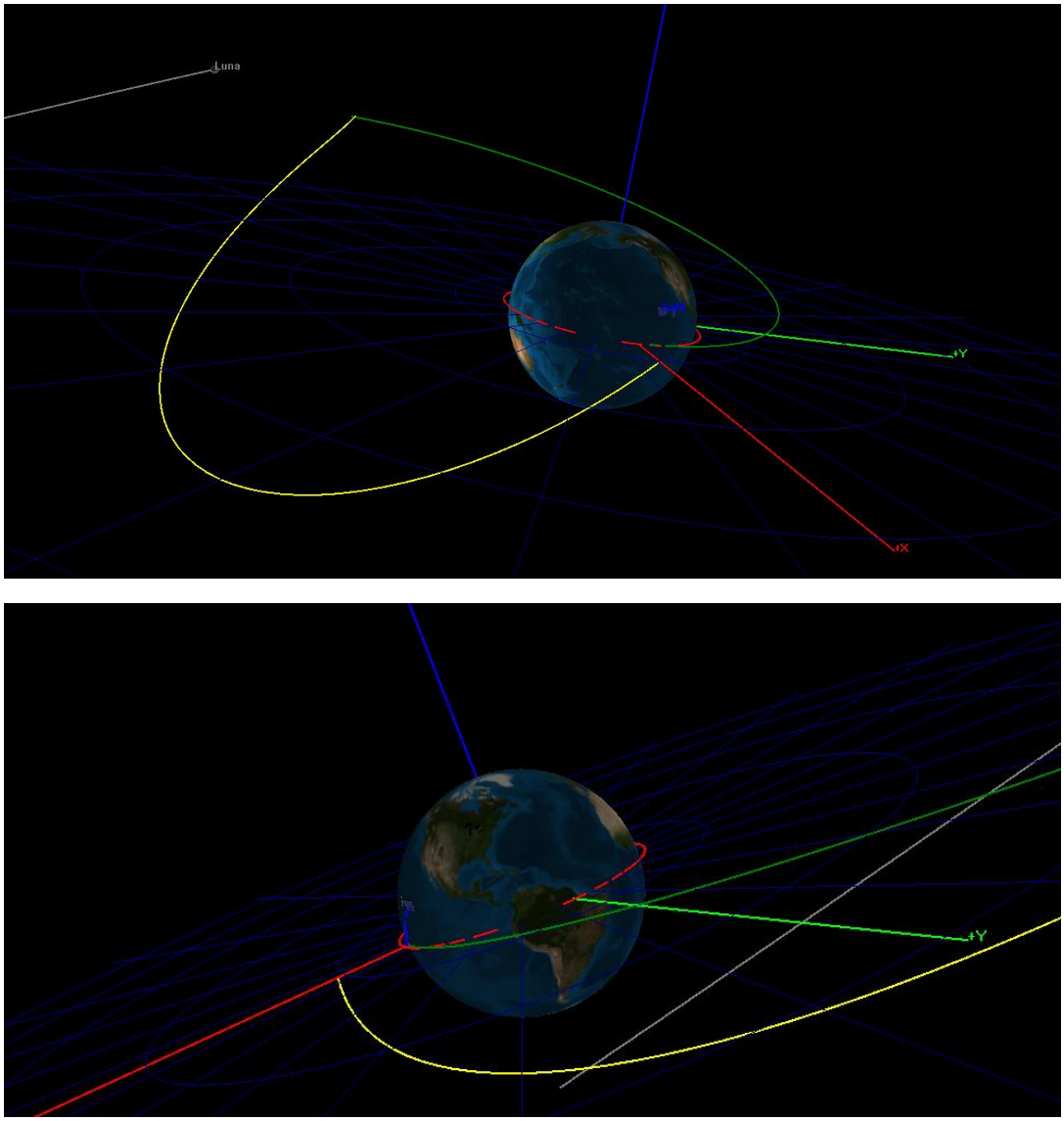




As we can see this did not change the simulation at all. This is because the Moon is on the opposite side relative to the transfer ellipse on date of October 30, 2020 which can be seen on the second figure. The positions of the bodies are not corresponding to the Earth-

spacecraft-moon scenario we presumed in our calculations in part (d). Thus, for the lunar gravity to show a larger effect on the orbit, we have to select a date in which the Moon becomes close to the spacecraft. This means that the date that you initiate the propagation matters.

To prove this, we select a date of "12 Nov 2020" where the Moon gets relatively close to the transfer ellipse.



With this date, you can see that after the second maneuver there is no third maneuver done due to the fact that the periapsis of the second transfer ellipse has been altered by the lunar gravity. This makes it viable to consider the lunar gravity because it can have a significant effect on the bi-ellipse transfer depending on the date.

Appendix

MATLAB CODE

Problem #1

```
% AAE 532 HW 7 Problem 1
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps7';
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
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)
% Set the given constants
a1 = 6*mars.mer;  % semi major axis [km]
AOP1 = 30;  %  argument of periapsis [deg]
e1 = 0.5;  % eccentricity
TA1 = 150;  % true anomaly [deg]
RAAN1 = 45;  % right ascension of the ascending node [deg]
i1 = 30;  % inclination [deg]
AOL1 = TA1 + AOP1;  % argument of latitude [deg]
mu = mars.gp;  % gravitational parameter

% Other orbital parameters at the maneuver point
p1 = a1*(1 - e1^2)  % semi latus rectum
r1 = p1 / (1 + e1 * cosd(TA1))  % radial distance
v1 = vis_viva(r1, a1, mu)  % velocity
FPA1 = acosd(sqrt(mu * p1) / r1 / v1)  % flight path angle
rp1 = a1 * (1 - e1);
rp1vec = orbit_frame_transform(0+AOP1, i1, RAAN1, [rp1, 0, 0], "orbital", "deg");

% Transform the delta v from the inertial frame to the orbital frame
dv_i = [0.1, 0.25, 0.35];  % delta v in the inertial frame
dvvec = orbit_frame_transform(AOL1, i1, RAAN1, dv_i, "inertial", "deg")
dvvec.i = dv_i;
temp = abs(dvvec.o(3))/norm(dvvec.o) * 100

dv_rtheta = dvvec.o(1:2)
v1vec.o = v1 * [sind(FPA1), cosd(FPA1), 0]
dv = norm(dvvec.o);
```

```
alpha = acos_dbval(dot(v1vec.o(1:2), dv_rtheta) / v1 / norm(dv_rtheta), "deg")

beta = asin_dbval(dvvec.o(3)/dv, "deg")

% determine phi
phi1 = acos_dbval(dvvec.o(2)/dv/cosd(beta), "deg")
phi2 = asin_dbval(dvvec.o(1)/dv/cosd(beta), "deg")
phi = intersect(round(phi1, 2), round(phi2, 2))

alpha = alpha(alpha < 0)
dvvec.vnb = dv * [cosd(beta)*cosd(alpha), cosd(beta)*sind(alpha), sind(beta)]

% (b)

% Transforming the position vectors into inertial frames and getting the
% vectors after the maneuver
r1vec.o = r1 * [1, 0, 0];
r1vec = catstruct(orbit_frame_transform(AOL1, i1, RAAN1, r1vec.o, "orbital",
"deg"), r1vec)
v1vec = catstruct(orbit_frame_transform(AOL1, i1, RAAN1, v1vec.o, "orbital",
"deg"), v1vec)

h1vec = cross(r1vec.i, v1vec.i)

v2vec.i = v1vec.i + dvvec.i

% (c)
r2vec = r1vec;
r2 = norm(r1vec.i)
v2 = norm(v2vec.i)
r2hat = r2vec.i / r2

a2 = -mu / (v2^2/2 - mu/r2)
h2vec = cross(r2vec.i, v2vec.i)
h2 = norm(h2vec)
e2 = sqrt(1 - h2^2 / mu / a2)
h2hat = h2vec / h2
theta2hat = cross(h2hat, r2hat)

i2 = acosd(h2hat(3))
RAAN21 = asin_dbval(h2hat(1)/sind(i2), "deg")
RAAN22 = acos_dbval(-h2hat(2)/sind(i2), "deg")
RAAN2 = intersect(round(RAAN21,5), round(RAAN22,5))
AOL21 = asin_dbval(r2hat(3)/sind(i2), "deg")
AOL22 = acos_dbval(theta2hat(3)/sind(i2), "deg")
AOL2 = intersect(round(AOL21,5),round(AOL22,5))
r2dot = dot(v2vec.i, r2hat)
p2 = h2^2/mu;
TA2 = acos_dbval(1/e2 * (p2/r2 - 1), "deg")
TA2 = TA2(TA2 > 0)
AOP2 = AOL2 - TA2
```

```
rp2 = a2 * (1 - e2);
rp2vec = orbit_frame_transform(0+AOP2, i2, RAAN2, [rp2, 0, 0], "orbital", "deg");

% (d)
% Plotting

% old orbit
TAs = 0:0.01:360;
AOLs = TAs + AOP1;
temp = p1 ./ (1 + e1*cosd(TAs));
temp = [temp.', zeros([numel(temp), 2])];
R_old.o = temp;
R_old = catstruct(orbit_frame_transform(AOLs, i1, RAAN1, R_old.o, "orbital",
"deg"), R_old);
% find AN and DN
temp = find_AN_DN(R_old.i);
nodes_old = [temp.AN.values; temp.DN.values];
% New orbit
AOLs = TAs + AOP2;
temp = p2 ./ (1 + e2*cosd(TAs));
temp = [temp.', zeros([numel(temp), 2])];
R_new.o = temp;
R_new = catstruct(orbit_frame_transform(AOLs, i2, RAAN2, R_new.o, "orbital",
"deg"), R_new);
% find AN and DN
temp = find_AN_DN(R_new.i);
nodes_new = [temp.AN.values; temp.DN.values];

fig1 = figure("Renderer","painters","Position",[10, 10, 900, 700]); ax = gca;
plot3(R_old.i(:,1),R_old.i(:,2),R_old.i(:,3))
hold on; grid on; grid minor; box on; axis equal;
plot3(R_new.i(:,1),R_new.i(:,2),R_new.i(:,3))

% Mars equator
xmin = min([min(R_old.i(:,1)), min(R_new.i(:,1))])
xmax = max([max(R_old.i(:,1)), max(R_new.i(:,1))])
ymin = min([min(R_old.i(:,2)), min(R_new.i(:,2))])
ymax = max([max(R_old.i(:,2)), max(R_new.i(:,2))])
[x_eq, y_eq] = meshgrid(xmin:100:xmax,ymin:100:ymax); % Generate x and y data
z_eq = zeros(size(x_eq)); % Generate z data
s_eq = surf(x_eq, y_eq, z_eq, "FaceAlpha",0.3); % Plot the surface
s_eq.EdgeColor = 'none';

% AN and DN
plot3(nodes_old(:,1), nodes_old(:,2), nodes_old(:,3), '-')
plot3(nodes_new(:,1), nodes_new(:,2), nodes_new(:,3), '-')

% Angular Momentum vector
plot3([0;h1vec(1)],[0;h1vec(2)],[0;h1vec(3)], '-')
plot3([0;h2vec(1)],[0;h2vec(2)],[0;h2vec(3)], '-')

% Periapsis Vectors
```

```
plot3([0;rp1vec.i(1)],[0;rp1vec.i(2)],[0;rp1vec.i(3)], '-')
plot3([0;rp2vec.i(1)],[0;rp2vec.i(2)],[0;rp2vec.i(3)], '-')

% Maneuver location
plot3(r1vec.i(1), r1vec.i(2), r1vec.i(3), '.r', "MarkerSize", 22)

hold off
title("Problem1: Old and New Orbit 3D Plot - T. Koike")
xlabel("$\hat{x}$")
ylabel("$\hat{y}$")
zlabel("$\hat{z}$")
saveas(fig1, fullfile(fdir, "p1_view1.png"));

fig2 = figure("Renderer","painters","Position",[10, 10, 900, 700]); ax = gca;
plot3(R_old.i(:,1),R_old.i(:,2),R_old.i(:,3))
hold on; grid on; grid minor; box on; axis equal;
plot3(R_new.i(:,1),R_new.i(:,2),R_new.i(:,3))

% Mars equator
xmin = min([min(R_old.i(:,1)), min(R_new.i(:,1))])
xmax = max([max(R_old.i(:,1)), max(R_new.i(:,1))])
ymin = min([min(R_old.i(:,2)), min(R_new.i(:,2))])
ymax = max([max(R_old.i(:,2)), max(R_new.i(:,2))])
[x_eq, y_eq] = meshgrid(xmin:100:xmax,ymin:100:ymax); % Generate x and y data
z_eq = zeros(size(x_eq)); % Generate z data
s_eq = surf(x_eq, y_eq, z_eq, "FaceAlpha",0.3); % Plot the surface
s_eq.EdgeColor = 'none';

% AN and DN
plot3(nodes_old(:,1), nodes_old(:,2), nodes_old(:,3), '-')
plot3(nodes_new(:,1), nodes_new(:,2), nodes_new(:,3), '-')

% Angular Momentum vector
plot3([0;h1vec(1)],[0;h1vec(2)],[0;h1vec(3)], '-')
plot3([0;h2vec(1)],[0;h2vec(2)],[0;h2vec(3)], '-')

% Periapsis Vectors
plot3([0;rp1vec.i(1)],[0;rp1vec.i(2)],[0;rp1vec.i(3)], '-')
plot3([0;rp2vec.i(1)],[0;rp2vec.i(2)],[0;rp2vec.i(3)], '-')

% Maneuver location
plot3(r1vec.i(1), r1vec.i(2), r1vec.i(3), '.r', "MarkerSize", 22)

hold off
title("Problem1: Old and New Orbit 3D Plot in Better View Angle - T. Koike")
xlabel("$\hat{x}$")
ylabel("$\hat{y}$")
zlabel("$\hat{z}$")
view([-24 12])
saveas(fig2, fullfile(fdir, "p1_view2.png"));

fig3 = figure("Renderer","painters","Position",[10, 10, 900, 700]); ax = gca;
```

```
plot3(R_old.i(:,1),R_old.i(:,2),R_old.i(:,3))
hold on; grid on; grid minor; box on; axis equal;
plot3(R_new.i(:,1),R_new.i(:,2),R_new.i(:,3))

% Mars equator
xmin = min([min(R_old.i(:,1)), min(R_new.i(:,1))])
xmax = max([max(R_old.i(:,1)), max(R_new.i(:,1))])
ymin = min([min(R_old.i(:,2)), min(R_new.i(:,2))])
ymax = max([max(R_old.i(:,2)), max(R_new.i(:,2))])
[x_eq, y_eq] = meshgrid(xmin:100:xmax,ymin:100:ymax); % Generate x and y data
z_eq = zeros(size(x_eq)); % Generate z data
s_eq = surf(x_eq, y_eq, z_eq, "FaceAlpha",0.3); % Plot the surface
s_eq.EdgeColor = 'none';

% AN and DN
plot3(nodes_old(:,1), nodes_old(:,2), nodes_old(:,3), '-')
plot3(nodes_new(:,1), nodes_new(:,2), nodes_new(:,3), '-')

% Angular Momentum vector
plot3([0;h1vec(1)],[0;h1vec(2)],[0;h1vec(3)], '-')
plot3([0;h2vec(1)],[0;h2vec(2)],[0;h2vec(3)], '-')

% Periapsis Vectors
plot3([0;rp1vec.i(1)],[0;rp1vec.i(2)],[0;rp1vec.i(3)], '-')
plot3([0;rp2vec.i(1)],[0;rp2vec.i(2)],[0;rp2vec.i(3)], '-')

% Maneuver location
plot3(r1vec.i(1), r1vec.i(2), r1vec.i(3), '.r', "MarkerSize", 22)

hold off
title("Problem1: Old and New Orbit 3D Plot in Side View - T. Koike")
xlabel("$\hat{x}$")
ylabel("$\hat{y}$")
zlabel("$\hat{z}$")
view([-90 0])

saveas(fig3, fullfile(fdir, "p1_view3.png"));
```

Problem #2

```
% AAE 532 HW 7 Problem 2
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps7';
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
sun = planet_consts.sun;  % structure of sun
earth = planet_consts.earth;  % structure of earth
mars = planet_consts.mars;  % structure of mars
jupiter = planet_consts.jupiter;  % structure of jupiter
G = 6.6743015e-20;  % Gravitational constant [km^3/kg/s^2]

% (c)
mu = sun.gp;
r1 = earth.smao
r2 = jupiter.smao
v1 = sqrt(mu / r1)
v2 = sqrt(mu / r2)

aT = mean([r1, r2])
e = (r2 - r1) / (r2 + r1)
vT1 = vis_viva(r1, aT, mu)
vT2 = vis_viva(r2, aT, mu)
dv1 = vT1 - v1
dv2 = v2 - vT2
dvtot = dv1 + dv2
TOF = pi * sqrt(aT^3/mu)
TOF_years = TOF / 60 / 60 / 24 / 365

% (d)
MM_j = sqrt(mu / r2^3);
phi = rad2deg(pi - MM_j * TOF)
MM_e = sqrt(mu / r1^3)
SP = 2*pi / (MM_e - MM_j)
SP_years = SP / 60 / 60 / 24 / 365

% (e)
aJ = r2
eJ = jupiter.eo
rpJ = aJ*(1 - eJ)
r2 = rpJ
v1 = sqrt(mu / r1)
v2 = sqrt(mu / r2)

aT = mean([r1, r2])
e = (r2 - r1) / (r2 + r1)
vT1 = vis_viva(r1, aT, mu)
vT2 = vis_viva(r2, aT, mu)
dv1 = vT1 - v1
dv2 = v2 - vT2
dvtot = dv1 + dv2
TOF = pi * sqrt(aT^3/mu)
TOF_years = TOF / 60 / 60 / 24 / 365
```

Problem #3

```
% AAE 532 HW 7 Problem 3
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps7';
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
sun = planet_consts.sun;  % structure of sun
earth = planet_consts.earth;  % structure of earth
mars = planet_consts.mars;  % structure of mars
moon = planet_consts.moon;
G = 6.6743015e-20;  % Gravitational constant [km^3/kg/s^2]

% (a)
mu = earth.gp;
i = 57;
r1 = 200 + earth.mer  % altitude
v1 = sqrt(mu / r1)
r2 = r1; v2 = v1;
beta = 180 - (180 - i) / 2
alpha = 0
dv = sqrt(v1^2 + v2^2 - 2*v1*v2*cosd(i))

% (b)
Re = earth.mer;
ri = 55 * Re
rpT =  r1;
aT = 0.5*(r1 + ri)
e = 1 - rpT/aT
vT11 = vis_viva(r1, aT, mu)
vT12 = vis_viva(ri, aT, mu)
vT21 = vT12;
vT22 = vT11;
v2 = v1;

dv1 = vT11 - v1
dv2 = sqrt(vT12^2 + vT21^2 - 2*vT12*vT21*cosd(i))
dv2vec = dv2 * [-cosd(beta), sind(beta)]
dv3 = abs(v2 - vT22)

% (c)
dv_tot = dv1 + dv2 + dv3
TOF = 2*pi*sqrt(aT^3 / mu)
TOF_days = TOF / 60 / 60 / 24
```

```
dv_save = dv - dv_tot

% (d)
d = 384400;
DT = earth.gp/ri^2
DPT = moon.gp / (d-ri)^2
IDPT = moon.gp / (d)^2
NPT = DPT - IDPT
```

Supplemental MATLAB Codes

```
%% Table of Constants

function planets = setup_planetary_constants()

    %{
      arp  : Axial Rotational Period (Rev / Day)
      mer  : Mean Equatorial Radius (km)
      gp   : Gravitational Parameter, mu (km^3 / s^2)
      smao : Semi-Major Axis of Orbit (km)
      op   : Orbital Period (s)
      eo   : Eccentricity of Orbit
      ioe  : Inclination of Orbit to Ecliptic (deg)
    %}

    % Sun
    sun.arp = 0.0394011;
    sun.mer = 695990;
    sun.gp  = 132712440017.99;
    sun.smao = NaN;
    sun.op   = NaN;
    sun.eo   = NaN;
    sun.ioe  = NaN;

    % Moon
    moon.arp  = 0.0366004;
    moon.mer  = 1738.2;
    moon.gp   = 4902.8005821478;
    moon.smao = 384400;
    moon.op   = 2360592;
    moon.eo   = 0.0554;
    moon.ioe  = 5.16;

    % Mercury
    mercury.arp  = 0.0170514;
    mercury.mer  = 2439.7;
    mercury.gp   = 22032.080486418;
    mercury.smao = 57909101;
    mercury.op   = 7600537;
    mercury.eo   = 0.20563661;
    mercury.ioe  = 7.00497902;

    % Venus
```

```
venus.arp  = 0.0041149;  % retrograde
venus.mer  = 6051.9;
venus.gp   = 324858.59882646;
venus.smao = 108207284;
venus.op   = 19413722;
venus.eo   = 0.00676399;
venus.ioe  = 3.39465605;

% Earth
earth.arp  = 1.0027378;
earth.mer  = 6378.1363;
earth.gp   = 398600.4415;
earth.smao = 149597898;
earth.op   = 31558205;
earth.eo   = 0.01673163;
earth.ioe  = 0.00001531;

% Mars
mars.arp  = 0.9747000;
mars.mer  = 3397;
mars.gp   = 42828.314258067;
mars.smao = 227944135;
mars.op   = 59356281;
mars.eo   = 0.09336511;
mars.ioe  = 1.84969142;

% Jupiter
jupiter.arp  = 2.4181573;
jupiter.mer  = 71492;
jupiter.gp   = 126712767.8578;
jupiter.smao = 778279959;
jupiter.op   = 374479305;
jupiter.eo   = 0.04853590;
jupiter.ioe  = 1.30439695;

% Saturn
saturn.arp  = 2.2522053;
saturn.mer  = 60268;
saturn.gp   = 37940626.061137;
saturn.smao = 1427387908;
saturn.op   = 930115906;
saturn.eo   = 0.05550825;
saturn.ioe  = 2.48599187;

% Uranus
uranus.arp  = 1.3921114;  % retrograde
uranus.mer  = 25559;
uranus.gp   = 5794549.0070719;
uranus.smao = 2870480873;
uranus.op   = 2652503938;
uranus.eo   = 0.04685740;
uranus.ioe  = 0.77263783;

% Neptune
neptune.arp  = 1.4897579;
neptune.mer  = 25269;
```

```
    neptune.gp   = 6836534.0638793;
    neptune.smao = 4498337290;
    neptune.op   = 5203578080;
    neptune.eo   = 0.00895439;
    neptune.ioe  = 1.77004347;

    % Pluto
    pluto.arp  = -0.1565620;  % retrograde
    pluto.mer  = 1162;
    pluto.gp   = 981.600887707;
    pluto.smao = 5907150229;
    pluto.op   = 7830528509;
    pluto.eo   = 0.24885238;
    pluto.ioe  = 17.14001206;

    % Return
    planets.sun = sun;
    planets.moon = moon;
    planets.mercury = mercury;
    planets.venus = venus;
    planets.earth = earth;
    planets.mars = mars;
    planets.jupiter = jupiter;
    planets.saturn = saturn;
    planets.uranus = uranus;
    planets.neptune = neptune;
    planets.pluto = pluto;
end
```

```
function v = vis_viva(r, a, mu)
 %{
       NAME:    VIS_VIVA
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) r:  POSITION (LENGTH) ON ORBIT
                (2) a:  SEMI MAJOR AXIS
                (3) mu: GRAVITATIONAL PARAMETER
       OUTPUTS: (1) v:  VELOCITY AT THE POSITION
       DESCRIPTION: CALCULATES THE VELOCITY FOR A CERTAIN POSITION ON A
       CONIC ORBIT.
 %}

    v = sqrt(mu * (2/r - 1/a));
end
```

```
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
```