College of Engineering
School of Aeronautics and Astronautics

AAE 532

Orbital Mechanics

PS 10

Transfer Orbit Analysis with Lambert's Equation

Author:
Tomoki Koike

Supervisor:
K. C. Howell

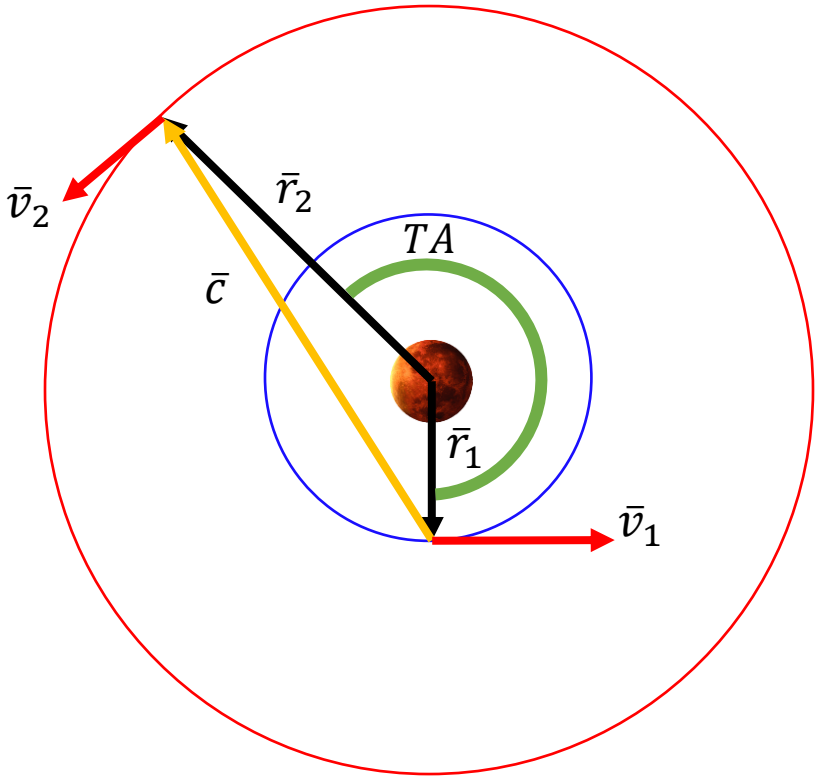
December 10th, 2020 Friday
Purdue University
West Lafayette, Indiana

Problem 1:

Recall the example problem that was discussed in class concerning a small robotic explorer sent to the Martian system to observe and characterize the two moons – Phobos and Deimos. The Martian moons Phobos and Deimos are assumed to be in circular and coplanar about Mars, with a radius equal to the semi-major axis listed in the Table of Constants for moons and dwarfs under Supplementary Documents on Brightspace. Let's again assume that the spacecraft has completed its observations in the orbit of Phobos and must transfer to the orbit of Deimos. But, now an option for a transfer with different characteristics is sought. [Assume that it is reasonable to assume a relative two-body problem and consider only the gravity of Mars.]

(a) In the class example, recall that the planned transfer is based on a 240° transfer angle and a minimum energy transfer. But recall that a wide variety of elliptical arcs could be used to connect these two orbits. Perhaps the maneuver costs could be improved by extending the transfer time. Use the space triangle but try to extend the transfer time to 15 hours.
Produce the transfer and include the following:
$type, a, p, e, \mathcal{E}, v_{dep}, v_{arr}, \theta^*_{dep}, \theta^*_{arr}, \gamma_{dep}, \gamma_{arr}$. As usual, supply all the appropriate justifications for these results. Include the r_p and r_a distances for the transfer ellipse. Does the difference in the true anomalies equal the transfer angle?

The transfer is as follows.



From the characteristics of Mars, Phobos, and Deimos, we know that

$$\mu_{mars} = 4.2828e + 4\ km^3/s^2$$

$$R_{mars} = 3397\ km$$

$$r_1 = a_{phobos} = 9376\ km$$

$$r_2 = a_{deimos} = 23458\ km$$

$$v_1 = \sqrt{\frac{\mu_{mars}}{r_1}} = 2.1373\ km/s$$

$$v_2 = \sqrt{\frac{\mu_{mars}}{r_2}} = 1.3512\ km/s\ .$$

From the given transfer angle, $TA = 240°$ we know that this is a TYPE-2 transfer. And from the geometry of the space triangle we also know the following values

$$chord \coloneqq c = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 cos(360° - TA)} = 2.9294e + 4\ km$$

$$semi - perimeter \coloneqq s = \frac{r_1 + r_2 + c}{2} = 3.1064e + 4\ km\ .$$

For the minimum energy transfer we know that the transfer orbit is elliptical, and the semi-major axis is the possible minimum value.

$$a_{min} = \frac{s}{2} = 1.5532e + 4\ km\ .$$

Then,

$$\alpha_o = 2arcsin\sqrt{\frac{s}{2a_{min}}} = 180°$$

$$\beta_o = 2\text{arcsin}\sqrt{\frac{s-c}{2a_{min}}} = 27.6207°\ .$$

Using the 2A/2B Lambert TOF equations we can compute the minimum TOF for the transfer orbit

$$TOF_{min} = \sqrt{\frac{a_{min}^3}{\mu}}\left((\alpha_o - sin\alpha_o) + (\beta_o - sin\beta_o)\right) = 2.9558e + 4\ s = 8.2104\ hrs\ .$$

The minimum TOF is smaller than our assumed TOF of 15 hours. Thus, we can update our transfer orbit to be a TYPE-2B orbit

Now, if our orbit is a TYPE-2B, we can iterate over the Lambert TOF equation to find the optimal semi-major axis for the assumed TOF. This is computationally done using the MATLAB function '`find_lambertEllip_SMA()`' (the code is in the Appendix).

$$\mathrm{a} = 1.8040e+4\ km$$

$$\alpha = 136.2190°$$

$$\beta = 25.5940°$$

$$\mathrm{p} = \frac{4a(s-r_1)(s-r_2)}{c^2}\sin^2\left(\frac{\alpha \pm \beta}{2}\right) = 1.3524e+4\ \mathrm{km}\ \text{ or } 0.9378\mathrm{e}+4\ \mathrm{km}$$

We decide which semi-latus rectum values is valid by checking the true anomalies.

If $p = 1.3524e+4\ \ km$,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1}-1\right)\right) = +27.8257°,\quad -27.8257°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2}-1\right)\right) = +147.8257°,\quad -147.8257°.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep}$$

$$TA = -147.8257° - (-27.8257°) = -120° = 240°$$

$$\theta^*_{dep} = -27.8257°,\qquad \theta^*_{arr} = -147.8257°$$

Now we verify this by calculating the actual TOF with the eccentric anomalies. From the relationship

$$tan\frac{\theta^*}{2} = \left(\frac{1+e}{1-e}\right)^{0.5} tan\frac{E}{2}$$

$$E_{dep} = -16.2718°,\qquad E_{arr} = 233.1035°$$

Then

$$TOF_{verify} = \sqrt{\frac{a^3}{\mu_{mars}}}\left(E_{arr} - E_{dep} - e\left(sinE_{arr} - sinE_{dep}\right)\right) = 5.4000e+4\ s\ =\ 15\ hrs.$$

If $p = 0.9378e + 4\ km$,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right) = +89.9786°, \quad -89.9786°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right) = +150.0214°, \quad -150.0214°\,.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep} \Rightarrow 240°$$

$$\theta^*_{dep} = -89.9786°, \qquad \theta^*_{arr} = +150.0214°$$

Now we verify this by calculating the actual TOF with the eccentric anomalies. From the relationship

$$tan\frac{\theta^*}{2} = \left(\frac{1+e}{1-e}\right)^{0.5} tan\frac{E}{2}$$

$$E_{dep} = -46.1240°, \qquad E_{arr} = 115.6890°$$

Then

$$TOF_{verify} = \sqrt{\frac{a^3}{\mu_{mars}}}\left(E_{arr} - E_{dep} - e\left(sinE_{arr} - sinE_{dep}\right)\right) = 1.9906e + 4\ s\ =\ 5.5295\ hrs.$$

This does not equal 15 hours.

$$\therefore \theta^*_{dep} = -27.8257°, \qquad \theta^*_{arr} = -147.8257°$$

The actual time of flight becomes 15 hours.

Then,

$$\therefore p = 1.3524e + 4\ \ km$$

$$e = \sqrt{1 - \frac{p}{a}} = 0.5003$$

$$\mathcal{E} = -\frac{\mu_{mars}}{2a} = -1.1871\ \ km^2/s^2\,.$$

$$v_{dep} = \sqrt{\mu_{mars}\left(\frac{2}{r_1} - \frac{1}{a}\right)} = 2.6003\ km/s$$

$$v_{arr} = \sqrt{\mu_{mars}\left(\frac{2}{r_2} - \frac{1}{a}\right)} = 1.1302\ km/s$$

$$\gamma_{dep} = arccos\left(\frac{\sqrt{\mu_{mars}p}}{r_1 v_{dep}}\right) = \pm 9.1963°$$

$$\gamma_{arr} = \arccos\left(\frac{\sqrt{\mu_{mars}p}}{r_1 v_{arr}}\right) = \pm 24.8006°.$$

From the true anomalies we can tell if the orbit is ascending or descending, which means that we can tell whether the flight path angles are positive or negative.

$$\therefore \gamma_{dep} = -9.1963°, \qquad \gamma_{arr} = -24.8006°.$$

The radial distance of the periapsis and the apoapsis are

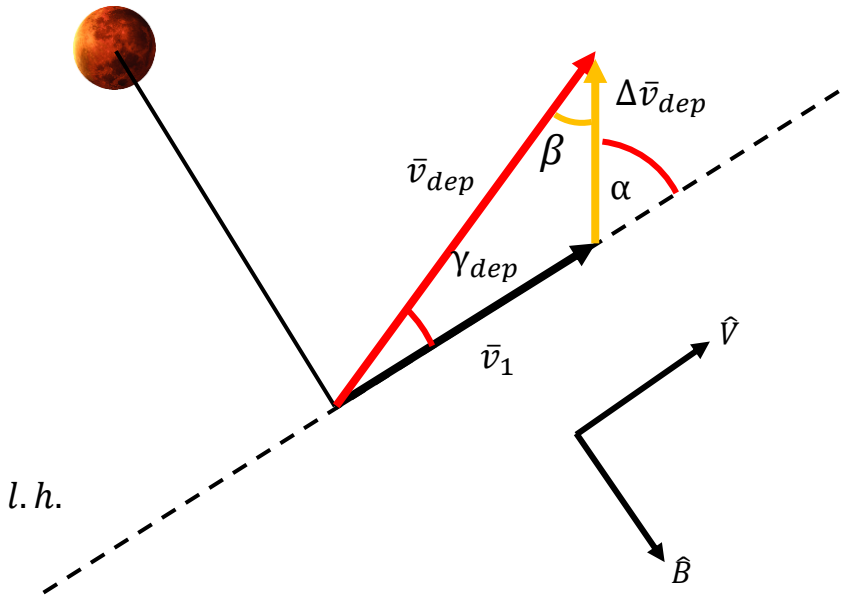
$$r_p = a(1-e) = 9.0145e + 3\ km$$

$$r_a = a(1+e) = 2.7065e + 4\ km.$$

From the calculations of the true anomalies, we can see that the difference between the true anomalies does equal the transfer orbit of $(-120°)\ 240°$.

(b) Determine the maneuvers at departure and arrival. i.e., $|\Delta\bar{v}|$ and α. Transform the maneuvers to VNB coordinates. How do the maneuvers compare to the minimum energy transfer in terms of time and total maneuver cost?

At departure:



From the cosine rule,

$$|\Delta\bar{v}_{dep}| = \sqrt{v_{dep}^2 + v_1^2 - 2v_{dep}v_1 cos\gamma_{dep}} = 0.5977\ km/s\,.$$

Then, from the sine rule and cosine rule

$$\beta = \arcsin\left(\frac{v_1}{\Delta v_{dep}} sin\gamma_{dep}\right) = -34.8510°,\quad -145.1490°$$

$$\beta = \arccos\left(\frac{\Delta v_{dep}^2 + v_{dep}^2 - v_1^2}{2v_{dep}\Delta v_{dep}}\right) = \pm 34.8510°$$

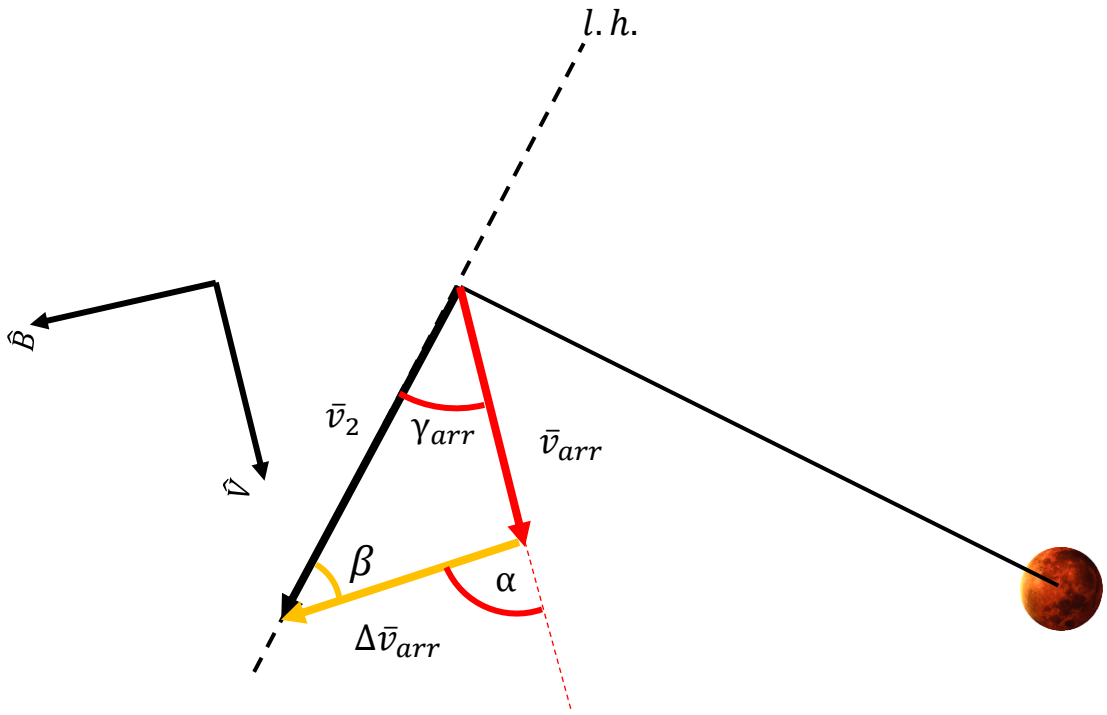
$$\therefore \beta = -34.8510°$$

$$\alpha = \beta + \gamma_{dep} = -44.0473°\,.$$

The delta-V can be expressed in the VNB coordinates with the α

$$\Delta\bar{v}_{dep} = |\Delta\bar{v}_{dep}|\left(cos\alpha\hat{V} + sin\alpha(\hat{B})\right) = 0.4296\hat{V} - 0.4156\hat{B}\ \text{km/s}\,.$$

At arrival:



From the cosine rule,

$$|\Delta\bar{v}_{arr}| = \sqrt{v_{arr}^2 + v_2^2 - 2v_{arr}v_2 cos\gamma_{arr}} = 0.5749 \text{ km/s}.$$

Then, from the sine rule and cosine rule

$$\beta = \arcsin\left(\frac{v_{arr}}{\Delta v_{arr}} sin\gamma_{arr}\right) = -55.5484°, \quad -124.4516°$$

$$\beta = arccos\left(\frac{v_2^2 + \Delta v_{arr}^2 - v_{arr}^2}{2v_2\Delta v_{arr}}\right) = \pm 55.5484°$$

$$\therefore \beta = -55.5484°$$

$$\alpha = \beta + \gamma_{arr} = -80.3490°.$$

The delta-V can be expressed in the VNB coordinates with the α

$$\Delta\bar{v}_{arr} = |\Delta\bar{v}_{arr}|\left(cos\alpha\hat{V} + sin|\alpha|\hat{B}\right) = 0.0964\hat{V} + 0.5568\hat{B} \text{ km/s}.$$

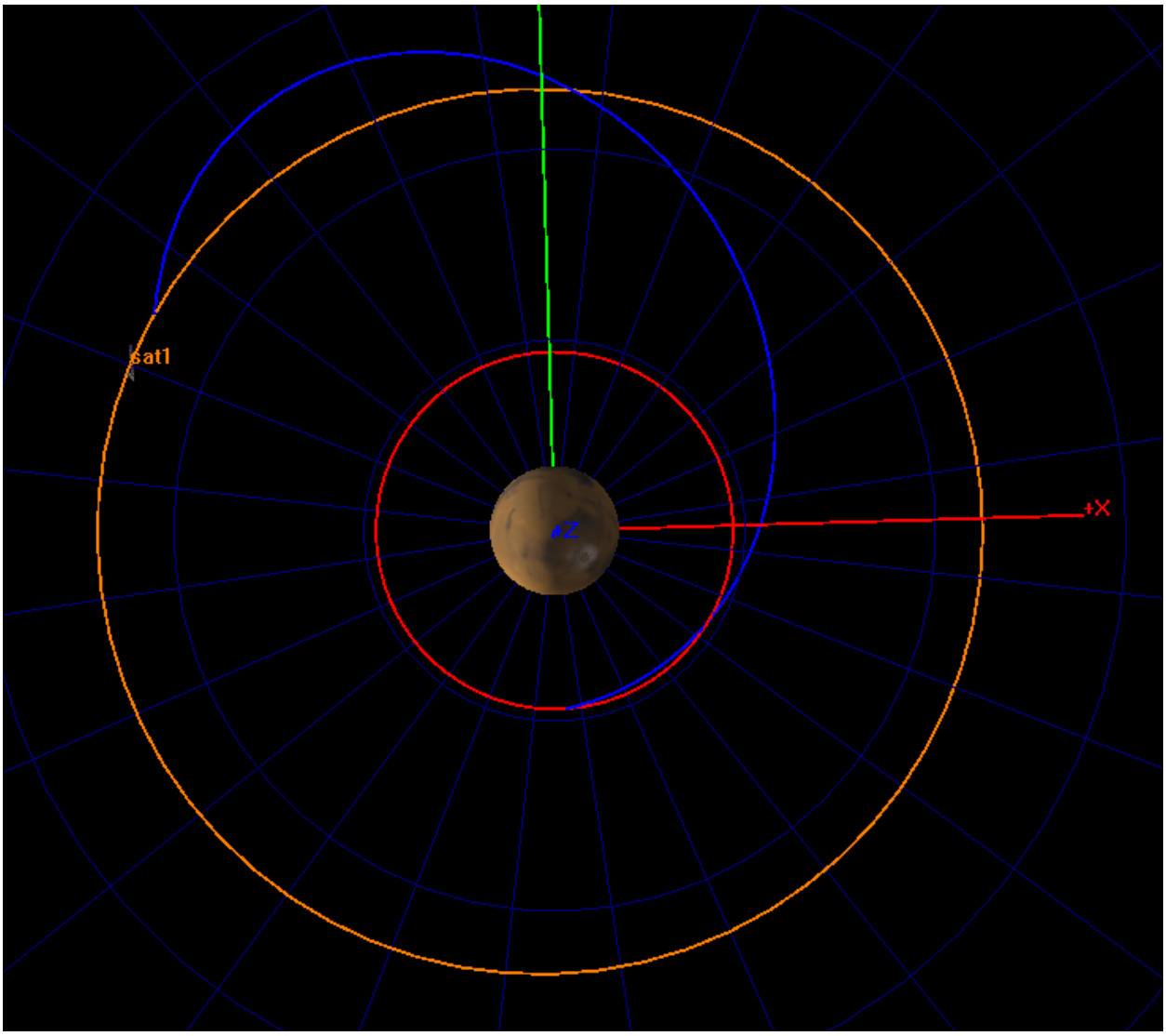
The total maneuver cost is

$$|\Delta\bar{v}_{tot}| = |\Delta\bar{v}_{dep}| + |\Delta\bar{v}_{arr}| = 1.1726\ km/s\ .$$

From the notes we know that for the minimal energy transfer case the total maneuver cost is 1.402 km/s. With a TOF of 15 hours there is an 16.3623% decrease in maneuver cost and 82.6847% increase in TOF. Whereby the transfer is efficient in terms of fuel but not in terms of time.

(c) Plot the transfer in GMAT. Plot a full revolution of the spacecraft orbit as the circular orbit of Phobos. Then apply the maneuver. Upon arrival, implement the arrival maneuver; end with a complete revolution in the final orbit (i.e., the circular orbit of Deimos).
Does the transfer pass through the periapsis or apoapsis?

The mission is Simulated in GMAT



We can see that the transfer passes through both the periapsis and apoapsis.

(d) To implement such a transfer and rendezvous with Deimos, it is necessary to phase the departure correctly. What is the required phase angle between Phobos and Deimos at departure? How often does the correct phase angle recur (in hours)? Compare this result to the periods of Phobos and Deimos.

The phase angle is computed as

$$\phi = TA - n_{deimos}(TOF) = TA - \sqrt{\frac{\mu_{mars}}{a_{demios}^3}}(TOF) = 61.7845° \ .$$

The synodic period of Phobos and Deimos is

$$s = \frac{2\pi}{n_{phobos} - n_{deimos}} = \frac{2\pi}{\sqrt{\frac{\mu_{mars}}{a_{phobos}^3}} - \sqrt{\frac{\mu_{mars}}{a_{demios}^3}}} = 3.6884e+4 \ s \ = \ 10.2456 \ hrs \ .$$

The periods of each orbit are

$$\mathbb{P}_{phobos} = \frac{1}{n_{phobos}} = 4.3869e+3 \ s \ = \ 1.2186 \ hrs$$

$$\mathbb{P}_{deimos} = \frac{1}{n_{deimos}} = 1.7361\text{e}+4 \ \text{s} \ = \ 4.8225 \ \text{hrs} \ .$$

For comparison

$$\frac{s}{\mathbb{P}_{phobos}} = 8.4077$$

$$\frac{s}{\mathbb{P}_{deimos}} = 2.1246$$

This shows that it takes Phobos 8.4077 periods and Deimos 2.1246 periods to have the correct phase angle between the two bodies.

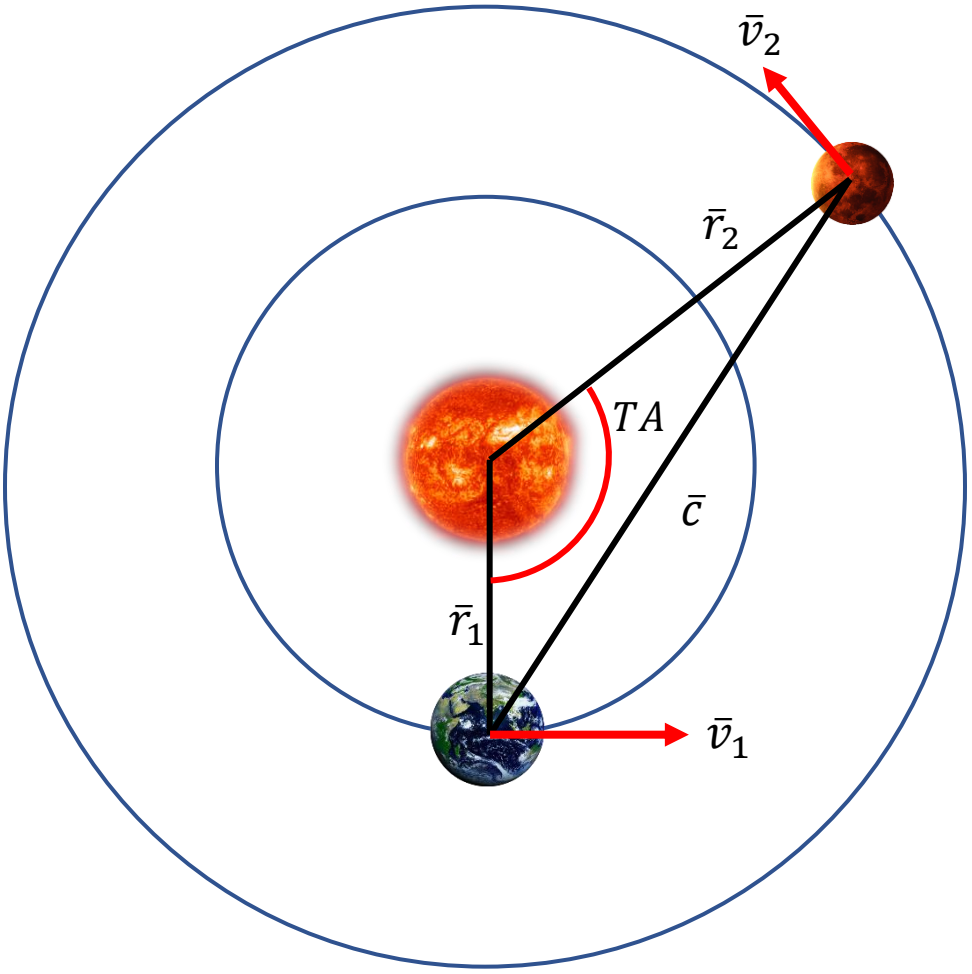
Problem 2:

Recall the 2015 movie "The Martian" where the character Mark Watney must be rescued from Mars. The astrodynamicist designs the critical transfer trajectory to send the spacecraft Hermes from Earth to Mars and enable the rescue. [Are there any other movies where the astrodynamicist is the star and 'saves the day'?!!]

Recall that Rich Purnell (the astrodynamicist) spent extensive time exploring Lambert arcs and incorporating an Earth gravity assist that could satisfy all the requirements! Let's explore the possible Earth-to-Mars transfer arcs. Assume that Earth and Mars move along circular coplanar orbits. For the Earth-to-Mars transfer, initially ignore the local fields; the relative two-body problem involves only solar gravity.

(a) Consider first a transfer with an angle of 120 degrees and a time of flight of 160 days. Given this space triangle, is the transfer elliptic or hyperbolic? A transfer of what type then emerges?

The orbit diagram is as follows.



From the characteristics of Earth and Mars, we know that

$$\mu_{sun} = 1.3271e + 11\ km^3/s^2$$

$$\mu_{earth} = 3.9860e + 5\ km^3/s^2$$

$$\mu_{mars} = 4.2828e + 4\ km^3/s^2$$

$$R_{mars} = 3397\ km$$

$$R_{earth} = 6378.1\ km$$

$$r_1 = a_{earth} = 149597898\ \text{km}$$

$$r_2 = a_{mars} = 227944135\ \text{km}$$

$$v_1 = \sqrt{\frac{\mu_{sun}}{r_1}} = 29.7847\ \text{km/s}$$

$$v_2 = \sqrt{\frac{\mu_{sun}}{r_2}} = 24.1291\ \text{km/s}\ .$$

From the given transfer angle, $TA = 120°$ we know that this is a TYPE-1 transfer. And from the geometry of the space triangle we also know the following values

$$chord \coloneqq c = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 cos(TA)} = 3.2930e + 8\ km$$

$$semi - perimeter \coloneqq s = \frac{r_1 + r_2 + c}{2} = 3.5342e + 8\ km\ .$$

Check if the transfer is elliptical or hyperbolic.

$$TOF_{par} = \frac{1}{3}\sqrt{\frac{2}{\mu_{sun}}}\left(s^{\frac{3}{2}} - (s-c)^{\frac{3}{2}}\right) = 97.7346\ days$$

Since, the TOF for the parabolic transfer orbit is smaller than the mission TOF (160 days) we know that the transfer orbit is going to be elliptic.

(b) Produce the transfer and include the following:
$type, a, p, e, \mathcal{E}, v_{dep}, v_{arr}, \theta^*_{dep}, \theta^*_{arr}, \gamma_{dep}, \gamma_{arr}$. As usual, supply all the appropriate justifications for these results. Include the r_p and r_a distances for the transfer ellipse. Does the difference in the true anomalies equal the transfer angle?

For the minimum energy transfer we know that the transfer orbit is elliptical, and the semi-major axis is the possible minimum value.

$$a_{min} = \frac{s}{2} = 1.7671e + 8\ km\ .$$

Then,

$$\alpha_o = 2arcsin\sqrt{\frac{s}{2a_{min}}} = 180°$$

$$\beta_o = 2\text{arcsin}\sqrt{\frac{s-c}{2a_{min}}} = 30.2884°\ .$$

Using the 1A/1B Lambert TOF equations we can compute the minimum TOF for the transfer orbit

$$TOF_{min} = \sqrt{\frac{a_{min}^3}{\mu}}\left((\alpha_o - sin\alpha_o) - (\beta_o - sin\beta_o)\right) = 2.0101e + 7\ s\ =\ 232.6504\ days\ .$$

Since the TOF for the minimal energy condition is larger than the mission TOF, we now know that the transfer orbit type is TYPE-1A.

Now, if our orbit is a TYPE-2B, we can iterate over the Lambert TOF equation to find the optimal semi-major axis for the assumed TOF. This is computationally done using the MATLAB function '`find_lambertEllip_SMA()`' (the code is in the Appendix).

$$\text{a} = 2.0028e + 8\ km$$

$$\alpha = 139.8777°$$

$$\beta = 28.4108°$$

$$\text{p} = \frac{4a(s-r_1)(s-r_2)}{c^2}sin^2\left(\frac{\alpha \pm \beta}{2}\right) = 1.8697\text{e} + 8\ \text{km}\ \text{ or } 1.2904\text{e} + 8\ \text{km}$$

We decide which semi-latus rectum values is valid by checking the true anomalies.

If $p =$ 1.2904e + 8 km,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right) = +103.3206°, \quad -103.3206°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right) = +136.6794°, \quad -136.6794°\,.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep} \Rightarrow 120°$$

No combinations of the true anomaly satisfy this condition, so this p is not valid.

If $p = 1.8697\text{e} + 8$ km,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right) = +14.2200°, \quad -14.2200°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right) = +134.2200°, \quad -134.2200°.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep}$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep} \Rightarrow 120°$$

$$\therefore \theta^*_{dep} = +14.2200°, \qquad \theta^*_{arr} = +134.2200°$$

Now we verify this by calculating the actual TOF with the eccentric anomalies. From the relationship

$$tan\frac{\theta^*}{2} = \left(\frac{1+e}{1-e}\right)^{0.5} tan\frac{E}{2}$$

$$E_{dep} = +10.9471°, \qquad E_{arr} = 122.4141°$$

Then

$$\text{TOF}_{verify} = \sqrt{\frac{a^3}{\mu_{sun}}}\left(E_{arr} - E_{dep} - e\left(sinE_{arr} - sinE_{dep}\right)\right) = 1.3824\text{e} + 6 \text{ s} = 159.9 \text{ days}.$$

This is equivalent to the expected TOF of 160 days.

$$\theta^*_{dep} = +14.2200°, \qquad \theta^*_{arr} = 134.2200°$$

$$\therefore p = 1.8697e + 8\ km$$

$$e = \sqrt{1 - \frac{p}{a}} = 0.2577$$

$$\mathcal{E} = -\frac{\mu_{sun}}{2a} = -331.3245 \ km^2/s^2 .$$

$$v_{dep} = \sqrt{\mu_{sun}\left(\frac{2}{r_1} - \frac{1}{a}\right)} = 33.3408 \ km/s$$

$$v_{arr} = \sqrt{\mu_{sun}\left(\frac{2}{r_2} - \frac{1}{a}\right)} = 22.4005 \ km/s$$

$$\gamma_{dep} = arccos\left(\frac{\sqrt{\mu_{sun}p}}{r_1 v_{dep}}\right) = \pm 2.8998°$$

$$\gamma_{arr} = \arccos\left(\frac{\sqrt{\mu_{sun}p}}{r_1 v_{arr}}\right) = \pm 12.6903°.$$

From the true anomalies we can tell if the orbit is ascending or descending, which means that we can tell whether the flight path angles are positive or negative.

$$\therefore \gamma_{dep} = +2.8998°, \qquad \gamma_{arr} = +12.6903°.$$

The radial distance of the periapsis and the apoapsis are

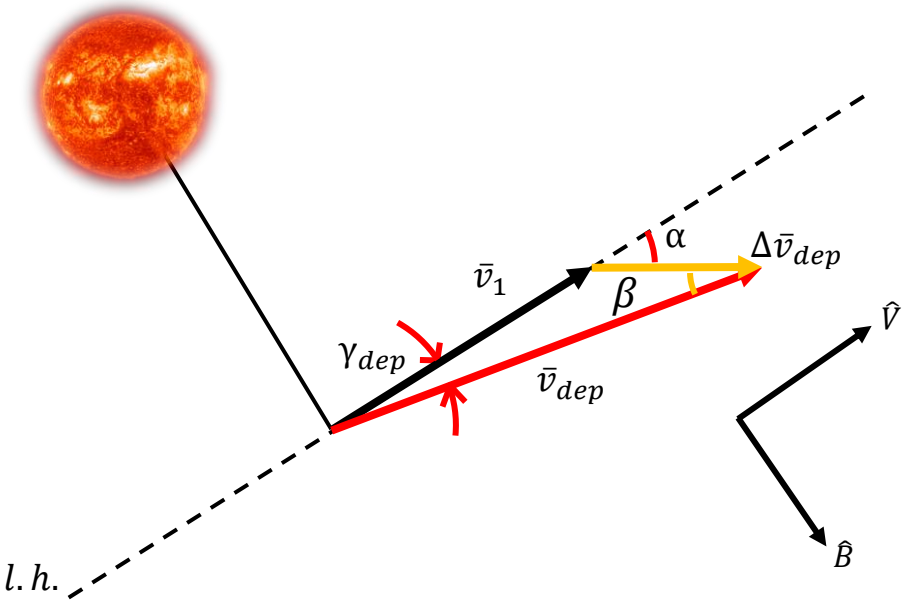
$$r_p = a(1 - e) = 1.4866e + 8 \ km$$

$$r_a = a(1 + e) = 2.5189e + 8 \ km.$$

From the calculations of the true anomalies, we can see that the difference between the true anomalies equal the transfer orbit of 120°.

(c) Determine the maneuvers at departure and arrival. i.e., $|\Delta\bar{v}|$ and α. Transform the maneuvers to VNB coordinates.

At departure:



From the cosine rule,

$$|\Delta\bar{v}_{dep}| = \sqrt{v_{dep}^2 + v_1^2 - 2v_{dep}v_1 cos\gamma_{dep}} = 3.8973\ km/s\,.$$

Then, from the sine rule and cosine rule

$$\beta = \arcsin\left(\frac{v_1}{\Delta v_{dep}} sin\gamma_{dep}\right) = 22.7450°,\ \ 157.2550°$$

$$\beta = \arccos\left(\frac{\Delta v_{dep}^2 + v_{dep}^2 - v_1^2}{2v_{dep}\Delta v_{dep}}\right) = \pm 22.7450°$$

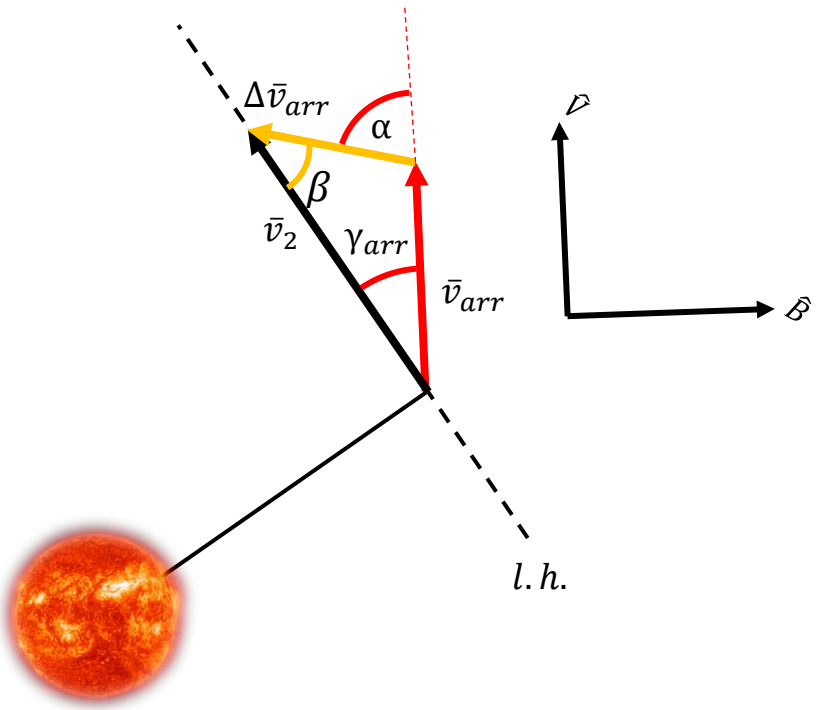
$$\therefore \beta = 22.7450°$$

$$\alpha = \beta + \gamma_{dep} = 25.6448°\,.$$

The delta-V can be expressed in the VNB coordinates with the α

$$\Delta\bar{v}_{dep} = |\Delta\bar{v}_{dep}|\left(cos|\alpha|\hat{V} + sin|\alpha|(\hat{B})\right) = 3.5134\hat{V} + 1.6867\hat{B}\ \text{km/s}\,.$$

At arrival:



From the cosine rule,

$$|\Delta\bar{v}_{arr}| = \sqrt{v_{arr}^2 + v_2^2 - 2v_{arr}v_2 cos\gamma_{arr}} = 5.4218 \text{ km/s}.$$

Then, from the sine rule and cosine rule

$$\beta = \arcsin\left(\frac{v_{arr}}{\Delta v_{arr}} sin\gamma_{arr}\right) = 65.1801°,\ \ 114.8199°$$

$$\beta = arccos\left(\frac{v_2^2 + \Delta v_{arr}^2 - v_{arr}^2}{2v_2\Delta v_{arr}}\right) = \pm 65.1801°$$

$$\therefore \beta = \ 65.1801°$$

$$\alpha = \beta + \gamma_{arr} = \ 77.8705°.$$

The delta-V can be expressed in the VNB coordinates with the α

$$\Delta\bar{v}_{arr} = |\Delta\bar{v}_{arr}|\left(cos\alpha\hat{V} + sin\alpha(-\hat{B})\right) = 1.1392\hat{V} - 5.3007\hat{B} \text{ km/s}.$$

The total maneuver cost is

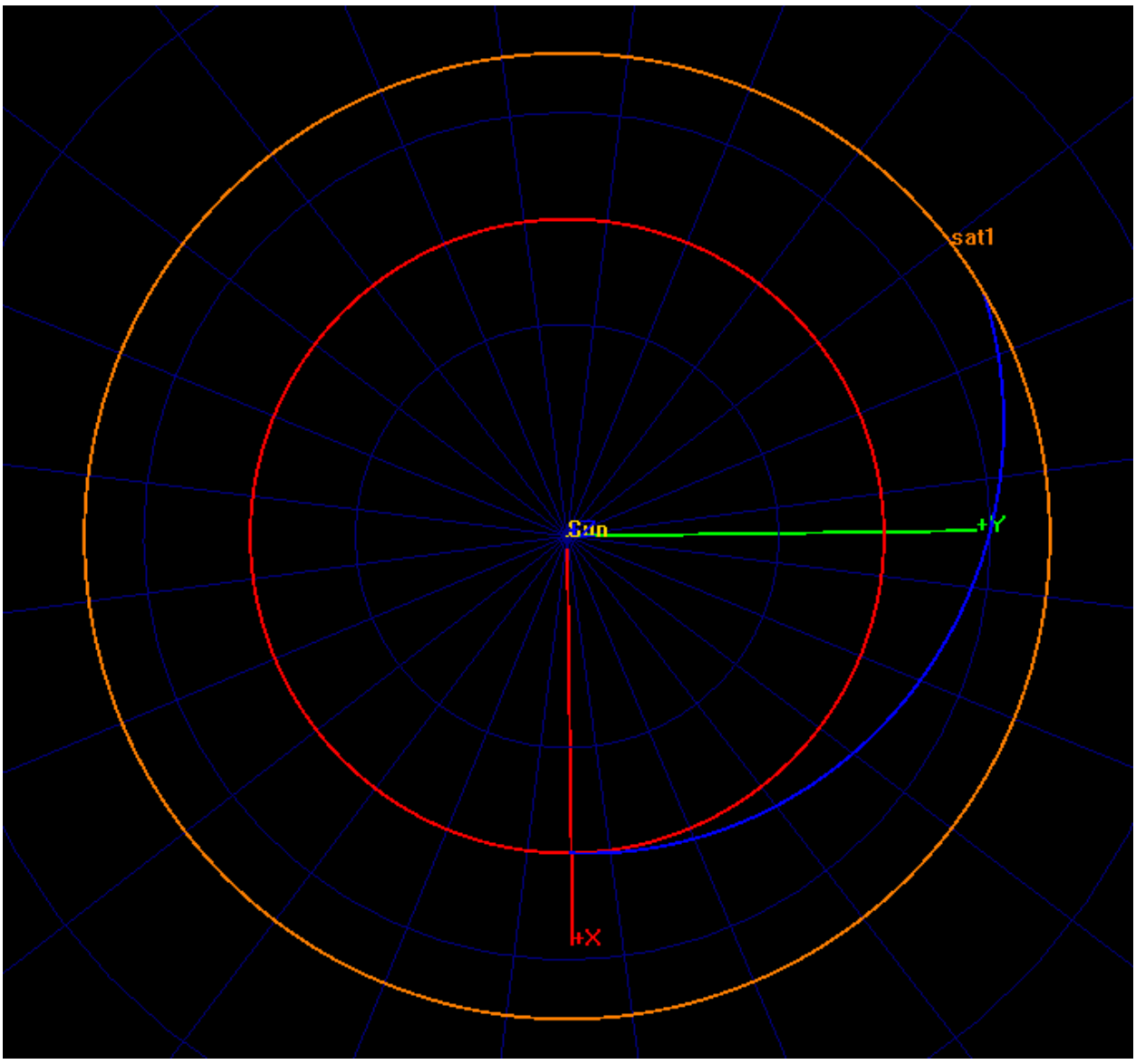
$$|\Delta\bar{v}_{tot}| = \left|\Delta\bar{v}_{dep}\right| + |\Delta\bar{v}_{arr}| = 9.3190\ km/s\,.$$

(d) Plot the transfer using either GMAT or Matlab. (Recall that GMAT gives you a chance to check your results.) Include the orbit of Earth; then apply the maneuver. After the suitable transfer angle, apply the second maneuver. Include the Mars orbit as well.

Red: Orbit of Earth

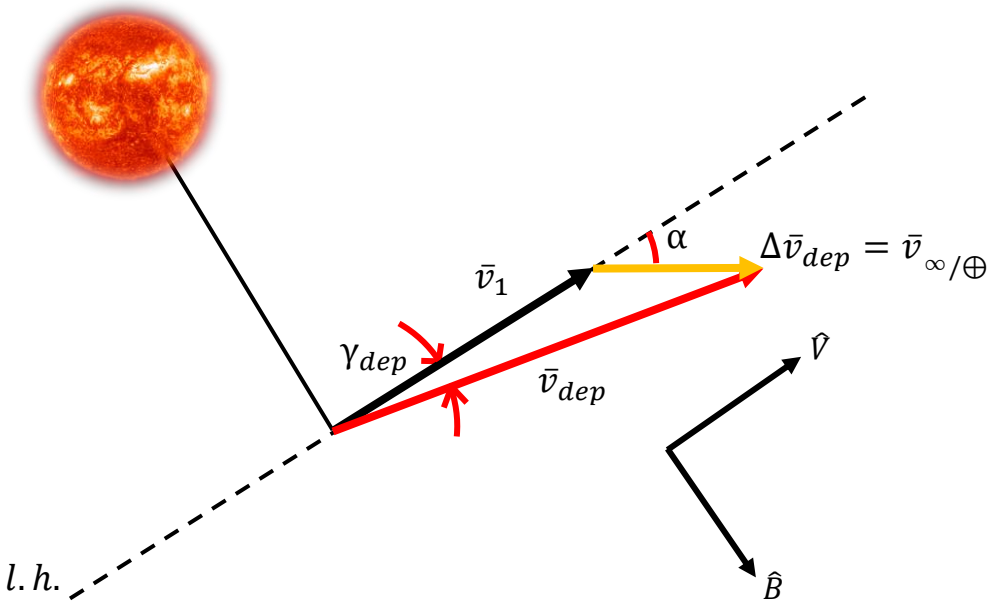
Blue: Transfer Orbit

Orange: Mars Orbit



(e) Now consider the Earth local field. [In the movie, a vehicle was launched from Earth to rendezvous with the Hermes rescue vehicle at Earth closest approach to receive additional supplies for the return trip to Mars.]
The transfer computed in (b)-(c) requires a $\bar{v}_{\infty/\oplus}$ relative to Earth in the Earth local view. What is the magnitude of this $\bar{v}_{\infty/\oplus}$? Assume that the pass distance at the Earth is required to be 1000 km altitude. What is the velocity magnitude at closest approach along the hyperbolic path?
Of course, a diagram of the local view is necessary. Should the rescue vehicle pass ahead or behind Earth?
Is it reasonable to attempt to rendezvous with the rendezvous vehicle moving at such a speed?

From the diagram,

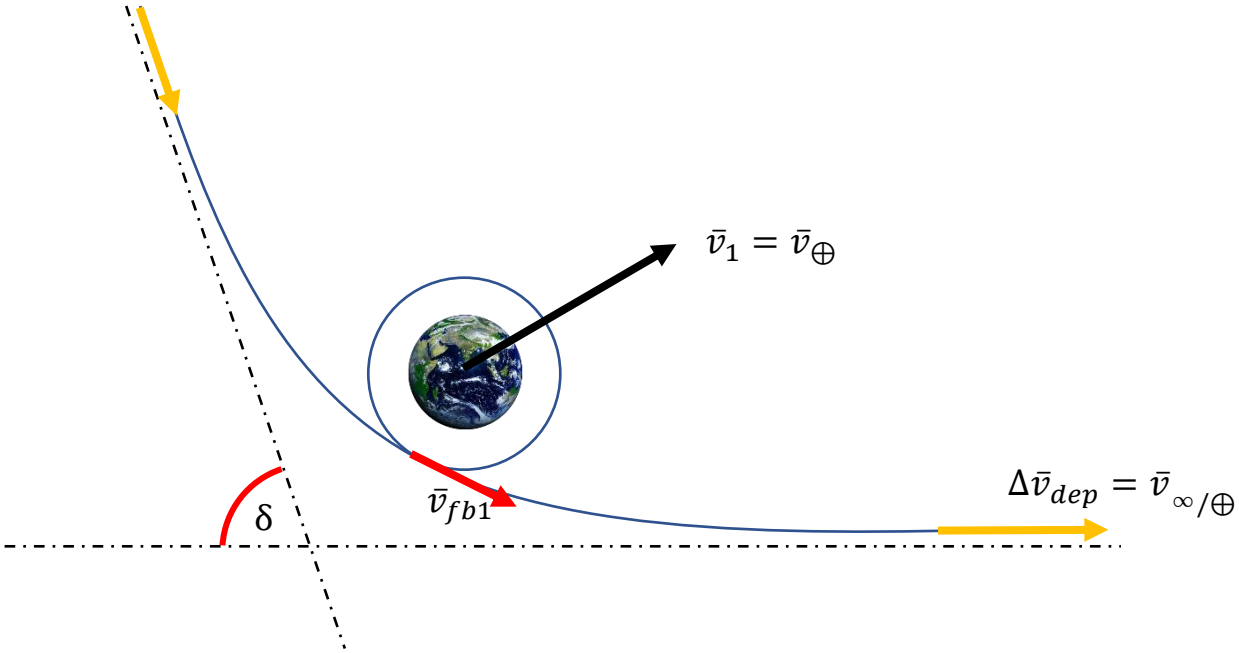


$$\bar{v}_{\infty/\oplus} = \Delta\bar{v}_{dep} = 3.5134\hat{V} + 1.6867\hat{B}\ \text{km/s}\,.$$

$$\therefore v_{\infty/\oplus} = 3.8973\ km/s\ .$$

To consider the local field of the Earth we have to view the departure in a geocentric perspective.

The geocentric view:



From the hyperbola characteristics

$$h = 1000$$

$$\xi_{\text{fb}} = \frac{\left(v_{\infty/\oplus}\right)^2}{2} = 7.5944 \ km^2/s^2$$

$$a_{\text{fb}} = -\frac{\mu_\oplus}{2\xi_{\text{fb}}} = -2.6243\text{e}+4 \ \text{km}$$

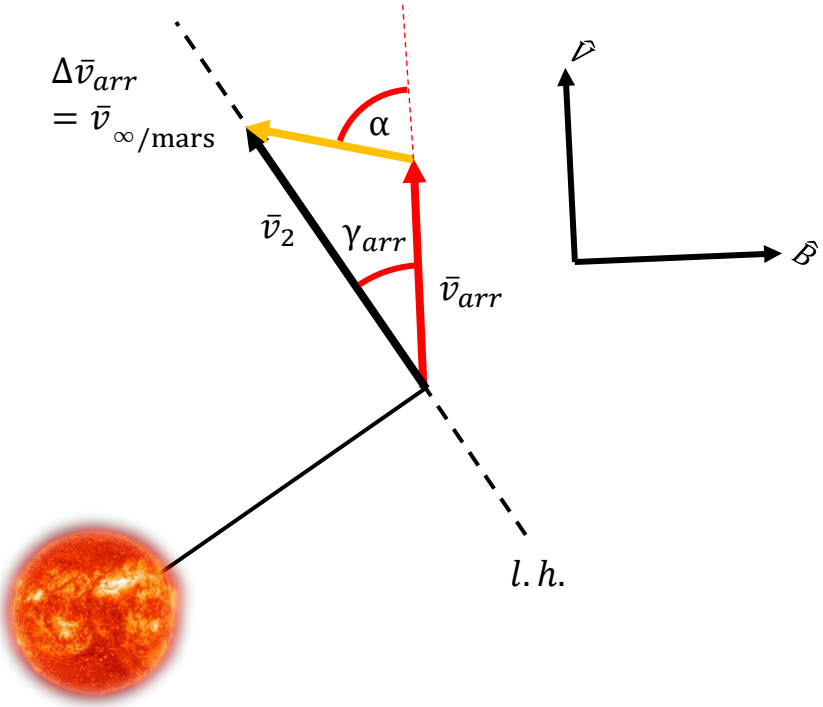
$$e_{fb} = 1 - \frac{R_\oplus + h}{a_{fb}} = 1.2811$$

$$\delta = 2arcsin\left(\frac{1}{e_{fb}}\right) = 102.6220°$$

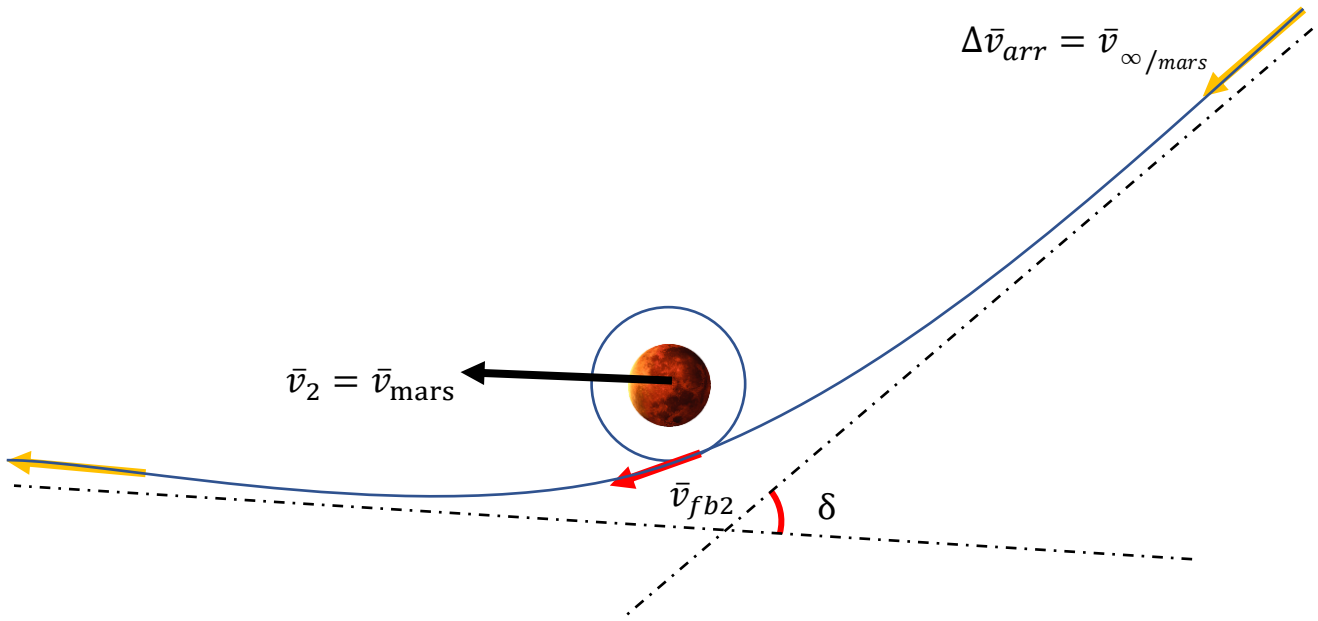
$$v_{fb1} = \sqrt{\mu_\oplus\left(\frac{2}{R_\oplus + h} - \frac{1}{a_{fb}}\right)} = 11.1013 \ km/s \ .$$

Since the new velocity v^+ becomes larger than the circular velocity the rescue vehicle should pass behind the Earth like in the diagram above to increase the energy. It is possible to rendezvous with a vehicle with such high velocity but it would be very difficult and it would be rather reasonable to seek for better solutions.

(f) In the Mars local field, what is the speed at periapsis if the Mars pass distance is 500 km altitude? [Recall that Mark Watney was required to achieve nearly this necessary speed as a result of his launch from the Martian surface to be rescued…hmmmm…..good thing that it is only a movie!]



The Mars centered view:



From the hyperbola characteristics

$$h = 500$$

$$\xi_{\text{fb}} = \frac{\left(v_{\infty/\text{mars}}\right)^2}{2} = 14.6977 \ km^2/s^2$$

$$a_{\text{fb}} = -\frac{\mu_{\text{mars}}}{2\xi_{\text{fb}}} = -1.4570\text{e}+3 \ \text{km}$$

$$e_{fb} = 1 - \frac{R_{\text{mars}} + h}{a_{fb}} = 3.6747$$

$$\delta = 2arcsin\left(\frac{1}{e_{fb}}\right) = 31.5820°$$

$$v_{fb2} = \sqrt{\mu_{\text{mars}}\left(\frac{2}{R_{mars} + h} - \frac{1}{a_{fb}}\right)} = 7.1677 \ \text{km/s}.$$

Problem 3:

Recall Problem 2. We are still trying to get to Mars to save Mark Watney—yep, we are still using the 2015 movie to explore that challenges of getting to Mars! Following up on Problem 2, consider the fact that the trip requires a significant time interval and survival on the Mars surface while awaiting rescue, Watney is running out of food. This time try to speed up the trip to Mars. Still assume all the same conditions as in Problem 2, i.e., use the SAME space triangle. But reduce the trip time—

(a) Consider a transfer with a transfer angle of 120 degrees and a time of flight of 92 days. Given this space triangle, is the transfer elliptic or hyperbolic? A transfer of what type then emerges?

From Problem 2 we know that the TOF for the parabolic transfer orbit is

$$TOF_{par} = 97.7346\ days\ .$$

Since the mission TOF of 92 days is smaller than this value we know that the transfer orbit for the mission is going to a hyperbolic trajectory. Furthermore, the TA of 120° indicate that the transfer orbit type is going to be TYPE-1H.

Repeat the steps in Problem 2 for this new time:

(b) Produce the transfer and include the following:
$type, a, p, e, \mathcal{E}, v_{dep}, v_{arr}, \theta^*_{dep}, \theta^*_{arr}, \gamma_{dep}, \gamma_{arr}$. As usual, supply all the appropriate justifications for these results. Include the r_p and r_a distances as determined. Does the difference in the true anomalies equal the transfer angle?

Now, if our orbit is a TYPE-1H, we can iterate over the Hyperbolic Lambert TOF equation

$$TOF = \sqrt{\frac{|a|^3}{\mu_{sun}}}\left((sinh\alpha - \alpha) - (sinh\beta - \beta)\right)$$

to find the optimal semi-major axis for the assumed TOF. This is computationally done using the MATLAB function '`find_lambertHyp_SMA()`' (the code is in the Appendix).

$$\mathrm{a} = -8.2606e + 8\ km$$

$$\alpha = 51.2724°$$

$$\beta = 13.8128°$$

$$\mathrm{p} = \frac{4|a|(s-r_1)(s-r_2)}{c^2} sinh^2\left(\frac{\alpha \pm \beta}{2}\right) = 2.7962\mathrm{e}+8\ \mathrm{km}\ \text{ or } 0.8629\mathrm{e}+8\ \mathrm{km}$$

We decide which semi-latus rectum values is valid by checking the true anomalies.

If $p = 0.8629\mathrm{e}+8$ km,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right) = +113.7475°,\ \ -113.7475°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right) = +126.2525°,\ \ -126.2525°\,.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep} \Rightarrow\ 120°$$

No combinations of the true anomaly satisfy this condition, so this p is not valid.

If $p = 2.7962\mathrm{e}+8$ km,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right) = +41.3007°,\ \ -41.3007°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right) = +78.6993°, \quad -78.6993°.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep}$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep} \Rightarrow 120°$$

$$\therefore \theta^*_{dep} = -41.3007°, \qquad \theta^*_{arr} = +78.6993°$$

Now we verify this by calculating the actual TOF with the eccentric anomalies. From the relationship

$$\tan\frac{\theta^*}{2} = \left(\frac{e+1}{e-1}\right)^{0.5} \tanh\frac{H}{2}$$

$$H_{dep} = -11.6896°, \qquad H_{arr} = 25.7699°$$

Then

$$TOF_{verify} = \sqrt{\frac{|a|^3}{\mu_{sun}}}\left(e\left(sinhH_{arr} - sinhH_{dep}\right) - \left(H_{arr} - H_{dep}\right)\right) = 7.9488e+6 \text{ s}$$
$$= 92 \text{ days.}$$

This is equivalent to the expected TOF of 92 days.

$$\theta^*_{dep} = -41.3007°, \qquad \theta^*_{arr} = +78.6993°$$

$$\therefore p = 2.7962e+8 \; km$$

$$e = \sqrt{1 - \frac{p}{a}} = 1.1569$$

$$\mathcal{E} = -\frac{\mu_{sun}}{2a} = 80.3290 \; km^2/s^2 \; .$$

$$v_{dep} = \sqrt{\mu_{sun}\left(\frac{2}{r_1} - \frac{1}{a}\right)} = 43.9877 \; km/s$$

$$v_{arr} = \sqrt{\mu_{sun}\left(\frac{2}{r_2} - \frac{1}{a}\right)} = 36.4018 \; km/s$$

$$\gamma_{dep} = arccos\left(\frac{\sqrt{\mu_{sun}p}}{r_1 v_{dep}}\right) = \pm 22.2211°$$

$$\gamma_{arr} = \arccos\left(\frac{\sqrt{\mu_{sun}p}}{r_1 v_{arr}}\right) = \pm 42.7637°.$$

From the true anomalies we can tell if the orbit is ascending or descending, which means that we can tell whether the flight path angles are positive of negative.

$$\therefore \gamma_{dep} = -22.2211°, \qquad \gamma_{arr} = +42.7637°.$$

The radial distance of the periapsis and the apoapsis are

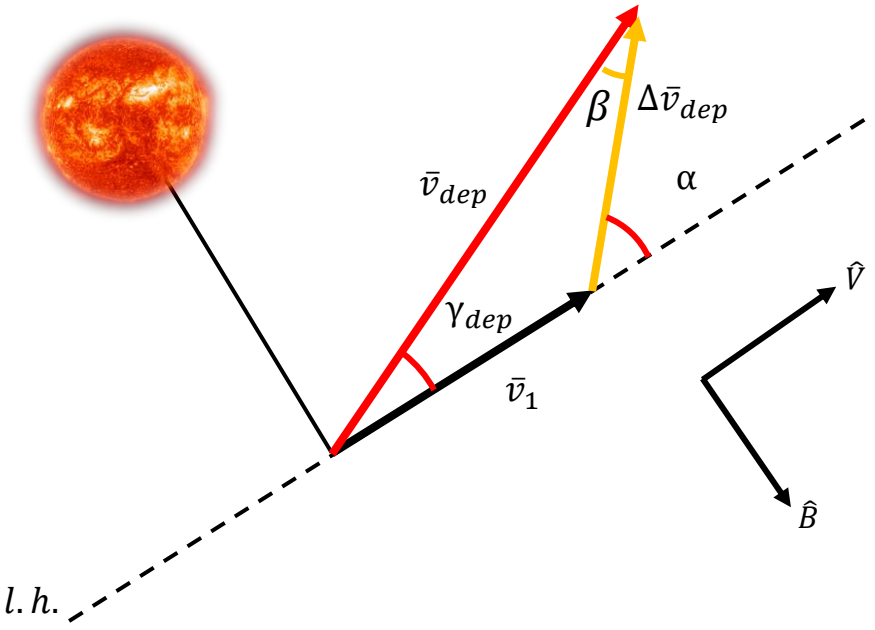
$$r_p = a(1-e) = 1.2964e + 8\ km$$

$$r_a = \infty.$$

From the calculations of the true anomalies, we can see that the difference between the true anomalies equal the transfer orbit of 120°.

(c) Determine the maneuvers at departure and arrival. i.e., $|\Delta\bar{v}|$ and α. Transform the maneuvers to VNB coordinates.

At departure:



From the cosine rule,

$$|\Delta\bar{v}_{dep}| = \sqrt{v_{dep}^2 + v_1^2 - 2v_{dep}v_1 cos\gamma_{dep}} = 19.9081\ km/s\ .$$

Then, from the sine rule and cosine rule

$$\beta = \arcsin\left(\frac{v_1}{\Delta v_{dep}} sin\gamma_{dep}\right) = -34.4580°,\quad -145.5420°$$

$$\beta = \arccos\left(\frac{\Delta v_{dep}^2 + v_{dep}^2 - v_1^2}{2v_{dep}\Delta v_{dep}}\right) = \pm 34.4580°$$

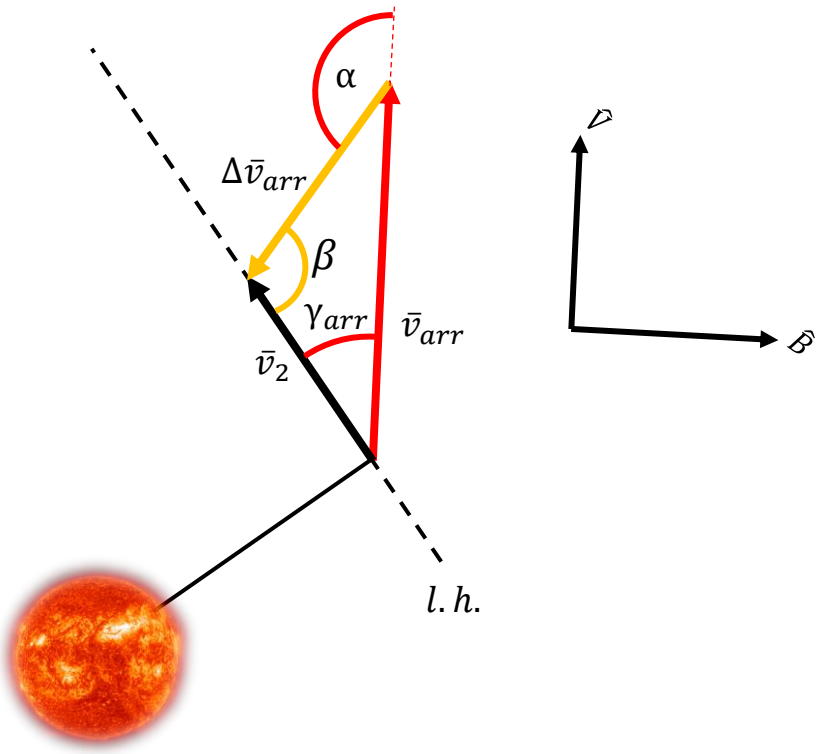
$$\therefore \beta = -34.4580°$$

$$\alpha = \beta + \gamma_{dep} = -56.6791°.$$

The delta-V can be expressed in the VNB coordinates with the α

$$\Delta\bar{v}_{dep} = |\Delta\bar{v}_{dep}|\left(cos\alpha\hat{V} + sin\alpha(\hat{B})\right) = 10.9361\hat{V} - 16.6353\hat{B}\ \text{km/s}\ .$$

At arrival:



From the cosine rule,

$$|\Delta\bar{v}_{arr}| = \sqrt{v_{arr}^2 + v_2^2 - 2v_{arr}v_2 cos\gamma_{arr}} = 24.8519 \text{ km/s}.$$

Then, from the sine rule and cosine rule

$$\beta = \arcsin\left(\frac{v_{arr}}{\Delta v_{arr}} sin\gamma_{arr}\right) = 84.0050°,\ \ 95.9950°$$

$$\beta = arccos\left(\frac{v_2^2 + \Delta v_{arr}^2 - v_{arr}^2}{2v_2\Delta v_{arr}}\right) = \pm 95.9950°$$

$$\therefore \beta = \ 95.9950°$$

$$\alpha = \beta + \gamma_{arr} = \ 138.7587°.$$

The delta-V can be expressed in the VNB coordinates with the α

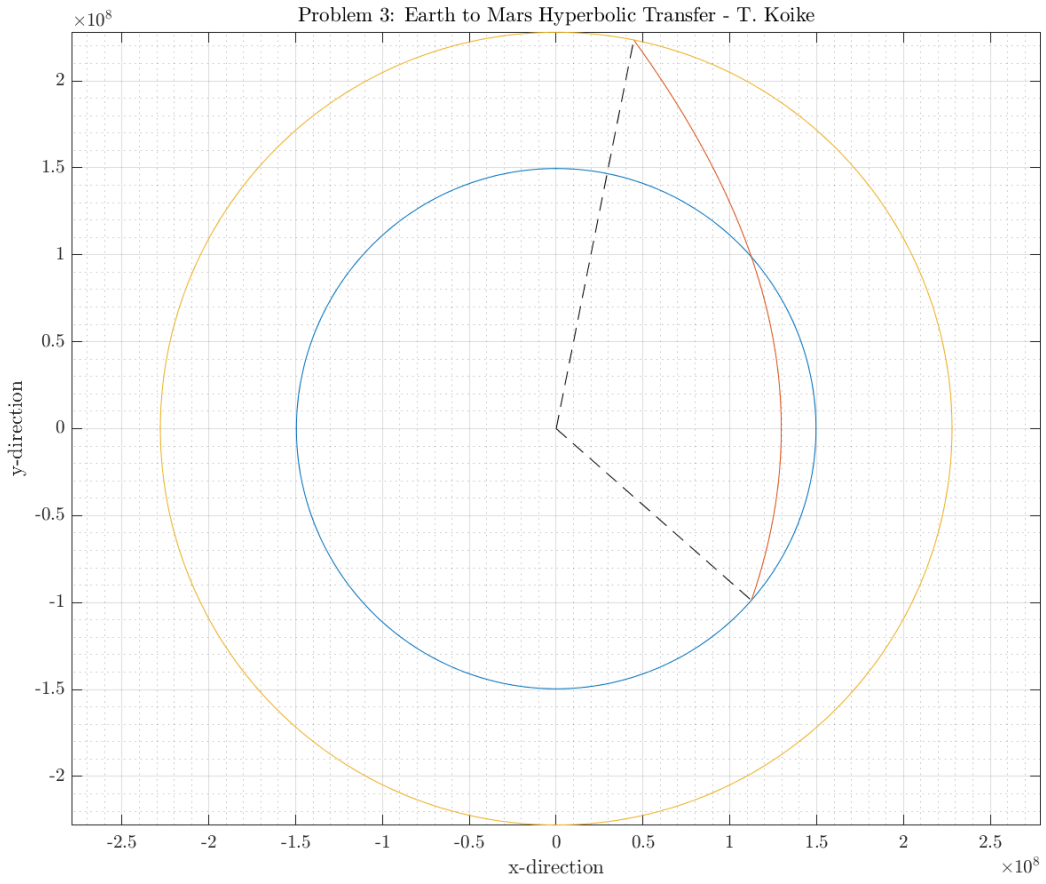
$$\Delta\bar{v}_{arr} = |\Delta\bar{v}_{arr}|\left(cos\alpha\hat{V} + sin\alpha\left(-\hat{B}\right)\right) = -18.6871\hat{V} - 16.3831\hat{B} \text{ km/s}.$$

The total maneuver cost is

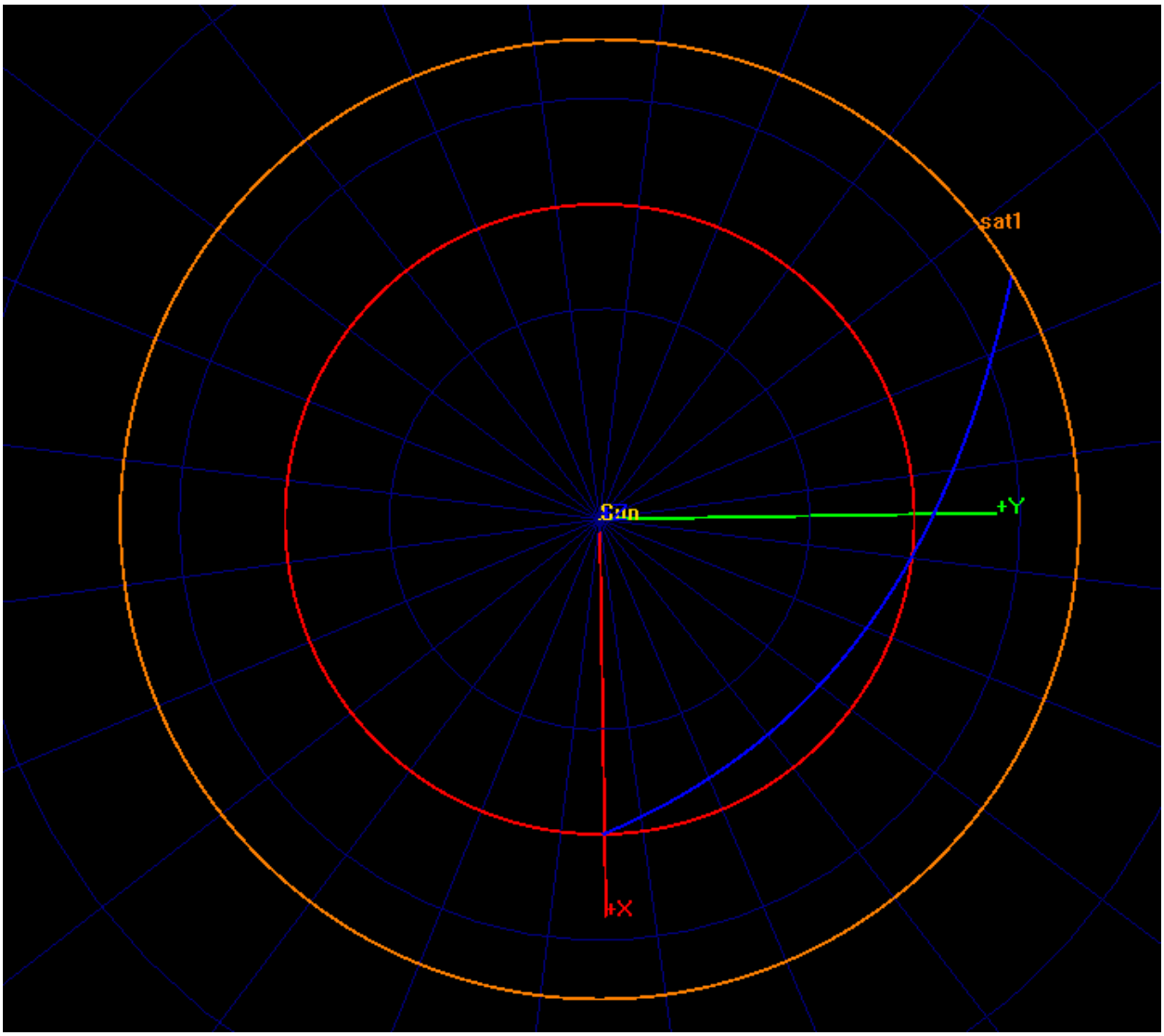
$$|\Delta\bar{v}_{tot}| = \left|\Delta\bar{v}_{dep}\right| + |\Delta\bar{v}_{arr}| = 44.7599\ km/s\ .$$

(d) Plot the transfer using either GMAT or MATLAB. (Recall that GMAT gives you a chance to check your results.) Include the orbit of Earth; then apply the maneuver. After the suitable transfer angle, apply the second maneuver. Include the Mars orbit as well.

MATLAB

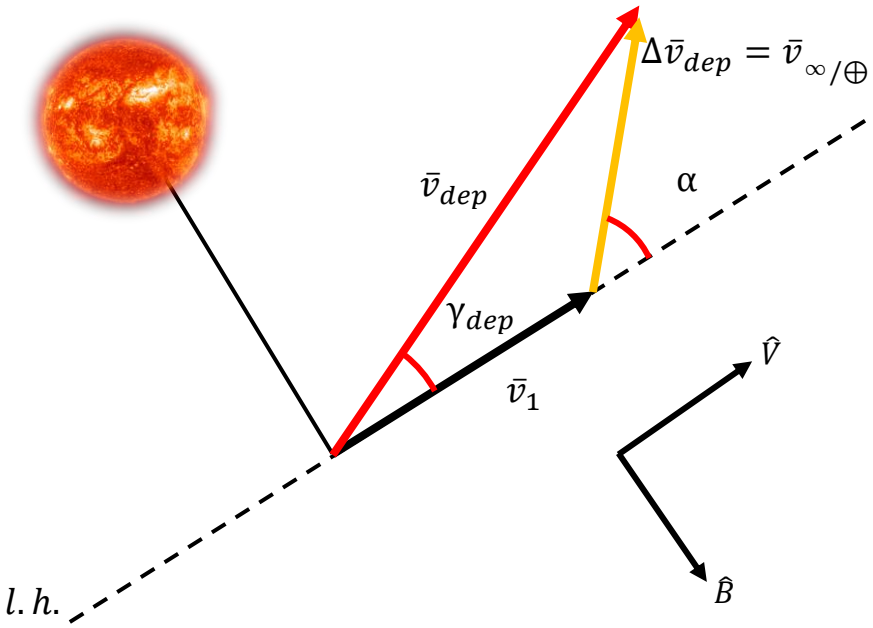


GMAT



(e) Now consider the Earth local field. [In the movie, a vehicle was launched from Earth to rendezvous with the Hermes rescue vehicle at Earth closest approach to receive additional supplies for the return trip to Mars.] The transfer computed in (b)-(c) requires a $\bar{v}_{\infty/\oplus}$ relative to Earth in the Earth local view. What is the magnitude of this $\bar{v}_{\infty/\oplus}$? Assume that the pass distance at the Earth is required to be 1000 km altitude.
What is the velocity magnitude at closest approach along the hyperbolic path?
Of course, a diagram of the local view is necessary. Should the rescue vehicle pass ahead or behind Earth?
Is it reasonable to attempt to rendezvous with the rendezvous vehicle moving at such a speed?

From the diagram,

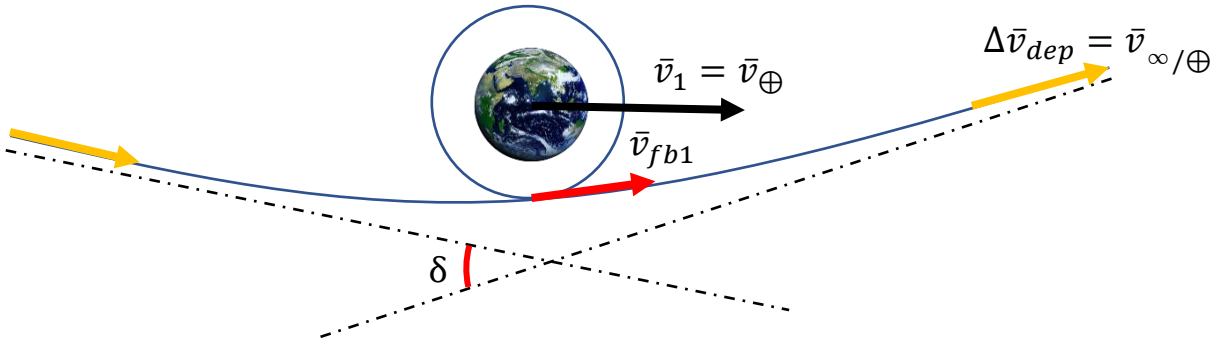


$$\bar{v}_{\infty/\oplus} = \Delta\bar{v}_{dep} = 10.9361\hat{V} - 16.6353\hat{B}\ \text{km/s}\,.$$

$$\therefore v_{\infty/\oplus} = 19.9081\ km/s\ .$$

To consider the local field of the Earth we have to view the departure in a geocentric perspective.

The geocentric view:



From the hyperbola characteristics

$$h = 1000$$

$$\xi_{\text{fb}} = \frac{\left(v_{\infty/\oplus}\right)^2}{2} = 198.1657 \ \ km^2/s^2$$

$$a_{\text{fb}} = -\frac{\mu_\oplus}{2\xi_{\text{fb}}} = -1.0057\text{e}+3 \text{ km}$$

$$e_{fb} = 1 - \frac{R_\oplus + h}{a_{fb}} = 8.3361$$

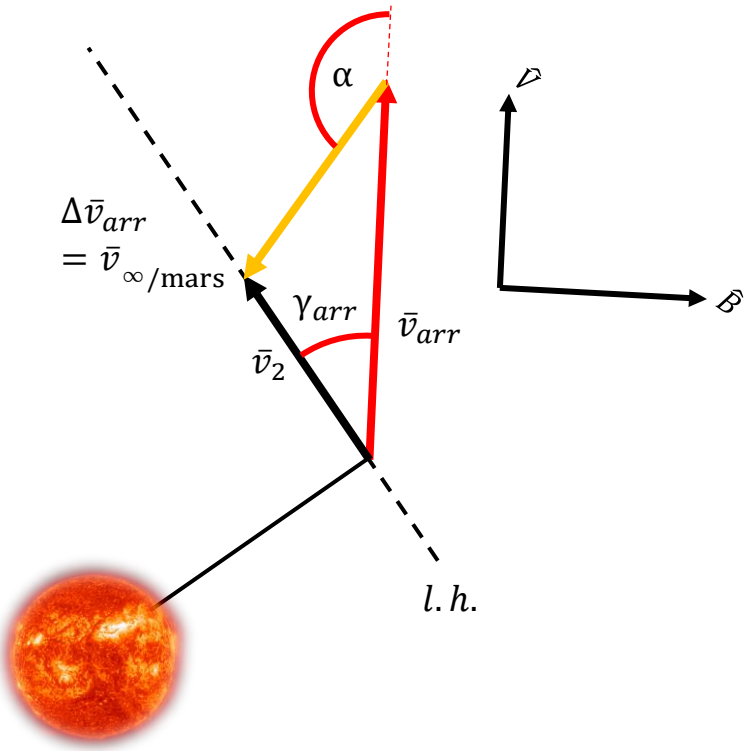
$$\delta = 2arcsin\left(\frac{1}{e_{fb}}\right) = 13.7795°$$

$$v_{fb1} = \sqrt{\mu_\oplus\left(\frac{2}{R_\oplus + h} - \frac{1}{a_{fb}}\right)} = 22.4584 \ \ km/s\,.$$

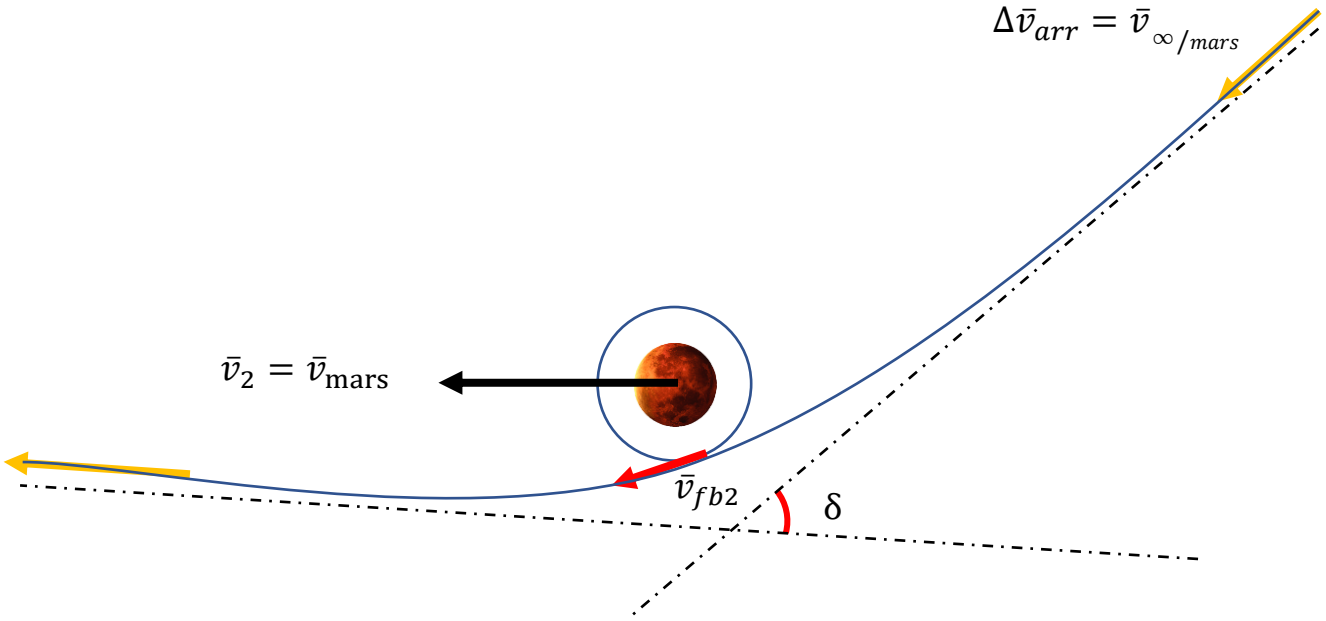
Since the new velocity v^+ becomes larger than the circular velocity the rescue vehicle should pass behind the Earth like in the diagram above to increase the energy. Considering the significantly large delta-V value and flyby velocity at the proximity of Earth, the attempt to rendezvous with the rendezvous would be devastating and unreasonable.

(f) In the Mars local field, what is the speed at periapsis if the Mars pass distance is 500 km altitude? Discuss: Is it likely that Watney could have reached this speed in his launch from the Mars surface?

Is it reasonable to try to reach Mars in 92 days?



Mars centered view:



From the hyperbola characteristics

$$h = 500$$

$$\xi_{\text{fb}} = \frac{\left(v_{\infty/\text{mars}}\right)^2}{2} = 300.8075 \ km^2/s^2$$

$$a_{\text{fb}} = -\frac{\mu_{\text{mars}}}{2\xi_{\text{fb}}} = -69.3447 \ \text{km}$$

$$e_{fb} = 1 - \frac{R_{\text{mars}} + h}{a_{fb}} = 57.1975$$

$$\delta = 2arcsin\left(\frac{1}{e_{fb}}\right) = 2.0035°$$

$$v_{fb2} = \sqrt{\mu_{\text{mars}}\left(\frac{2}{R_{mars} + h} - \frac{1}{a_{fb}}\right)} = 25.2902 \ \text{km/s}.$$

With the limited amount of resource and impending situation it would have be unlikely to gain such a significantly large velocity at the Mars surface. It would be only possible with some bizarre alien technology. From this conclusion, we can say that the large velocity maneuvers render it unreasonable to reach Mars in 92 days. It is considerably inefficient than having a longer TOF with an elliptical transfer orbit.

Problem 4:

Recall Problems 2 and 3. Both problems considered the challenge of an Earth-to-Mars transfer that was reasonably efficient in terms of DV but also arriving as quickly as possible by reducing TOF. Both Problems 2 and 3 employed the same space triangle but used different TOFs.

For the final Problem of the semester, try to further explore the DV vs TOF trade-off. Examine one more transfer and assess whether you can improve on the DV cost for an different TOF.

(a) Focus on one of two options: (a) You can define the same space triangle and use a new TOF; OR (b) try a new space triangle and one of the TOFs Problem 2 and 3. [You cannot use a Hohmann or bielliptic transfer!]

Justify your space triangle / TOF combination. Why do you think your choices might yield a better result?

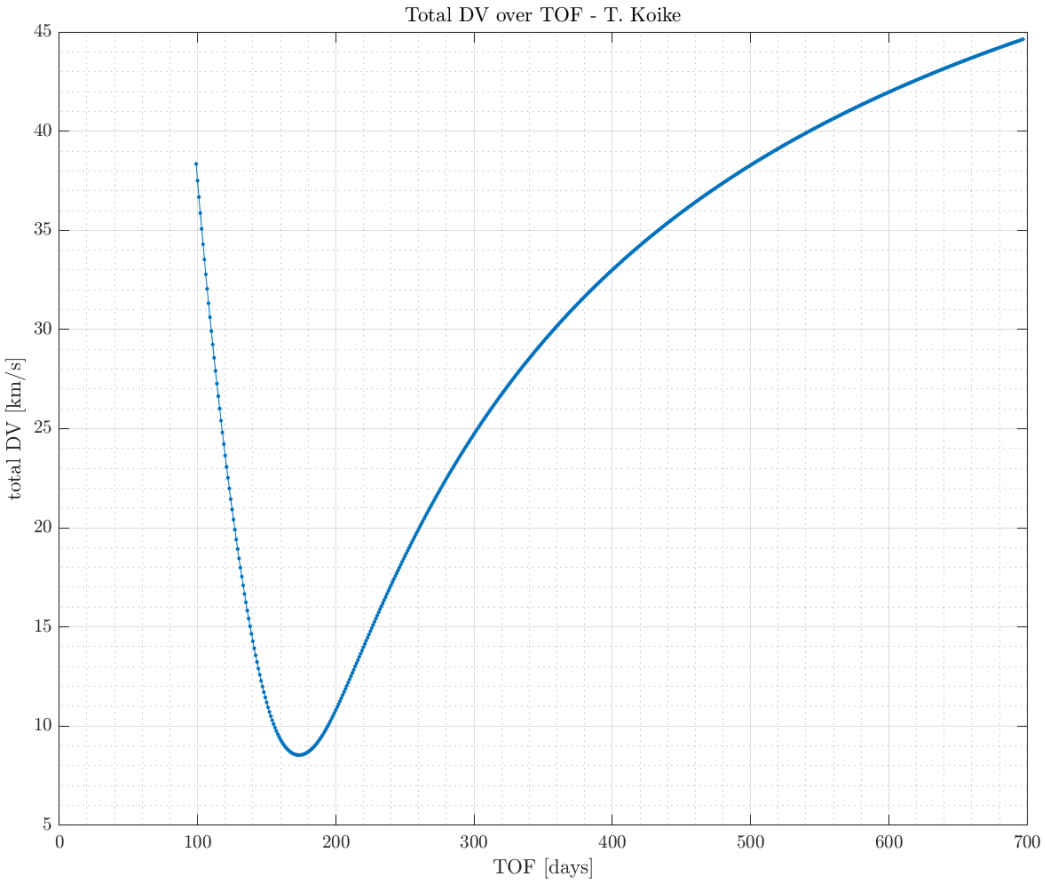
We will focus on using the same space triangle but changing the TOF to minimize the DV cost. But from Problem 3, we know that having a hyperbolic transfer orbit is unreasonable, so we will find a TOF that keeps the transfer orbit an ellipse.

With MATLAB, we create a code that iterates through all the TOFs from

$$TOF \in \left[TOF_{para} + 1,\ \ TOF_{min} \times 3\right]$$

and calculates the delta-V for each TOF. We store the TOF and delta-V data for each iteration so that we can plot the results afterwards. The code is in the Appendix.

This gives us the following plot



From the plot, we can tell that there is instance where the total DV approaches a local minimum. We can find this TOF using MATLAB.

TOF [days]	Total DV [km/s]
173	8.5319

Thus, we will use TOF of 173 days. From the plot on the previous page, we have verified that empirically that the orbit will yield a more cost-efficient solution for the mission.

(b) Produce results consistent with the results in the earlier problems. Assess and discuss your results:

Did your new combination produce 'improved' results?
Why did the DV improve? What dynamically changed about the transfer that led to improvements?

OR

Why was there not an improvement? What dynamic conditions during the transfer likely resulted in a less desirable transfer?

Since the TOF for the minimal energy condition is larger than the mission TOF, we now know that the transfer orbit type is TYPE-1A.

Now, if our orbit is a TYPE-2B, we can iterate over the Lambert TOF equation to find the optimal semi-major axis for the assumed TOF. This is computationally done using the MATLAB function '`find_lambertEllip_SMA()`' (the code is in the Appendix).

$$\mathrm{a} = 1.8976e + 8\ km$$

$$\alpha = 149.5993°$$

$$\beta = 29.2047°$$

$$\mathrm{p} = \frac{4a(s-r_1)(s-r_2)}{c^2}\sin^2\left(\frac{\alpha \pm \beta}{2}\right) = 1.7899\mathrm{e}+8\ \mathrm{km}\ \text{ or } 1.3479\mathrm{e}+8\ \mathrm{km}$$

We decide which semi-latus rectum values is valid by checking the true anomalies.

If $p = 1.3479\mathrm{e}+8$ km,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1}-1\right)\right) = +100.5959°,\quad -100.5959°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2}-1\right)\right) = +139.4041°,\quad -139.4041°\,.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep} \Rightarrow 120°$$

No combinations of the true anomaly satisfy this condition, so this p is not valid.

If $p = 1.7899\mathrm{e}+8$ km,

$$\theta^*_{dep} = arccos\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right) = +34.3923°, \quad -34.3923°$$

$$\theta^*_{arr} = \arccos\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right) = +154.3923°, \quad -154.3923°.$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep}$$

From the relationship

$$TA = \theta^*_{arr} - \theta^*_{dep} \Rightarrow 120°$$

$$\therefore \theta^*_{dep} = +34.3923°, \qquad \theta^*_{arr} = +154.3923°$$

Now we verify this by calculating the actual TOF with the eccentric anomalies. From the relationship

$$tan\frac{\theta^*}{2} = \left(\frac{1+e}{1-e}\right)^{0.5} tan\frac{E}{2}$$

$$E_{dep} = +27.2908°, \qquad E_{arr} = 147.6853°$$

Then

$$\text{TOF}_{verify} = \sqrt{\frac{a^3}{\mu_{sun}}}\left(E_{arr} - E_{dep} - e\left(sinE_{arr} - sinE_{dep}\right)\right) = 1.4947\text{e}+7 \text{ s} = 172.9 \text{ days}.$$

This is equivalent to the expected TOF of 173 days.

$$\theta^*_{dep} = +34.3923°, \qquad \theta^*_{arr} = +154.3923°$$

$$\therefore p = 1.7899e + 8\ km$$

$$e = \sqrt{1 - \frac{p}{a}} = 0.2381$$

$$\mathcal{E} = -\frac{\mu_{sun}}{2a} = -349.6936\ km^2/s^2\ .$$

$$v_{dep} = \sqrt{\mu_{sun}\left(\frac{2}{r_1} - \frac{1}{a}\right)} = 32.7852\ km/s$$

$$v_{arr} = \sqrt{\mu_{sun}\left(\frac{2}{r_2} - \frac{1}{a}\right)} = 21.5648\ km/s$$

$$\gamma_{dep} = arccos\left(\frac{\sqrt{\mu_{sun}p}}{r_1 v_{dep}}\right) = \pm 6.4143°$$

$$\gamma_{arr} = \arccos\left(\frac{\sqrt{\mu_{sun}p}}{r_1 v_{arr}}\right) = \pm 7.4672°.$$

From the true anomalies we can tell if the orbit is ascending or descending, which means that we can tell whether the flight path angles are positive or negative.

$$\therefore \gamma_{dep} = +6.4143°, \qquad \gamma_{arr} = +7.4672°.$$

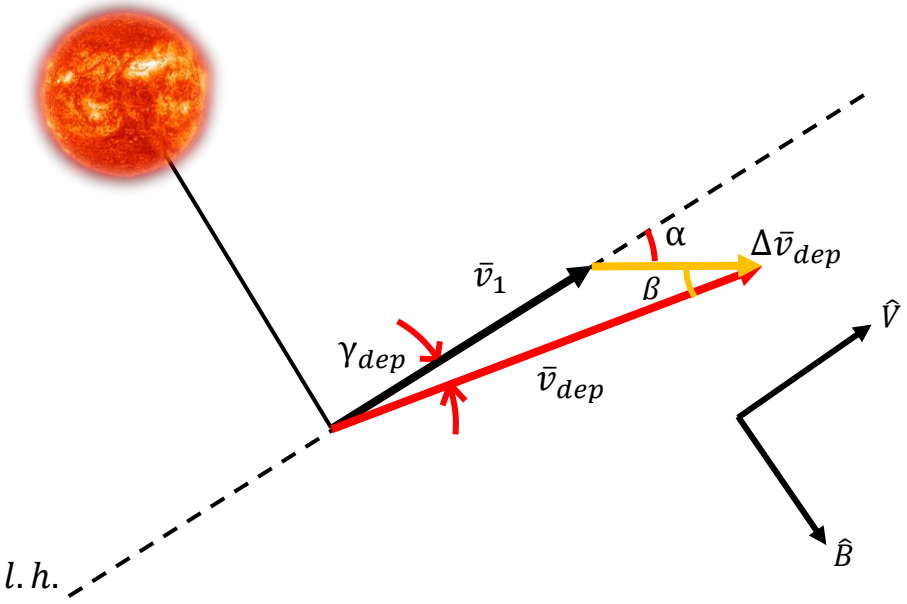
The radial distance of the periapsis and the apoapsis are

$$r_p = a(1-e) = 1.4457e + 8\ km$$

$$r_a = a(1+e) = 2.3494e + 8\ km.$$

From the calculations of the true anomalies, we can see that the difference between the true anomalies equal the transfer orbit of 120°.

At departure:



From the cosine rule,

$$\left|\Delta\bar{v}_{dep}\right| = \sqrt{v_{dep}^2 + v_1^2 - 2v_{dep}v_1 cos\gamma_{dep}} = 4.6074\ km/s\,.$$

Then, from the sine rule and cosine rule

$$= \arcsin\left(\frac{v_1}{\Delta v_{dep}} sin\gamma_{dep}\right) = 46.2355°,\ \ 133.7645°$$

$$\beta = \arccos\left(\frac{\Delta v_{dep}^2 + v_{dep}^2 - v_1^2}{2v_{dep}\Delta v_{dep}}\right) = \pm 46.2355°$$

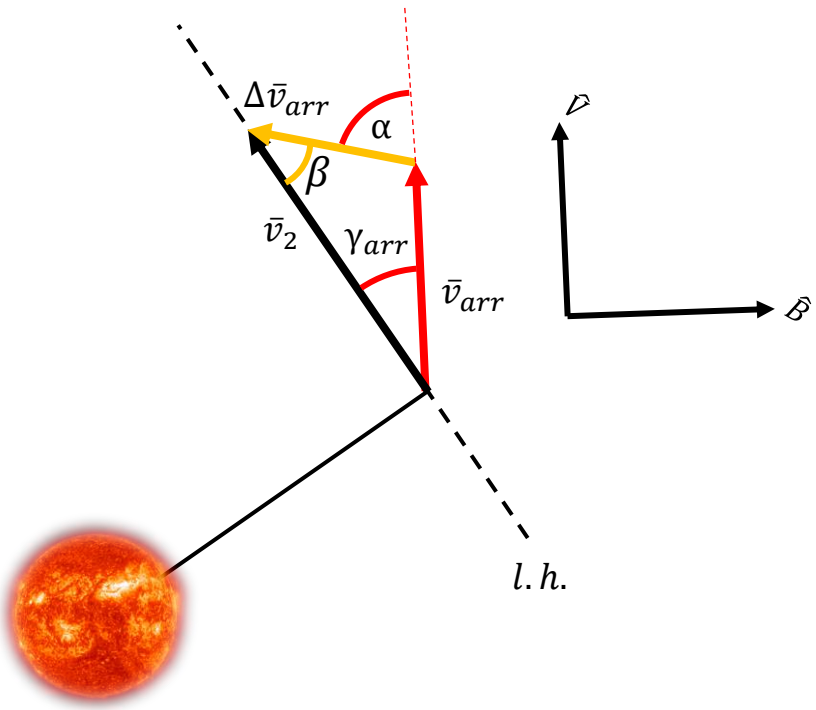
$$\therefore \beta = -34.4580°$$

$$\alpha = arcsin\left(\frac{v_1}{\Delta v_{dep}} sin\left|\gamma_{dep}\right|\right) + \left|\gamma_{dep}\right| = 52.6498°\,.$$

The delta-V can be expressed in the VNB coordinates with the α

$$\Delta\bar{v}_{dep} = \left|\Delta\bar{v}_{dep}\right|\left(cos|\alpha|\hat{V} + sin|\alpha|\left(\hat{B}\right)\right) = 2.7953\hat{V} + 3.6626\hat{B}\ \text{km/s}\,.$$

At arrival:



From the cosine rule,

$$|\Delta\bar{v}_{arr}| = \sqrt{v_{arr}^2 + v_2^2 - 2v_{arr}v_2 cos\gamma_{arr}} = 3.9244 \text{ km/s}.$$

Then, from the sine rule and cosine rule

$$\beta = \arcsin\left(\frac{v_{arr}}{\Delta v_{arr}} sin\gamma_{arr}\right) = 45.5715°,\ \ 134.4285°$$

$$\beta = arccos\left(\frac{v_2^2 + \Delta v_{arr}^2 - v_{arr}^2}{2v_2\Delta v_{arr}}\right) = \pm 45.5715°$$

$$\therefore \beta = \ 95.9950°$$

$$\alpha = \arcsin\left(\frac{v_{arr}}{\Delta v_{arr}} sin\gamma_{arr}\right) + \gamma_{arr} = \ 53.0387°.$$

The delta-V can be expressed in the VNB coordinates with the α

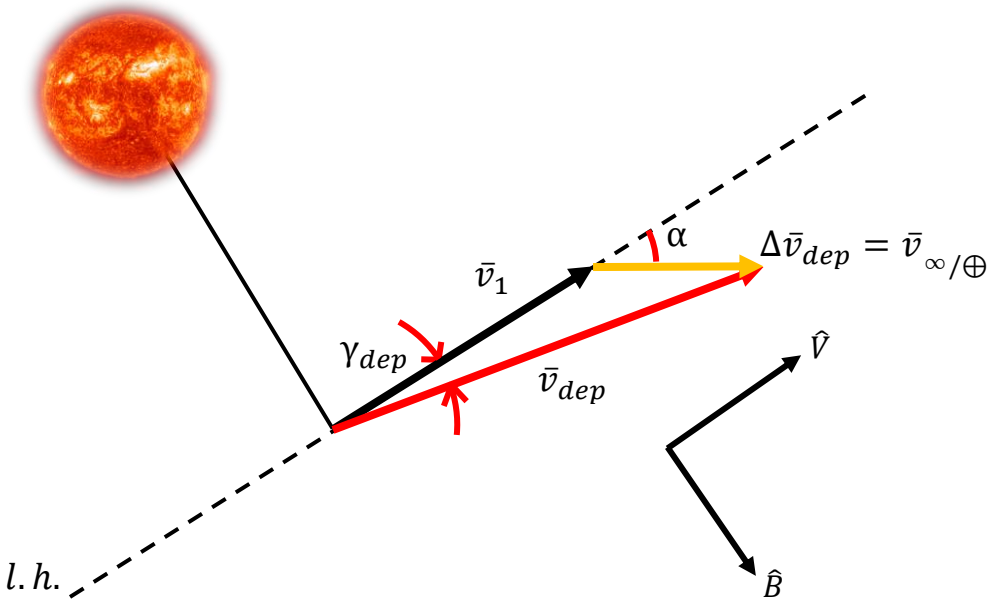
$$\Delta\bar{v}_{arr} = |\Delta\bar{v}_{arr}|\left(cos\alpha\hat{V} + sin\alpha(-\hat{B})\right) = 2.3597\hat{V} - 3.1358\hat{B} \text{ km/s}.$$

The total maneuver cost is

$$|\Delta\bar{v}_{tot}| = \left|\Delta\bar{v}_{dep}\right| + |\Delta\bar{v}_{arr}| = 8.5319\ km/s\,.$$

Now, if we include local fields…

From the diagram,

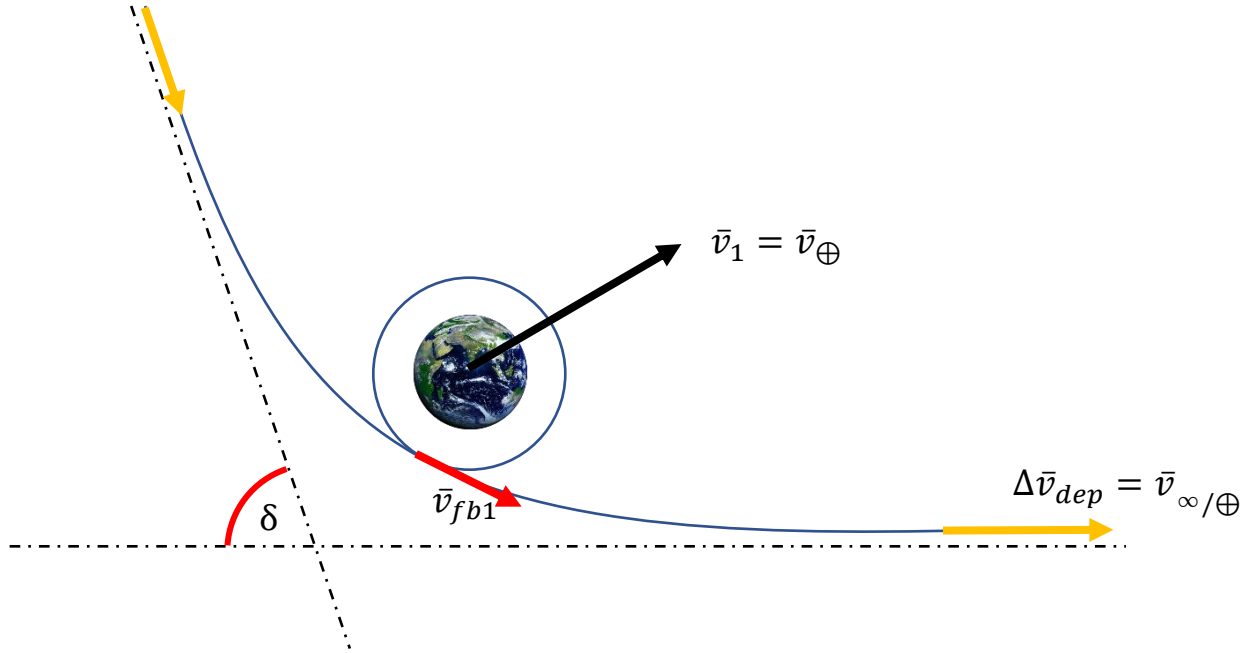


$$\bar{v}_{\infty/\oplus} = \Delta\bar{v}_{dep} = 2.7953\hat{V} + 3.6626\hat{B} \text{ km/s.}$$

$$\therefore v_{\infty/\oplus} = 4.6074\ km/s \quad .$$

To consider the local field of the Earth we have to view the departure in a geocentric perspective.

The geocentric view:



From the hyperbola characteristics

$$h = 1000$$

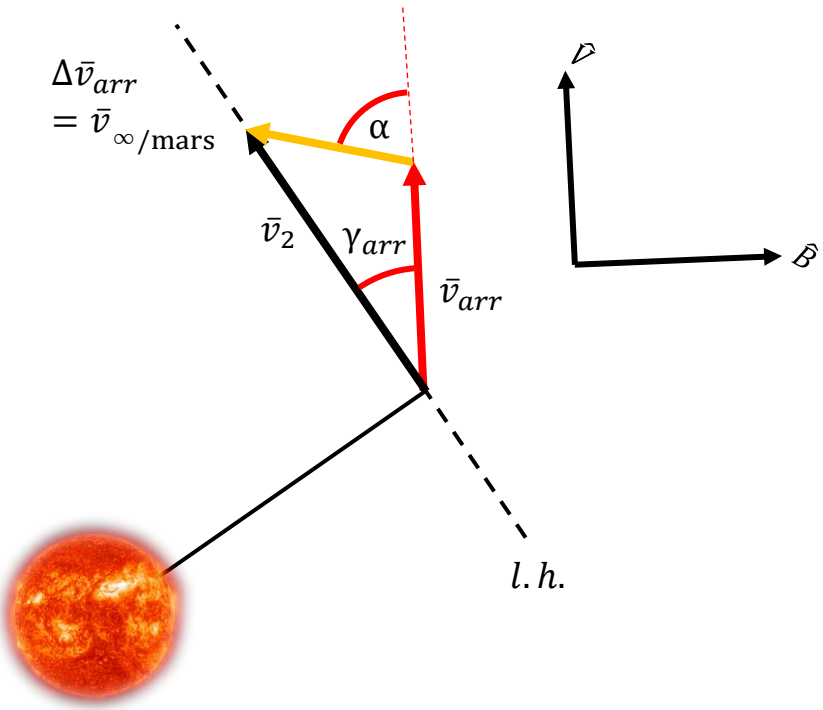
$$\xi_{\text{fb}} = \frac{\left(v_{\infty/\oplus}\right)^2}{2} = 10.6142 \ \ km^2/s^2$$

$$a_{\text{fb}} = -\frac{\mu_\oplus}{2\xi_{\text{fb}}} = -1.8777\text{e}+4 \ \text{km}$$

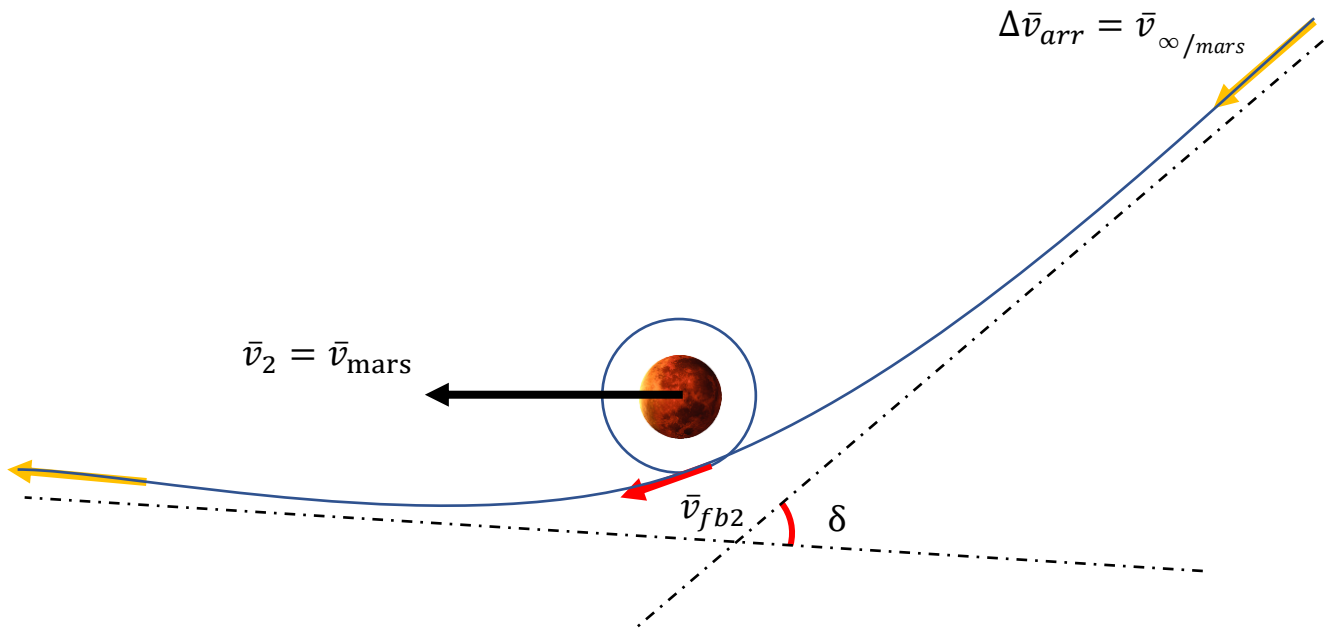
$$e_{fb} = 1 - \frac{R_\oplus + h}{a_{fb}} = 1.3929$$

$$\delta = 2arcsin\left(\frac{1}{e_{fb}}\right) = 91.7637°$$

$$v_{fb1} = \sqrt{\mu_\oplus\left(\frac{2}{R_\oplus + h} - \frac{1}{a_{fb}}\right)} = 11.3700 \ \ km/s\,.$$



The Mars centered view:



From the hyperbola characteristics

$$h = 500$$

$$\xi_{\text{fb}} = \frac{\left(v_{\infty/\text{mars}}\right)^2}{2} = 7.7005 \ km^2/s^2$$

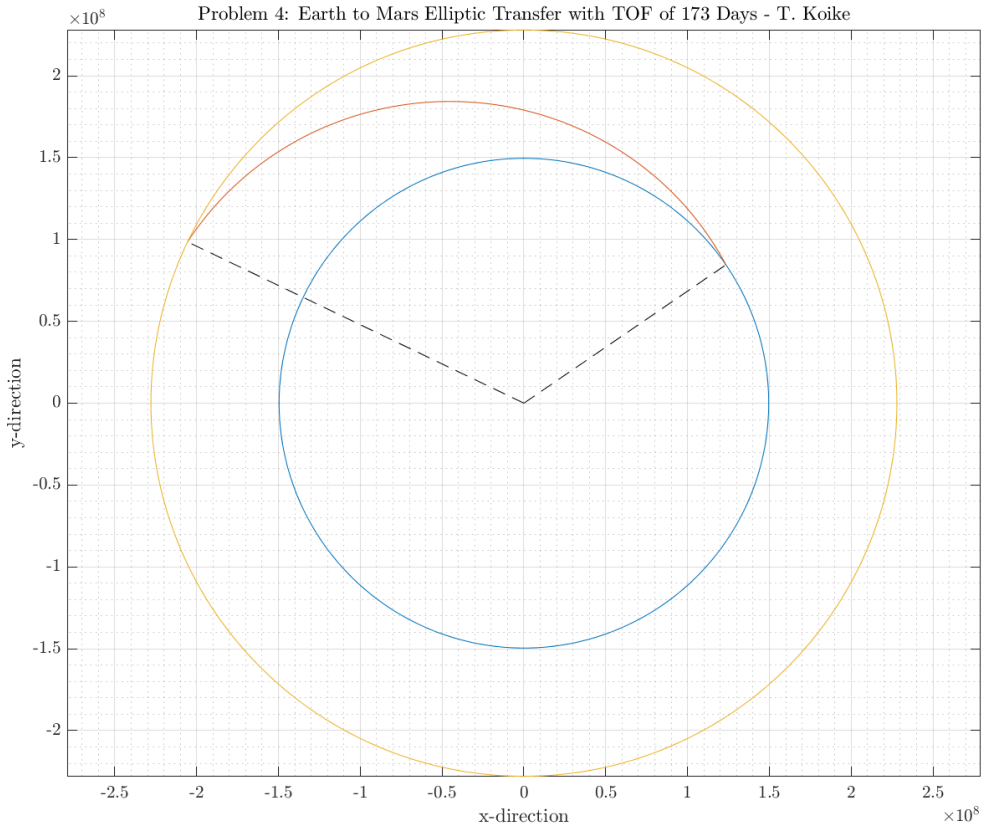
$$a_{\text{fb}} = -\frac{\mu_{\text{mars}}}{2\xi_{\text{fb}}} = -2.7809\text{e}+3 \ \text{km}$$

$$e_{fb} = 1 - \frac{R_{\text{mars}} + h}{a_{fb}} = 2.4014$$

$$\delta = 2arcsin\left(\frac{1}{e_{fb}}\right) = 49.2189°$$

$$v_{fb2} = \sqrt{\mu_{\text{mars}}\left(\frac{2}{R_{mars} + h} - \frac{1}{a_{fb}}\right)} = 6.1140 \ \text{km/s}.$$

The transfer orbit is plotted as follows



Discussion:

The new TOF of 173 days has improved the cost efficiency of the mission. Furthermore, it has reduced the flyby speeds at the vicinity of Mars (increased for the vicinity of Earth however) which makes it easier for rendezvous and other extensive missions to be conducted at Mars.

The mission has improved because we have selected a TOF that specifically minimizes the total delta-V with an empirical approach. Moreover, the transfer orbit plot in Problem 2 depicts that if we make the flight path angle for the arrival condition smaller the delta-V will decrease. Thus, making the TOF larger by a small portion would make the eccentricity slightly larger, and as a result would make the flight path angle at the arrival condition smaller.

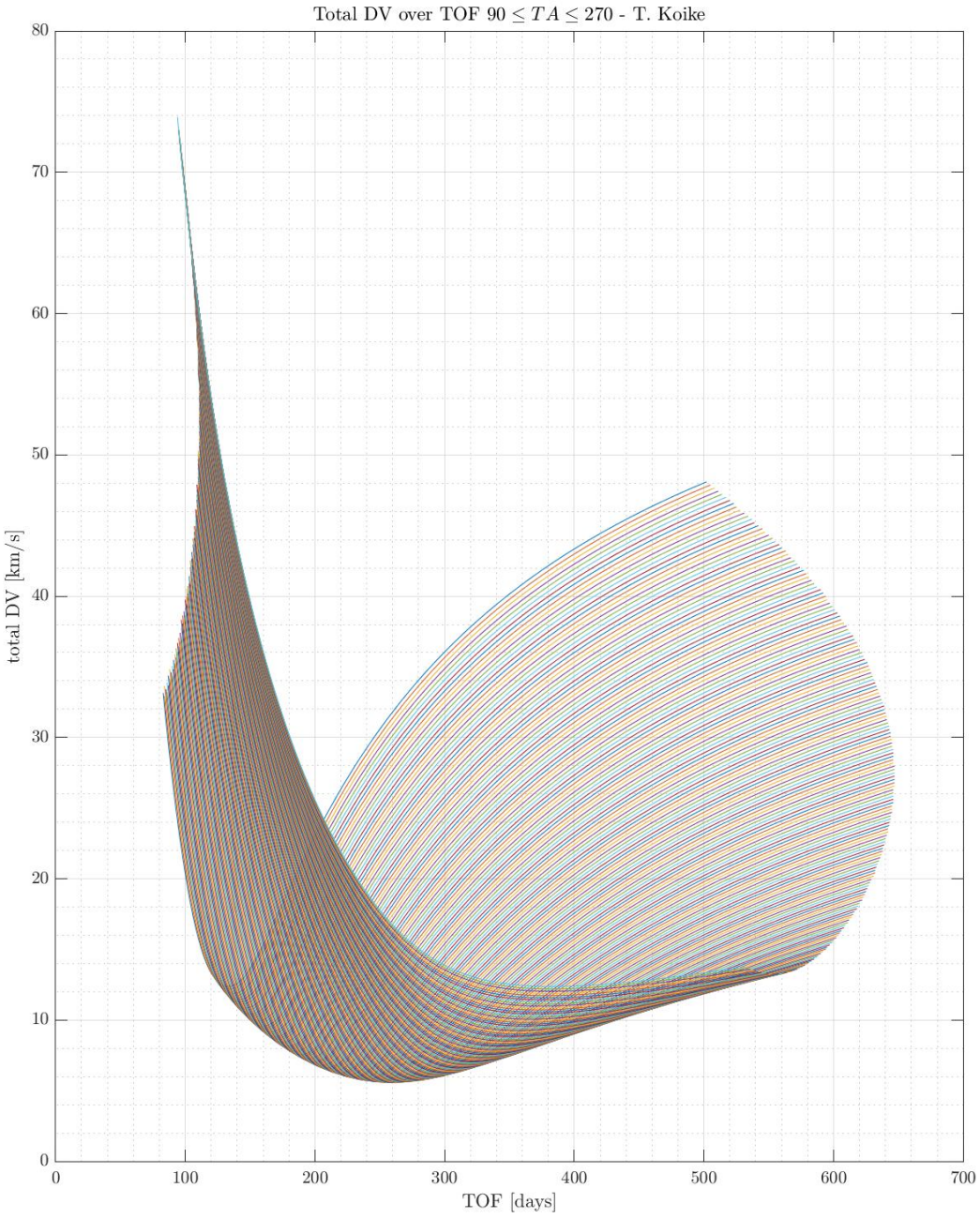
The dynamical change that led to this improvement is the flight path angle at the arrival condition. For a TOF of 173 days, the flight path angle at arrival has become approximately half the FPA in Problem 2. However, this was achieved at the expense of making the flight path angle at the departure condition slightly larger. But still both FPAs were under 8 degrees and would require a small delta-V value to reach Mars. This can be observed in the plot on the previous page.

(c) What is the next step? What combination would you try next time? Why?

The next step is to change the space triangle itself by altering the transfer angle. We do the same thing we did in Problem 4 by plotting the TOF vs total delta-V and identifying the TOF (for an elliptical transfer orbit) which is the smallest and gives the lowest total delta-V value. By doing this for all transfer angles we can find the combination of with the lowest delta-V value which leads to the most efficient transfer.

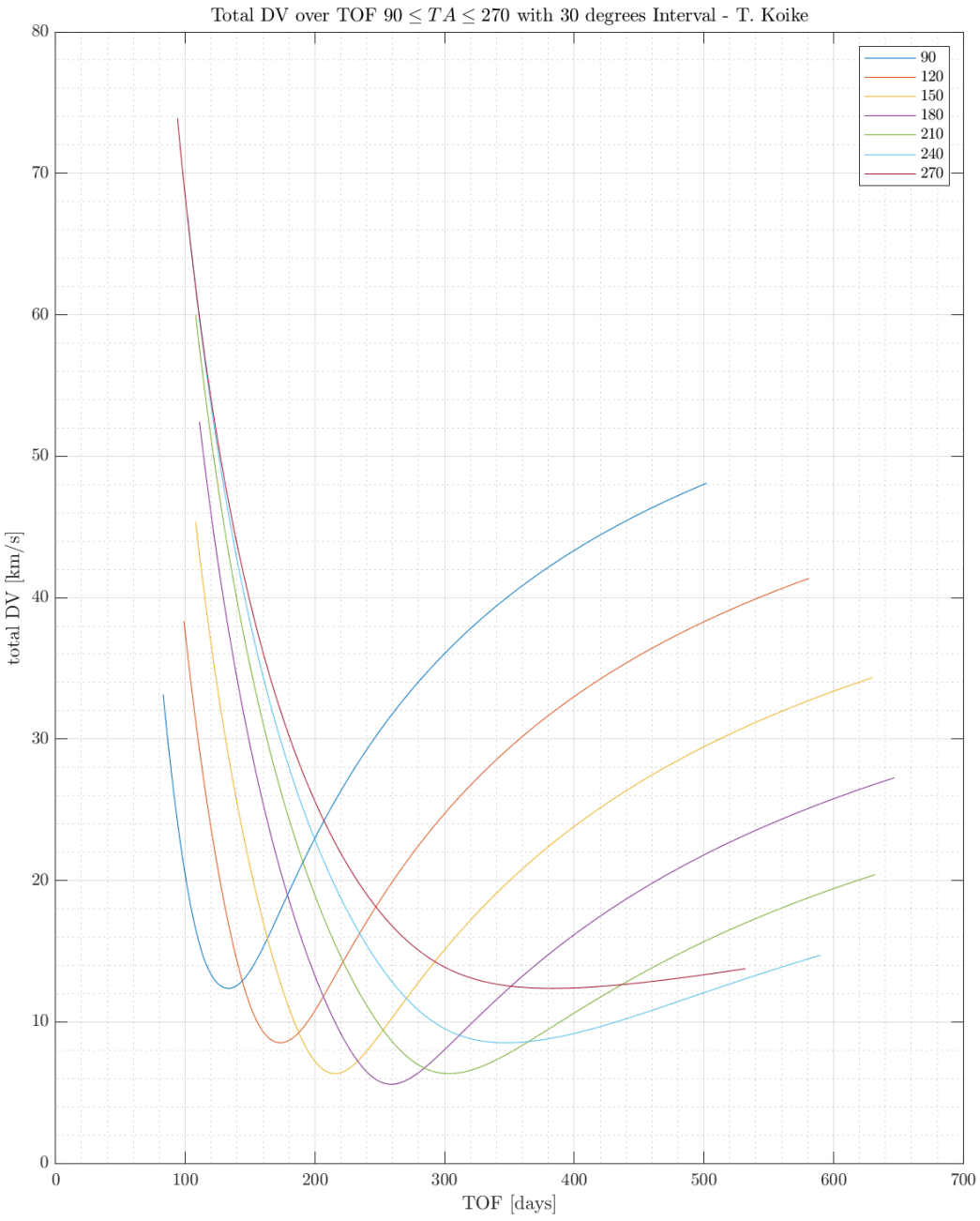
To accomplish this, we create a MATLAB code which has two for-loops. One for the transfer angles from 90° to 270° and the second loop for the TOF ranging from $(TOF_{para} + 1)days$ to $(TOF_{min} \times 2.5)days$. We obtain the TOF versus total delta-V plot for each combination of transfer angle and TOF. Then for each plot we identify the local minima and find the local minimum with the lowest TOF value. We select the lowest TOF to accommodate for the time efficiency of the mission. Moreover, we find the local minima using the "islocalmin()" command in MATLAB. Next, we store all the minimum delta-V values and corresponding TOF values for each combination and plot them over transfer angles. This will enable us to identify the transfer angle and TOF combination with the most cost-efficient as well as the time-efficient solution for the Earth-to-Mars mission.

The plots for this analysis are on the following pages and the code is in the Appendix.

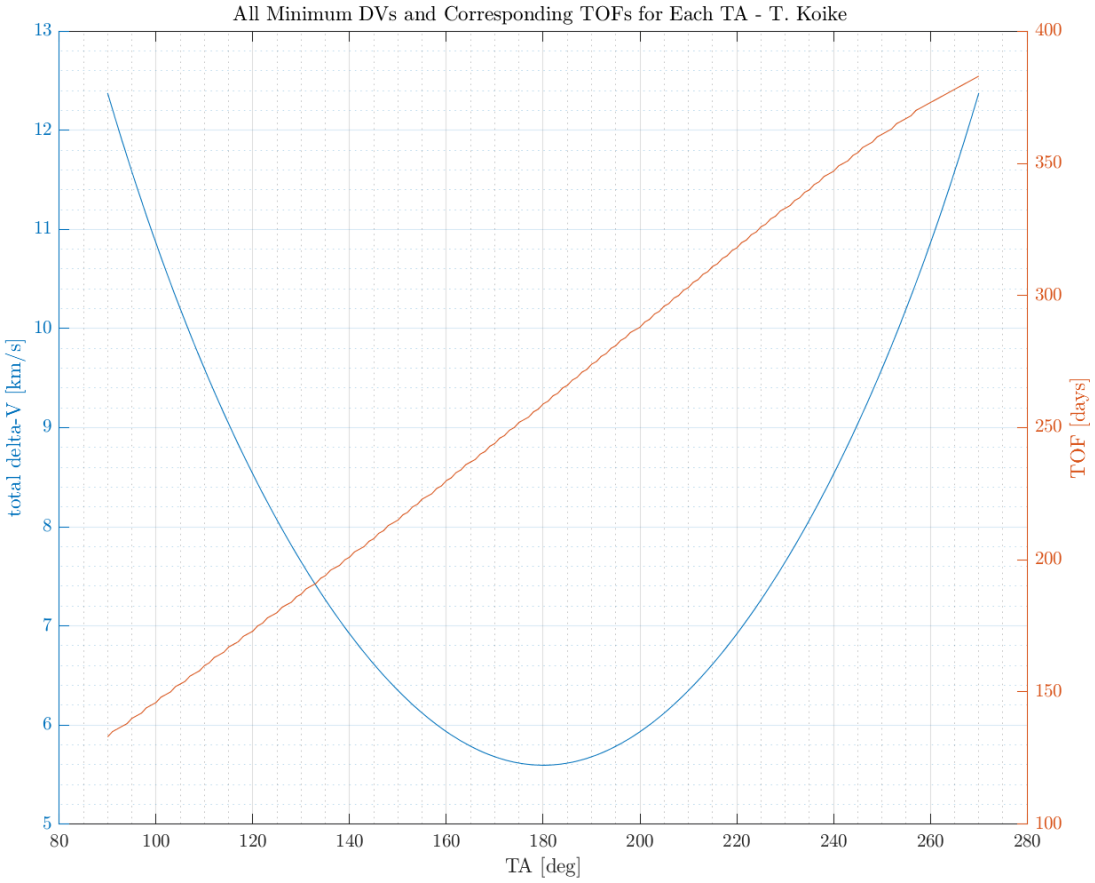


Each line represents the total delta-V vs TOF [days] for a certain transfer angle, and in this graph all the plots for transfer angle of 90° to 270° are shown.

For a clearer visualization, the following plot shows the same results but with a 30 degrees interval.



The following graph plots the minimum delta-V for the lowest TOF value for a certain TA. Additionally, the plot of the corresponding TOF value versus the certain TA is superimposed.



From the graph above we can see that we can still improve the total delta-V value while maintaining the TOF value under 200 days (200 days is selected arbitrarily as a baseline). For the next analysis, we shall select the transfer orbit that has a local minimum at TOF of 200 days and a corresponding total delta-V value of 6.984 km/s which can be identified from the graph above.

We select this because though the TOF is increased by approximately one month the total delta-V values are reduced significantly by over 1 km/s from the mission analyzed in part (a)-(b) of Problem 4. Also, we do not select TA of 180° despite that it gives the lowest total delta-V because it has a higher TOF value and our mission objective is to leverage the Lambert's equation and not a Hohmann transfer.

Appendix

MATLAB SETUP FILES

Setup_planetary_constants.m

```
%% 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
```

setup_moon_constants.m

```
%% Table of moons and dwarfs constants

function satellites = setup_moon_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)
    %}

    %% About Pluto
    % Charon
    charon.arp  = 'synchronous';
    charon.mer  = 603.60;
    charon.gp   = 102.30;
    charon.smao = 17536;
    charon.op   = 551856.7066;
    charon.eo   = 0.0022;
    charon.ioe  = 0.001;
    % Nix
    nix.arp  = NaN;
    nix.mer  = 13.0;
    nix.gp   = 0.0013;
    nix.smao = 48708;
    nix.op   = 2147558.4;
    nix.eo   = 0.0030;
    nix.ioe  = 0.195;
    % Hydra
    hydra.arp  = NaN;
    hydra.mer  = 30.5;
    hydra.gp   = 0.0065;
    hydra.smao = 64749;
    hydra.op   = 3300998.4;
    hydra.eo   = 0.0051;
    hydra.ioe  = 0.212;

    %% About Jupiter
    % Io
    io.arp  = 'synchronous';
```

```
    io.mer  = 1821.60;
    io.gp   = 5959.916;
    io.smao = 421800;
    io.op   = 152853.5047;
    io.eo   = 0.0041;
    io.ioe  = 0.036;
    % Europa
    europa.arp  = 'synchronous';
    europa.mer  = 1560.80;
    europa.gp   = 3202.739;
    europa.smao = 671100;
    europa.op   = 306822.0384;
    europa.eo   = 0.0094;
    europa.ioe  = 0.466;
    % Ganymede
    ganymede.arp  = 'synchronous';
    ganymede.mer  = 2631.20;
    ganymede.gp   = 9887.834;
    ganymede.smao = 1070400;
    ganymede.op   = 618153.3757;
    ganymede.eo   = 0.0013;
    ganymede.ioe  = 0.177;
    % Callisto
    callisto.arp  = 'synchronous';
    callisto.mer  = 2410.30;
    callisto.gp   = 7179.289;
    callisto.smao = 1882700;
    callisto.op   = 1441931.19;
    callisto.eo   = 0.0074;
    callisto.ioe  = 0.192;

    %% About Saturn
    % Titan
    titan.arp  = 'synchronous';
    titan.mer  = 2574.730;
    titan.gp   = 8978.1382;
    titan.smao = 1221865;
    titan.op   = 1377648;
    titan.eo   = 0.0288;
    titan.ioe  = 0.312;

    %% About Uranus
    % Titania
    titania.arp  = 'assumed synchronous';
    titania.mer  = 788.90;
    titania.gp   = 228.2;
    titania.smao = 436300;
    titania.op   = 752218.6176;
    titania.eo   = 0.0011;
    titania.ioe  = 0.079;

    %% Main belt asteroid
    % Ceres
    ceres.arp  = 2.6448030;
    ceres.mer  = 476.20;
    ceres.gp   = 63.2;
    ceres.smao = 413690604;
    ceres.op   = 145238090.6;
    ceres.eo   = 0.0757972598;
    ceres.ioe  = 10.593981;
```

```
    %% About Mars
    % Phobos
    phobos.arp  = 'synchronous';
    phobos.mer  = 11.10;
    phobos.gp   = 0.0007112;
    phobos.smao = 9376;
    phobos.op   = 27552.96;
    phobos.eo   = 0.01510000;
    phobos.ioe  = 1.075000;
    % Deimos
    deimos.arp  = 'synchronous';
    deimos.mer  = 6.20;
    deimos.gp   = 0.0000985;
    deimos.smao = 23458;
    deimos.op   = 109071.36;
    deimos.eo   = 0.0002;
    deimos.ioe  = 1.78800;
    % Triton
    triton.arp  = 'synchronous';
    triton.mer  = 1350.00;
    triton.gp   = 1427.598;
    triton.smao = 354759;
    triton.op   = 0.5877 * 60 * 60 * 24;
    triton.eo   = 0.000016;
    triton.ioe  = 156.86500;

    % Output of Function
    satellites.charon = charon;
    satellites.nix = nix;
    satellites.hydra = hydra;
    satellites.io = io;
    satellites.europa = europa;
    satellites.ganymede = ganymede;
    satellites.callisto = callisto;
    satellites.titan = titan;
    satellites.titania = titania;
    satellites.ceres = ceres;
    satellites.phobos = phobos;
    satellites.deimos = deimos;
    satellites.triton = triton;

end
```

MATLAB MAIN FILES

aae532_hw10_p1.mlx

```
% AAE 532 HW 10 Problem 1
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps10';
set(groot, 'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% Set constants
planet_consts = setup_planetary_constants();  % Function that sets up all the constants
in the table
mars = planet_consts.mars;  % structure of mars
G = 6.6743015e-20;  % Gravitational constant [km^3/kg/s^2]
moon_consts = setup_moon_constants();  % Function that sets up all the moon/satellite
constants
phobos = moon_consts.phobos;  % structure of phobos
deimos = moon_consts.deimos;  % structure of deimos

% (a)
% Assumptions
TrA = 240;  % transfer angle [deg]
TOF = 15 * 60 * 60;  % TOF [s]

% Determine the TYPE of the transfer orbit
mu_mars = mars.gp  % gravitational parameter of Mars
R_mars = mars.mer  % radius of Mars
r1 = phobos.smao  % the first leg of the space triangle equal to the semi-major axis of
Phobos
r2 = deimos.smao  % the second leg of the space triangle equal to the semi-major axis of
Deimos
v1 = sqrt(mu_mars / r1)  % initial velocity in the orbit of Phobos
v2 = sqrt(mu_mars / r2)  % final velocity in the orbit of Deimos
c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(360 - TrA))  % chord length of the space triangle
s = (r1 + r2 + c) / 2  % semi-perimeter
a_min = s/2  % the minimum possible semi-major axis of the transfer orbit
alpha_o = 2*asin_dbval(sqrt(s / 2 / a_min), "deg")  % base alpha parameter
beta_o = 2*asin_dbval(sqrt((s - c) / 2 / a_min), "deg")  % base beta parameter
alpha_o = alpha_o(1)
beta_o = beta_o(beta_o == min(beta_o))

% Calculate the minimum TOF using lambert equation
TOF_min = lambert_ellip(deg2rad(alpha_o), deg2rad(beta_o), mu_mars, "TOF", "2A", a_min)
TOF_min_hrs = TOF_min / 60 / 60

% The orbit is TYPE-2B
[a, alpha, beta, fval] = find_lambertEllip_SMA(s, c, TOF, mu_mars, a_min, "2B")

[p, e, TA_dep, TA_arr, p_db, TA_dep_db, TA_arr_db] = find_lambertX_ellip(a, alpha, ...
    beta, TrA, s, c, r1, r2, mu_mars, 15, "hour")

TA_dep
TA_arr

% Calculate energy
En = -mu_mars / 2 / a
```

```
v_dep = vis_viva(r1, a, mu_mars)
v_arr = vis_viva(r2, a, mu_mars)
FPA_dep = acos_dbval(sqrt(mu_mars * p)/r1 / v_dep, "deg")
FPA_arr = acos_dbval(sqrt(mu_mars * p)/r2 / v_arr, "deg")
FPA_dep = FPA_dep(FPA_dep < 0)
FPA_arr = FPA_arr(FPA_arr < 0)

rp = a * (1 - e)
ra = a * (1 + e)

% (b)
% Departure
Dv_dep = sqrt(v_dep^2 + v1^2 - 2 * v_dep * v1 * cosd(FPA_dep))
beta1 = asin_dbval(v1 / Dv_dep * sind(FPA_dep), "deg")
beta2 = acos_dbval((Dv_dep^2 + v_dep^2 - v1^2)/2/v_dep/Dv_dep, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_dep = beta + FPA_dep
Dv_dep_vec.vnb = Dv_dep * [cosd(alpha_dep), sind(alpha_dep)]

% Arrival
Dv_arr = sqrt(v_arr^2 + v2^2 - 2 * v_arr * v2 * cosd(FPA_arr))
beta1 = asin_dbval(v_arr / Dv_arr * sind(FPA_arr), "deg")
beta2 = acos_dbval((v2^2 + Dv_arr^2 - v_arr^2)/2/v2/Dv_arr, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_arr =  beta + FPA_arr
Dv_arr_vec.vnb = Dv_arr * [cosd(alpha_arr), sind(abs(alpha_arr))]

% Total
Dv_tot = Dv_dep + Dv_arr

% (d)
% Phase angle
n_deimos = sqrt(mu_mars / r2^3);
n_phobos = sqrt(mu_mars / r1^3);
phi = rad2deg(deg2rad(TrA) -n_deimos * TOF)
% the synodic period
s = 2* pi / (n_phobos - n_deimos)
s_hrs = s / 60 / 60
% The periods of each orbit
IP_phobos = 1 / n_phobos
IP_phobos_hrs = IP_phobos / 60 / 60
IP_deimos = 1 / n_deimos
IP_deimos_hrs = IP_deimos / 60 / 60

% Comparison
x1 = s / IP_phobos
x2 = s / IP_deimos
```

aae532_hw10_p2.mlx

```
% AAE 532 HW 10 Problem 2
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps10';
set(groot, 'defaulttextinterpreter','latex');
```

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

% Set constants
planet_consts = setup_planetary_constants();  % Function that sets up all the constants
in the table
mars = planet_consts.mars;  % structure of mars
earth = planet_consts.earth;  % structure of earth
sun = planet_consts.sun;  % structure of sun
G = 6.6743015e-20;  % Gravitational constant [km^3/kg/s^2]

% (a)
% Assumptions
% ** TYPE-1 transfer
TrA = 120;  % transfer angle [deg]
TOF_days = 160;  % [days]
TOF = TOF_days * 60 * 60 * 24;  % TOF [s]

% Determine the TYPE of the transfer orbit
mu_sun = sun.gp  % gravitational parameter of sun
mu_earth = earth.gp  % gravitational parameter of earth
mu_mars = mars.gp  % gravitational parameter of mars
R_mars = mars.mer  % radius of Mars
R_earth = earth.mer  % radius of earth

r1 = earth.smao  % the first leg of the space triangle equal to the semi-major axis of
earth
r2 = mars.smao  % the second leg of the space triangle equal to the semi-major axis of
mars
v1 = sqrt(mu_sun / r1)  % initial velocity in the orbit of earth
v2 = sqrt(mu_sun / r2)  % final velocity in the orbit of mars

c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(TrA))  % chord length of the space triangle
s = (r1 + r2 + c) / 2  % semi-perimeter

% Check if the transfer orbit is elliptical or hyperbolic
TOF_para = lambertTOF_para(mu_sun, s, c, "1")
TOF_para_days = TOF_para / 60 / 60 / 24

% (b)
% Check which type of elliptical transfer
a_min = s/2  % the minimum possible semi-major axis of the transfer orbit
alpha_o = 2*asin_dbval(sqrt(s / 2 / a_min), "deg")  % base alpha parameter
beta_o = 2*asin_dbval(sqrt((s - c) / 2 / a_min), "deg")  % base beta parameter
alpha_o = alpha_o(1)
beta_o = beta_o(beta_o == min(beta_o))

% Calculate the minimum TOF using lambert equation
TOF_min = lambert_ellip(deg2rad(alpha_o), deg2rad(beta_o), mu_sun, "TOF", "1A", a_min)
TOF_min_days = TOF_min / 60 / 60 / 24

% The orbit is TYPE-1A
[a, alpha, beta, fval] = find_lambertEllip_SMA(s, c, TOF, mu_sun, a_min, "1A")
rad2deg(alpha)
rad2deg(beta)

[p, e, TA_dep, TA_arr, p_db, TA_dep_db, TA_arr_db] = find_lambertX_ellip(a, alpha,
beta, ...
```

```
    TrA, s, c, r1, r2)

E1 = T2E_anomaly(e, TA_dep, "deg")
E2 = T2E_anomaly(e, TA_arr, "deg")
t1 = sqrt(a^3 / mu_sun) * (deg2rad(E1) - e * sind(E1))
t2 = sqrt(a^3 / mu_sun) * (deg2rad(E2) - e * sind(E2))
T = t2 - t1
IP = 2 * pi * sqrt(a^3 / mu_sun)
if round((IP - T) / 60 / 60) == (TOF / 60 / 60)
    TA_dep = -TA_dep;
    TA_arr = -TA_arr;
end
TA_dep
TA_arr

% Calculate energy
En = -mu_sun / 2 / a

v_dep = vis_viva(r1, a, mu_sun)
v_arr = vis_viva(r2, a, mu_sun)
FPA_dep = acos_dbval(sqrt(mu_sun * p)/r1 / v_dep, "deg")
FPA_arr = acos_dbval(sqrt(mu_sun * p)/r2 / v_arr, "deg")
FPA_dep = FPA_dep(FPA_dep > 0)
FPA_arr = FPA_arr(FPA_arr > 0)
rp = a * (1 - e)
ra = a * (1 + e)

% (c)
% Departure
Dv_dep = sqrt(v_dep^2 + v1^2 - 2 * v_dep * v1 * cosd(FPA_dep))
beta1 = asin_dbval(v1 / Dv_dep * sind(FPA_dep), "deg")
beta2 = acos_dbval((Dv_dep^2 + v_dep^2 - v1^2)/2/v_dep/Dv_dep, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_dep = beta + FPA_dep
Dv_dep_vec.vnb = Dv_dep * [cosd(abs(alpha_dep)), sind(abs(alpha_dep))]

% Arrival
Dv_arr = sqrt(v_arr^2 + v2^2 - 2 * v_arr * v2 * cosd(FPA_arr))
beta1 = asin_dbval(v_arr / Dv_arr * sind(FPA_arr), "deg")
beta2 = acos_dbval((v2^2 + Dv_arr^2 - v_arr^2)/2/v2/Dv_arr, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_arr =  beta + FPA_arr
Dv_arr_vec.vnb = Dv_arr * [cosd(alpha_arr), -sind(alpha_arr)]

% Total
Dv_tot = Dv_dep + Dv_arr

% (e)
v_inf_earth = Dv_dep;
h = 1000;
En_fb = v_inf_earth^2 / 2
a_fb = -mu_earth / 2 / En_fb
e_fb = 1 - (earth.mer + h) / a_fb
delta = 2 * asind(1 / e_fb)
v_fb1 = vis_viva(earth.mer + h, a_fb, mu_earth)

% (f)
v_inf_mars = Dv_arr;
```

```
h = 500;
En_fb = v_inf_mars^2 / 2
a_fb = -mu_mars / 2 / En_fb
e_fb = 1 - (mars.mer + h) / a_fb
delta = 2 * asind(1 / e_fb)
v_fb2 = vis_viva(mars.mer + h, a_fb, mu_mars)
```

aae532_hw10_p3.mlx

```
% AAE 532 HW 10 Problem 3
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps10';
set(groot, 'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% Set constants
planet_consts = setup_planetary_constants();  % Function that sets up all the constants
in the table
mars = planet_consts.mars;  % structure of mars
earth = planet_consts.earth;  % structure of earth
sun = planet_consts.sun;  % structure of sun
G = 6.6743015e-20;  % Gravitational constant [km^3/kg/s^2]

% (a)
% Assumptions
% ** TYPE-1 transfer
TrA = 120;  % transfer angle [deg]
TOF_days = 92;  % [days]
TOF = TOF_days * 60 * 60 * 24;  % TOF [s]

% Determine the TYPE of the transfer orbit
mu_sun = sun.gp  % gravitational parameter of sun
mu_earth = earth.gp  % gravitational parameter of earth
mu_mars = mars.gp  % gravitational parameter of mars
R_mars = mars.mer  % radius of Mars
R_earth = earth.mer  % radius of earth

r1 = earth.smao  % the first leg of the space triangle equal to the semi-major axis of
earth
r2 = mars.smao  % the second leg of the space triangle equal to the semi-major axis of
mars
v1 = sqrt(mu_sun / r1)  % initial velocity in the orbit of earth
v2 = sqrt(mu_sun / r2)  % final velocity in the orbit of mars

c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(TrA))  % chord length of the space triangle
s = (r1 + r2 + c) / 2  % semi-perimeter

% (b)
% The orbit is TYPE-1H
a_min = s/2  % the minimum possible semi-major axis of the transfer orbit
[a, alpha, beta, fval] = find_lambertHyp_SMA(s, c, TOF, mu_sun, a_min, "1H")
rad2deg(alpha)
rad2deg(beta)
```

```
[p, p_db, e, TA_dep, TA_arr, TA_dep_db, TA_arr_db] = find_lambertX_hyp(a, alpha,
beta, ...
    TrA, s, c, r1, r2)

H1 = T2H_anomaly(e, TA_dep, "deg")
H2 = T2H_anomaly(e, TA_arr, "deg")
t1 = sqrt(abs(a)^3 / mu_sun) * (e * sinh(deg2rad(H1)) - deg2rad(H1))
t2 = sqrt(abs(a)^3 / mu_sun) * (e * sinh(deg2rad(H2)) - deg2rad(H2))
T = t2 - t1
if ~(round(T / 60 / 60 / 24) == (TOF / 60 / 60 / 24))
    error("TA is not the difference of true anomalies")
end
TA_dep
TA_arr

% Calculate energy
En = -mu_sun / 2 / a

v_dep = vis_viva(r1, a, mu_sun)
v_arr = vis_viva(r2, a, mu_sun)
FPA_dep = acos_dbval(sqrt(mu_sun * p)/r1 / v_dep, "deg")
FPA_arr = acos_dbval(sqrt(mu_sun * p)/r2 / v_arr, "deg")
FPA_dep = FPA_dep(FPA_dep < 0)
FPA_arr = FPA_arr(FPA_arr > 0)

rp = a * (1 - e)

% (c)
% Departure
Dv_dep = sqrt(v_dep^2 + v1^2 - 2 * v_dep * v1 * cosd(FPA_dep))
beta1 = asin_dbval(v1 / Dv_dep * sind(FPA_dep), "deg")
beta2 = acos_dbval((Dv_dep^2 + v_dep^2 - v1^2)/2/v_dep/Dv_dep, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_dep = beta + FPA_dep
Dv_dep_vec.vnb = Dv_dep * [cosd(alpha_dep), sind(alpha_dep)]

% Arrival
Dv_arr = sqrt(v_arr^2 + v2^2 - 2 * v_arr * v2 * cosd(FPA_arr))
beta1 = asin_dbval(v_arr / Dv_arr * sind(FPA_arr), "deg")
beta2 = acos_dbval((v2^2 + Dv_arr^2 - v_arr^2)/2/v2/Dv_arr, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_arr =  beta + FPA_arr
Dv_arr_vec.vnb = Dv_arr * [cosd(alpha_arr), -sind(alpha_arr)]

% Total
Dv_tot = Dv_dep + Dv_arr

% (d)
% Plotting

% Earth Orbit
angles = 0:360;
X_earth = earth.smao * cosd(angles); Y_earth = earth.smao * sind(angles);

% Transfer Orbit
angles = TA_dep:0.0001:TA_arr
R_x = p ./ (1 + e * cosd(angles));
X_x = R_x .* cosd(angles); Y_x = R_x .* sind(angles);
```

```
% Mars Orbit
angles = 0:360;
X_mars = mars.smao * cosd(angles); Y_mars = mars.smao * sind(angles);

fig = figure("Renderer","painters","Position",[10 10 900 700]);
plot(X_earth, Y_earth)
hold on; grid on; grid minor; box on; axis equal;
plot(X_x, Y_x)
plot(X_mars, Y_mars)
plot([0, X_x(1)], [0, Y_x(1)], '--k')
plot([0, X_x(end)], [0, Y_x(end)], '--k')
hold off
title('Problem 3: Earth to Mars Hyperbolic Transfer - T. Koike')
xlabel('x-direction')
ylabel('y-direction')
saveas(fig, fullfile(fdir, 'p3_transfer.png'));

% (e)
v_inf_earth = Dv_dep;
h = 1000;
En_fb = v_inf_earth^2 / 2
a_fb = -mu_earth / 2 / En_fb
e_fb = 1 - (earth.mer + h) / a_fb
delta = 2 * asind(1 / e_fb)
v_fb1 = vis_viva(earth.mer + h, a_fb, mu_earth)

% (f)
v_inf_mars = Dv_arr;
h = 500;
En_fb = v_inf_mars^2 / 2
a_fb = -mu_mars / 2 / En_fb
e_fb = 1 - (mars.mer + h) / a_fb
delta = 2 * asind(1 / e_fb)
v_fb2 = vis_viva(mars.mer + h, a_fb, mu_mars)
```

aae532_hw10_p4.mlx

```
% AAE 532 HW 10 Problem 3
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps10';
set(groot, 'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% Set constants
planet_consts = setup_planetary_constants();  % Function that sets up all the constants
in the table
mars = planet_consts.mars;  % structure of mars
earth = planet_consts.earth;  % structure of earth
sun = planet_consts.sun;  % structure of sun
G = 6.6743015e-20;  % Gravitational constant [km^3/kg/s^2]

mu_sun = sun.gp  % gravitational parameter of sun
mu_earth = earth.gp  % gravitational parameter of earth
```

```
mu_mars = mars.gp  % gravitational parameter of mars
R_mars = mars.mer  % radius of Mars
R_earth = earth.mer  % radius of earth

r1 = earth.smao  % the first leg of the space triangle equal to the semi-major axis of
earth
r2 = mars.smao  % the second leg of the space triangle equal to the semi-major axis of
mars
v1 = sqrt(mu_sun / r1)  % initial velocity in the orbit of earth
v2 = sqrt(mu_sun / r2)  % final velocity in the orbit of mars

TrA = 120;  % transfer angle [deg]
c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(TrA))  % chord length of the space triangle
s = (r1 + r2 + c) / 2  % semi-perimeter

% TOF for parabolic orbit
TOF_para = lambertTOF_para(mu_sun, s, c, "1")
TOF_para_days = TOF_para / 60 / 60 / 24

% Check which type of elliptical transfer
a_min = s/2  % the minimum possible semi-major axis of the transfer orbit
alpha_o = 2*asin(sqrt(s / 2 / a_min))  % base alpha parameter
beta_o = 2*asin(sqrt((s - c) / 2 / a_min))  % base beta parameter

% Calculate the minimum TOF using lambert equation
TOF_min = lambert_ellip(alpha_o, beta_o, mu_sun, "TOF", "1A", a_min)
TOF_min_days = TOF_min / 60 / 60 / 24

% Preallocate total DV vector
Dv_tot_array = [];
% For loop to find the optimal TOF for given space triangle
TOF_days_array = round(TOF_para_days)+1:TOF_min_days*3;
for TOF_days = TOF_days_array
    revert = 0;
    TOF = TOF_days * 60 * 60 * 24;  % TOF [s]
    if TOF > TOF_min
        type = "1B";
    else
        type = "1A";
    end
    % Find the SMA corresponding to the TOF
    [a, alpha, beta, fval] = find_lambertEllip_SMA(s, c, TOF, mu_sun, a_min, type);

    [p, e, TA_dep, TA_arr, p_db, TA_dep_db, TA_arr_db] = find_lambertX_ellip(a,
alpha, ...
    beta, TrA, s, c, r1, r2, mu_sun, TOF_days, "day");

    % Calculate energy
    En = -mu_sun / 2 / a;
    % Velocity and FPA
    v_dep = vis_viva(r1, a, mu_sun);
    v_arr = vis_viva(r2, a, mu_sun);
    FPA_dep = acos_dbval(sqrt(mu_sun * p)/r1 / v_dep, "deg");
    FPA_arr = acos_dbval(sqrt(mu_sun * p)/r2 / v_arr, "deg");
    if TA_dep > 0
        FPA_dep = FPA_dep(FPA_dep > 0);
    else
        FPA_dep = FPA_dep(FPA_dep < 0);
```

```
    end
    if TA_arr > 0
        FPA_arr = FPA_arr(FPA_arr > 0);
    else
        FPA_arr = FPA_arr(FPA_arr < 0);
    end


    % Departure
    Dv_dep = sqrt(v_dep^2 + v1^2 - 2 * v_dep * v1 * cosd(FPA_dep));
    % Arrival
    Dv_arr = sqrt(v_arr^2 + v2^2 - 2 * v_arr * v2 * cosd(FPA_arr));
    % Total
    Dv_tot = Dv_dep + Dv_arr;
    if isempty(Dv_tot)
        Dv_tot_array = [Dv_tot_array, NaN];
    else
        Dv_tot_array = [Dv_tot_array, Dv_tot];
    end
end

fig = figure("Renderer","painters","Position",[10 10 900 700]);
plot(TOF_days_array, Dv_tot_array, '.-')
grid on; grid minor; box on;
title('Total DV over TOF - T. Koike')
xlabel('TOF [days]')
ylabel('total DV [km/s]')
saveas(fig, fullfile(fdir, "p4_DV_TOF.png"));

% Find instance of minimal Dv
[Dv_tot_array_sort, idx] = sort(Dv_tot_array);
TOF_days_array_sort = TOF_days_array(idx);

% Select TOF of 173 days
% Assumptions
% ** TYPE-1 transfer
TOF_days = 173;  % [days]
TOF = TOF_days * 60 * 60 * 24;  % TOF [s]

% The orbit is TYPE-1A
[a, alpha, beta, fval] = find_lambertEllip_SMA(s, c, TOF, mu_sun, a_min, "1A")
rad2deg(alpha)
rad2deg(beta)

[p, e, TA_dep, TA_arr, p_db, TA_dep_db, TA_arr_db] = find_lambertX_ellip(a, alpha,
beta, ...
    TrA, s, c, r1, r2, mu_sun, TOF_days, "day")

% Calculate energy
En = -mu_sun / 2 / a

v_dep = vis_viva(r1, a, mu_sun)
v_arr = vis_viva(r2, a, mu_sun)
FPA_dep = acos_dbval(sqrt(mu_sun * p)/r1 / v_dep, "deg")
FPA_arr = acos_dbval(sqrt(mu_sun * p)/r2 / v_arr, "deg")
FPA_dep = FPA_dep(FPA_dep > 0)
FPA_arr = FPA_arr(FPA_arr > 0)
```

```
rp = a * (1 - e)
ra = a * (1 + e)

% (c)
% Departure
Dv_dep = sqrt(v_dep^2 + v1^2 - 2 * v_dep * v1 * cosd(FPA_dep))
beta1 = asin_dbval(v1 / Dv_dep * sind(FPA_dep), "deg")
beta2 = acos_dbval((Dv_dep^2 + v_dep^2 - v1^2)/2/v_dep/Dv_dep, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_dep = beta + FPA_dep
Dv_dep_vec.vnb = Dv_dep * [cosd(abs(alpha_dep)), sind(abs(alpha_dep))]

% Arrival
Dv_arr = sqrt(v_arr^2 + v2^2 - 2 * v_arr * v2 * cosd(FPA_arr))
beta1 = asin_dbval(v_arr / Dv_arr * sind(FPA_arr), "deg")
beta2 = acos_dbval((v2^2 + Dv_arr^2 - v_arr^2)/2/v2/Dv_arr, "deg")
beta = intersect(round(beta1,4), round(beta2,4))
alpha_arr =  beta + FPA_arr
Dv_arr_vec.vnb = Dv_arr * [cosd(alpha_arr), -sind(alpha_arr)]

% Total
Dv_tot = Dv_dep + Dv_arr

% (e)
v_inf_earth = Dv_dep;
h = 1000;
En_fb = v_inf_earth^2 / 2
a_fb = -mu_earth / 2 / En_fb
e_fb = 1 - (earth.mer + h) / a_fb
delta = 2 * asind(1 / e_fb)
v_fb1 = vis_viva(earth.mer + h, a_fb, mu_earth)

% (f)
v_inf_mars = Dv_arr;
h = 500;
En_fb = v_inf_mars^2 / 2
a_fb = -mu_mars / 2 / En_fb
e_fb = 1 - (mars.mer + h) / a_fb
delta = 2 * asind(1 / e_fb)
v_fb2 = vis_viva(mars.mer + h, a_fb, mu_mars)

% Plotting

% Earth Orbit
angles = 0:360;
X_earth = earth.smao * cosd(angles); Y_earth = earth.smao * sind(angles);

% Transfer Orbit
angles = TA_dep:0.0001:TA_arr;
R_x = p ./ (1 + e * cosd(angles));
X_x = R_x .* cosd(angles); Y_x = R_x .* sind(angles);

% Mars Orbit
angles = 0:360;
X_mars = mars.smao * cosd(angles); Y_mars = mars.smao * sind(angles);

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

```
hold on; grid on; grid minor; box on; axis equal;
plot(X_x, Y_x)
plot(X_mars, Y_mars)
plot([0, X_x(1)], [0, Y_x(1)], '--k')
plot([0, X_x(end)], [0, Y_x(end)], '--k')
hold off
title('Problem 4: Earth to Mars Elliptic Transfer with TOF of 173 Days - T. Koike')
xlabel('x-direction')
ylabel('y-direction')
saveas(fig, fullfile(fdir, 'p4_transfer.png'));
```

aae532_hw10_p4_extra.mlx

```
% AAE 532 HW 10 Problem 4 Extra
% Tomoki Koike
close all; clear all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\matlab\outputs\ps10';
set(groot, 'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% Set constants
planet_consts = setup_planetary_constants();  % Function that sets up all the constants
in the table
mars = planet_consts.mars;  % structure of mars
earth = planet_consts.earth;  % structure of earth
sun = planet_consts.sun;  % structure of sun
G = 6.6743015e-20;  % Gravitational constant [km^3/kg/s^2]

mu_sun = sun.gp;  % gravitational parameter of sun
mu_earth = earth.gp;  % gravitational parameter of earth
mu_mars = mars.gp;  % gravitational parameter of mars
R_mars = mars.mer;  % radius of Mars
R_earth = earth.mer;  % radius of earth

r1 = earth.smao  % the first leg of the space triangle equal to the semi-major axis of
earth
r2 = mars.smao  % the second leg of the space triangle equal to the semi-major axis of
mars
v1 = sqrt(mu_sun / r1);  % initial velocity in the orbit of earth
v2 = sqrt(mu_sun / r2);  % final velocity in the orbit of mars

Dv_min_all = [];
TOF_min_all = [];
TrA_array = 90:270;
% Plot
fig1 = figure("Renderer","painters","Position",[10 10 900 1200]);
for TrA = TrA_array

    if TrA < 180
        c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(TrA));  % chord length of the space triangle
        typehold = "1";
    else
        c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(360 - TrA));
        typehold = "2";
    end
```

```
    s = (r1 + r2 + c) / 2;  % semi-perimeter
    % TOF for parabolic orbit
    TOF_para = lambertTOF_para(mu_sun, s, c, typehold);
    TOF_para_days = TOF_para / 60 / 60 / 24;

    % Check which type of elliptical transfer
    a_min = s/2;  % the minimum possible semi-major axis of the transfer orbit
    alpha_o = 2*asin(sqrt(s / 2 / a_min));  % base alpha parameter
    beta_o = 2*asin(sqrt((s - c) / 2 / a_min));  % base beta parameter
    % Calculate the minimum TOF using lambert equation
    temp = typehold + "A";
    TOF_min = lambert_ellip(alpha_o, beta_o, mu_sun, "TOF", temp, a_min);
    TOF_min_days = TOF_min / 60 / 60 / 24;

    % Preallocate total DV vector
    Dv_tot_array = [];

    % For loop to find the optimal TOF for given space triangle
    TOF_days_array = round(TOF_para_days)+1:TOF_min_days*2.5;
    for TOF_days = TOF_days_array
        revert = 0;
        TOF = TOF_days * 60 * 60 * 24;  % TOF [s]
        if TOF > TOF_min
            type = typehold + "B";
        else
            type = typehold + "A";
        end
        % Find the SMA corresponding to the TOF
        [a, alpha, beta, fval] = find_lambertEllip_SMA(s, c, TOF, mu_sun, a_min, type);

        [p, e, TA_dep, TA_arr, p_db, TA_dep_db, TA_arr_db] = find_lambertX_ellip(a,
alpha, ...
            beta, TrA, s, c, r1, r2, mu_sun, TOF_days, "day");

        % Calculate energy
        En = -mu_sun / 2 / a;
        % Velocity and FPA
        v_dep = vis_viva(r1, a, mu_sun);
        v_arr = vis_viva(r2, a, mu_sun);
        FPA_dep = acos_dbval(sqrt(mu_sun * p)/r1 / v_dep, "deg");
        FPA_arr = acos_dbval(sqrt(mu_sun * p)/r2 / v_arr, "deg");
        if TA_dep > 0
            FPA_dep = FPA_dep(FPA_dep > 0);
        else
            FPA_dep = FPA_dep(FPA_dep < 0);
        end
        if TA_arr > 0
            FPA_arr = FPA_arr(FPA_arr > 0);
        else
            FPA_arr = FPA_arr(FPA_arr < 0);
        end

        % Departure
        Dv_dep = sqrt(v_dep^2 + v1^2 - 2 * v_dep * v1 * cosd(FPA_dep));
        % Arrival
        Dv_arr = sqrt(v_arr^2 + v2^2 - 2 * v_arr * v2 * cosd(FPA_arr));
        % Total
```

```
        Dv_tot = Dv_dep + Dv_arr;
        if isempty(Dv_tot)
            Dv_tot_array = [Dv_tot_array, NaN];
        else
            Dv_tot_array = [Dv_tot_array, Dv_tot];
        end
    end
    [TF, P] = islocalmin(Dv_tot_array);
    Dv_mins = Dv_tot_array(TF);
    TOF_mins = TOF_days_array(TF);
    dvmin = Dv_mins(1);
    tofmin = TOF_mins(1);
    TOF_min_all = [TOF_min_all, tofmin];
    Dv_min_all = [Dv_min_all, dvmin];


    hold on; grid on; grid minor; box on;
    plot(TOF_days_array, Dv_tot_array, '-')
end
hold off;
title('Total DV over TOF $90\leq TA\leq 270$ - T. Koike')
xlabel('TOF [days]')
ylabel('total DV [km/s]')
saveas(fig1, fullfile(fdir, "p4_DV_TOF_TArange.png"));

% Another plot
fig2 = figure("Renderer","painters","Position",[10 10 900 700]);
yyaxis left
plot(TrA_array, Dv_min_all)
ylabel('total delta-V [km/s]')
yyaxis right
plot(TrA_array, TOF_min_all)
ylabel('TOF [days]')
title('All Minimum DVs and Corresponding TOFs for Each TA - T. Koike ')
xlabel('TA [deg]')
grid on; grid minor; box on;
saveas(fig2, fullfile(fdir, 'p4_minDV_TOF.png'));

Dv_min_all = [];
TOF_min_all = [];
TrA_array = 90:30:270;
% Plot
fig3 = figure("Renderer","painters","Position",[10 10 900 1200]);
for TrA = TrA_array

    if TrA < 180
        c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(TrA));  % chord length of the space triangle
        typehold = "1";
    else
        c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(360 - TrA));
        typehold = "2";
    end

    s = (r1 + r2 + c) / 2;  % semi-perimeter
    % TOF for parabolic orbit
    TOF_para = lambertTOF_para(mu_sun, s, c, typehold);
    TOF_para_days = TOF_para / 60 / 60 / 24;
```

```
    % Check which type of elliptical transfer
    a_min = s/2;  % the minimum possible semi-major axis of the transfer orbit
    alpha_o = 2*asin(sqrt(s / 2 / a_min));  % base alpha parameter
    beta_o = 2*asin(sqrt((s - c) / 2 / a_min));  % base beta parameter
    % Calculate the minimum TOF using lambert equation
    temp = typehold + "A";
    TOF_min = lambert_ellip(alpha_o, beta_o, mu_sun, "TOF", temp, a_min);
    TOF_min_days = TOF_min / 60 / 60 / 24;

    % Preallocate total DV vector
    Dv_tot_array = [];

    % For loop to find the optimal TOF for given space triangle
    TOF_days_array = round(TOF_para_days)+1:TOF_min_days*2.5;
    for TOF_days = TOF_days_array
        revert = 0;
        TOF = TOF_days * 60 * 60 * 24;  % TOF [s]
        if TOF > TOF_min
            type = typehold + "B";
        else
            type = typehold + "A";
        end
        % Find the SMA corresponding to the TOF
        [a, alpha, beta, fval] = find_lambertEllip_SMA(s, c, TOF, mu_sun, a_min, type);

        [p, e, TA_dep, TA_arr, p_db, TA_dep_db, TA_arr_db] = find_lambertX_ellip(a,
alpha, ...
            beta, TrA, s, c, r1, r2, mu_sun, TOF_days, "day");

        % Calculate energy
        En = -mu_sun / 2 / a;
        % Velocity and FPA
        v_dep = vis_viva(r1, a, mu_sun);
        v_arr = vis_viva(r2, a, mu_sun);
        FPA_dep = acos_dbval(sqrt(mu_sun * p)/r1 / v_dep, "deg");
        FPA_arr = acos_dbval(sqrt(mu_sun * p)/r2 / v_arr, "deg");
        if TA_dep > 0
            FPA_dep = FPA_dep(FPA_dep > 0);
        else
            FPA_dep = FPA_dep(FPA_dep < 0);
        end
        if TA_arr > 0
            FPA_arr = FPA_arr(FPA_arr > 0);
        else
            FPA_arr = FPA_arr(FPA_arr < 0);
        end

        % Departure
        Dv_dep = sqrt(v_dep^2 + v1^2 - 2 * v_dep * v1 * cosd(FPA_dep));
        % Arrival
        Dv_arr = sqrt(v_arr^2 + v2^2 - 2 * v_arr * v2 * cosd(FPA_arr));
        % Total
        Dv_tot = Dv_dep + Dv_arr;
        if isempty(Dv_tot)
            Dv_tot_array = [Dv_tot_array, NaN];
        else
            Dv_tot_array = [Dv_tot_array, Dv_tot];
        end
```

```
    end
    [TF, P] = islocalmin(Dv_tot_array);
    Dv_mins = Dv_tot_array(TF);
    TOF_mins = TOF_days_array(TF);
    dvmin = Dv_mins(1);
    tofmin = TOF_mins(1);
    TOF_min_all = [TOF_min_all, tofmin];
    Dv_min_all = [Dv_min_all, dvmin];


    hold on; grid on; grid minor; box on;
    plot(TOF_days_array, Dv_tot_array, '-')
end
hold off;
title('Total DV over TOF $90\leq TA\leq 270$ with 30 degrees Interval - T. Koike')
xlabel('TOF [days]')
ylabel('total DV [km/s]')
legend('90','120','150','180','210','240','270')
saveas(fig3, fullfile(fdir, "p4_DV_TOF_TArange_30.png"));
```

MATLAB FUNCTION FILES

vis_viva.mlx

```
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
```

acos_dbval.mlx

```
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
```

asin_dbval.mlx

```
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
```

lambert_ellip.mlx

```
function res = lambert_ellip(alpha, beta, mu, goal, type, param)
%{
       NAME:    lambert_ellip
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) alpha:  alpha parameter **[rad]
                (2) beta:   beta parameter  **[rad]
                (3) goal:   'a' or 'TOF'
                (4) mu:     gravitational parameter
                (5) type:   '1A', '1B', '2A', or '2B'
                (6) param:  the parameter (a or TOF) depending on your goal
       OUTPUTS: (1) res: a or TOF depending on your goal
       DESCRIPTION: Finds the semi-major axis of TOF using Lambert equation
       for ellipses.
 %}

    % Input validations
    if goal ~= "a" && goal ~= "TOF"
        error("Accepted goals are only 'a' or 'TOF'.");
    elseif type ~= "1A" && type ~= "1B" && type ~= "2A" && type ~= "2B"
        error("Accepted types are only '1A', '1B', '2A', or '2B'.");
    end

    % The value that changes depending on the type
    if type == "1A"
        X = ((alpha - sin(alpha)) - (beta - sin(beta)));
    elseif type == "2B"
        X = (2*pi - (alpha - sin(alpha)) + (beta - sin(beta)));
    elseif type == "1B"
        X = (2*pi - (alpha - sin(alpha)) - (beta - sin(beta)));
    else
        X = ((alpha - sin(alpha) + (beta - sin(beta))));
    end

    if goal == "TOF"
        a = param;
        res = sqrt(a^3 / mu) * X;
    else
        TOF = param;
        res = (sqrt(mu) * TOF / X)^(2/3);
    end
end
```

lambertTOF_para.mlx

```
function TOF = lambertTOF_para(mu, s, c, type)
    % Function that calculates the TOF for parabolic transfer from
    % Lambert's equation
    if type == "1"
        TOF = 1/3 * sqrt(2/mu) * (s^(3/2) - (s - c)^(3/2));
    else
        TOF = 1/3 * sqrt(2/mu) * (s^(3/2) + (s - c)^(3/2));
    end
end
```

find_lambertEllip_SMA.mlx

```
function [a_opt, alpha_opt, beta_opt, fval] = find_lambertEllip_SMA(s, c, TOF, mu, a_min,
type)
    %{
        Description: This function interatively finds the optimal semi major
        axis value from the given time of flight value using Lambert's TOF
        equation for elliptical transfer orbits of type 1A, 2A, 1B, 2B.

        Author: Tomoki Koike
    %}

    lb = a_min; ub = inf;
    J = []; b = [];
    Aeq = []; beq = [];
    options=optimset('Display','off');
    objfunc = @(a) lambertEllip_opt_SMA(a, s, c, TOF, mu, type);
    [a_opt, fval] = patternsearch(objfunc,a_min,J,b,Aeq,beq,lb,ub,options);
    % alpha and beta
    alpha_opt = 2 * asin(sqrt(s / 2 / a_opt));
    beta_opt = 2 * asin(sqrt((s - c) / 2 / a_opt));

    % Nested optimization objective function
    function obj = lambertEllip_opt_SMA(a, s, c, TOF, mu, type)
        if type ~= "1A" && type ~= "1B" && type ~= "2A" && type ~= "2B"
            error("Accepted types are only '1A', '1B', '2A', or '2B'.");
        end
        % alpha and beta
        alpha = 2 * asin(sqrt(s / 2 / a));
        beta = 2 * asin(sqrt((s - c) / 2 / a));
        % The value that changes depending on the type
        if type == "1A"
            X = ((alpha - sin(alpha)) - (beta - sin(beta)));
        elseif type == "2B"
            X = (2*pi - (alpha - sin(alpha)) + (beta - sin(beta)));
        elseif type == "1B"
            X = (2*pi - (alpha - sin(alpha)) - (beta - sin(beta)));
        else
            X = ((alpha - sin(alpha) + (beta - sin(beta))));
        end

        obj = abs(TOF - (sqrt(a^3 / mu) * X));
    end
end
```

find_lambertHyp_SMA.mlx

```
function [a_opt, alpha_opt, beta_opt, fval] = find_lambertHyp_SMA(s, c, TOF, mu, a_min,
type)
    %{
        Description: This function interatively finds the optimal semi major
        axis value from the given time of flight value using Lambert's TOF
        equation for hyperbolic transfer orbits of type 1H, 2H.

        Author: Tomoki Koike
    %}

    lb = -inf; ub = 0;
    J = []; b = [];
    Aeq = []; beq = [];
    objfunc = @(a) lambertHyp_opt_SMA(a, s, c, TOF, mu, type);
    [a_opt, fval] = patternsearch(objfunc,-a_min,J,b,Aeq,beq,lb,ub);

    % alpha and beta
    alpha_opt = 2 * asinh(sqrt(s / 2 / abs(a_opt)));
    beta_opt = 2 * asinh(sqrt((s - c) / 2 / abs(a_opt)));

    % Nested optimization objective function
    function obj = lambertHyp_opt_SMA(a, s, c, TOF, mu, type)
        if type ~= "1H" && type ~= "2H"
            error("Accepted types are only '1H' or '2H'.");
        end
        % alpha and beta
        alpha = 2 * asinh(sqrt(s / 2 / abs(a)));
        beta = 2 * asinh(sqrt((s - c) / 2 / abs(a)));

        % The value that changes depending on the type
        if type == "1H"
            X = ((sinh(alpha) - alpha) - (sinh(beta) - beta));
        else
            X = ((sinh(alpha) - alpha) + (sinh(beta) - beta));
        end

        obj = abs(TOF - (sqrt(abs(a)^3 / mu) * X));
    end
end
```

find_lambertX_ellip.mlx

```
function [p, e, TA_dep, TA_arr, p_db, TA_dep_db, TA_arr_db] = find_lambertX_ellip(a,
alpha, ...
    beta, TrA, s, c, r1, r2, mu, TOF, timeunit)
    p = (4*a*(s - r1)*(s - r2) / c^2 * ...
            [sin((alpha + beta) / 2)^2, sin((alpha - beta) / 2)^2]);  % semi-latus rectum
    p_db = p;
    e = sqrt(1 - p./a);
    e_db = e;
    % Transfer orbit characteristics
    TA_dep_db1 = acos_dbval(1/e(1) * (p(1)/r1 - 1), "deg");
    TA_arr_db1 = acos_dbval(1/e(1) * (p(1)/r2 - 1), "deg");
    TA_dep_db2 = acos_dbval(1/e(2) * (p(2)/r1 - 1), "deg");
```

```
    TA_arr_db2 = acos_dbval(1/e(2) * (p(2)/r2 - 1), "deg");
    TA_deps = [TA_dep_db1, TA_dep_db2];
    TA_arrs = [TA_arr_db1, TA_arr_db2]';
    TA_deps = repmat(TA_deps, numel(TA_deps), 1);
    TA_arrs = repmat(TA_arrs, 1, numel(TA_arrs));
    D_TA = TA_arrs - TA_deps;
    [I1, J1] = find(round(D_TA) == TrA);
    [I2, J2] = find(round(D_TA) == TrA - 360);

    flag = 0;
    for ix = I1'
        flag = 0;
        for jy = J1'
            if (1 <= ix && ix <= 2) && (1 <= jy && jy <= 2)
                p = p_db(1); e = e_db(1);
            elseif (3 <= ix && ix <= 4) && (3 <= jy && jy <= 4)
                p = p_db(2); e = e_db(2);
            end
            if checkLambert_TOF(a,e,mu,TA_deps(ix,jy),TA_arrs(ix,jy),TOF,timeunit)
                i = ix; j = jy;
                flag = 1;
            end
        end
        if flag == 1
            break
        end
    end
    if flag ~= 1
        for ix = I2'
            flag = 0;
            for jy = J2'
                if (1 <= ix && ix <= 2) && (1 <= jy && jy <= 2)
                    p = p_db(1); e = e_db(1);
                elseif (3 <= ix && ix <= 4) && (3 <= jy && jy <= 4)
                    p = p_db(2); e = e_db(2);
                end
                if checkLambert_TOF(a,e,mu,TA_deps(ix,jy),TA_arrs(ix,jy),TOF,timeunit)
                    i = ix; j = jy;
                    flag = 1;
                end
            end
            if flag == 1
                break
            end
        end
    end
    TA_dep_db = TA_deps(1,1:end);
    TA_arr_db = TA_arrs(1:end,1);
    TA_dep = TA_deps(i, j);
    TA_arr = TA_arrs(i, j);

end
```

find_lambertX_hyp.mlx

```
function [p, p_db, e, TA_dep, TA_arr, TA_dep_db, TA_arr_db] = find_lambertX_hyp(a,
alpha, ...
    beta, TrA, s, c, r1, r2)
    p = (4*abs(a)*(s - r1)*(s - r2) / c^2 * ...
        [sinh((alpha + beta) / 2)^2, sinh((alpha - beta) / 2)^2]);  % semi-latus rectum
    p_db = p;
    % Choose the right p
    flag = 1;
    for pi = p
        % Eccnetricity
        ei = sqrt(1 - pi/a);

        % Transfer orbit characteristics
        TA_dep_db = acos_dbval(1/ei * (pi/r1 - 1), "deg");
        TA_arr_db = acos_dbval(1/ei * (pi/r2 - 1), "deg");

        % Choose the right TAs
        for theta1 = TA_dep_db
            for theta2 = TA_arr_db
                temp = round(theta2 - theta1);
                if temp == TrA
                    TA_dep = theta1;
                    TA_arr = theta2;
                    p = pi;
                    e = ei;
                    flag = 0;
                    break
                end
            end
            if flag == 0
                break
            end
        end
        if flag == 0
            break
        end
    end
    TA_dep_db = []; TA_arr_db = [];
    for pi = p_db
        % Eccnetricity
        ei = sqrt(1 - pi/a);

        % Transfer orbit characteristics
        TA_dep_db = horzcat(TA_dep_db, acos_dbval(1/ei * (pi/r1 - 1), "deg"));
        TA_arr_db = horzcat(TA_arr_db, acos_dbval(1/ei * (pi/r2 - 1), "deg"));
    end
end
```

T2E_anomaly.mlx

```
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
```

T2H_anomaly.mlx

```
function H = T2H_anomaly(e, theta_star, unit)
%{
       NAME:    T2H_anomaly
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) e:          ECCENTRICITY
                (2) theta_star: TRUE ANOMALY
                (3) unit:       DEGREES OR RADIANS
       OUTPUTS: (1) H:          HYPERBOLIC ANOMALY
       DESCRIPTION: CALCULATES THE HYPERBOLIC ANOMALY FROM THE TRUE ANOMALY.
 %}

    ee = sqrt((e - 1) / (e + 1));
    if unit == "deg"
        H = rad2deg(2*atanh(ee * tand(theta_star / 2)));
    else
        H = 2*atanh(ee * tan(theta_star / 2));
    end
end
```

checkLambert_TOF.mlx

```
function flag = checkLambert_TOF(a, e, mu, TA_dep, TA_arr, TOF, timeunit)
%{
    Function that checks whether or not the lambert arc satisfies the
    mission TOF. Angles are in degrees and TOF is in
%}

% Eccentric Anomalies
E1 = T2E_anomaly(e, TA_dep, "deg");
E2 = T2E_anomaly(e, TA_arr, "deg");

if E2 < 0
    E2 = E2 + 360;
end

% Time elapses
t1 = sqrt(a^3 / mu) * (deg2rad(E1) - e * sind(E1));
t2 = sqrt(a^3 / mu) * (deg2rad(E2) - e * sind(E2));
```

```
T = (t2 - t1);

if timeunit == "year"
    T = T / 60 / 60 / 24 / 365;
elseif timeunit == "day"
    T = T / 60 / 60 / 24;
elseif timeunit == "hour"
    T = T / 60 / 60;
else
    T = T / 60;
end

flag = round(T) == TOF;

end
```

GMAT Scripts

ps10_p1.script

```
%General Mission Analysis Tool(GMAT) Script
%Created: 2020-11-30 19:49:49


%----------------------------------------
%---------- Spacecraft
%----------------------------------------

Create Spacecraft sat1;
GMAT sat1.DateFormat = UTCGregorian;
GMAT sat1.Epoch = '01 Jan 2000 11:59:28.000';
GMAT sat1.CoordinateSystem = Mars2000Eq;
GMAT sat1.DisplayStateType = Keplerian;
GMAT sat1.SMA = 9376.000000043741;
GMAT sat1.ECC = 1.486692424452262e-12;
GMAT sat1.INC = 0;
GMAT sat1.RAAN = 0;
GMAT sat1.AOP = 0;
GMAT sat1.TA = 0;
GMAT sat1.DryMass = 850;
GMAT sat1.Cd = 2.2;
GMAT sat1.Cr = 1.8;
GMAT sat1.DragArea = 15;
GMAT sat1.SRPArea = 1;
GMAT sat1.SPADDragScaleFactor = 1;
GMAT sat1.SPADSRPScaleFactor = 1;
GMAT sat1.NAIFId = -10000001;
GMAT sat1.NAIFIdReferenceFrame = -9000001;
GMAT sat1.OrbitColor = Red;
GMAT sat1.TargetColor = Teal;
GMAT sat1.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0 0 0 0 ; 0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0
1e+70 0 ; 0 0 0 0 0 1e+70 ];
GMAT sat1.CdSigma = 1e+70;
GMAT sat1.CrSigma = 1e+70;
GMAT sat1.Id = 'SatId';
GMAT sat1.Attitude = CoordinateSystemFixed;
GMAT sat1.SPADSRPInterpolationMethod = Bilinear;
GMAT sat1.SPADSRPScaleFactorSigma = 1e+70;
GMAT sat1.SPADDragInterpolationMethod = Bilinear;
GMAT sat1.SPADDragScaleFactorSigma = 1e+70;
GMAT sat1.ModelFile = 'aura.3ds';
GMAT sat1.ModelOffsetX = 0;
GMAT sat1.ModelOffsetY = 0;
GMAT sat1.ModelOffsetZ = 0;
GMAT sat1.ModelRotationX = 0;
GMAT sat1.ModelRotationY = 0;
GMAT sat1.ModelRotationZ = 0;
GMAT sat1.ModelScale = 1;
GMAT sat1.AttitudeDisplayStateType = 'Quaternion';
GMAT sat1.AttitudeRateDisplayStateType = 'AngularVelocity';
GMAT sat1.AttitudeCoordinateSystem = EarthMJ2000Eq;
GMAT sat1.EulerAngleSequence = '321';
```

```
%----------------------------------------
%---------- ForceModels
%----------------------------------------

Create ForceModel MarsTwoBody_ForceModel;
GMAT MarsTwoBody_ForceModel.CentralBody = Mars;
GMAT MarsTwoBody_ForceModel.PrimaryBodies = {Mars};
GMAT MarsTwoBody_ForceModel.Drag = None;
GMAT MarsTwoBody_ForceModel.SRP = Off;
GMAT MarsTwoBody_ForceModel.RelativisticCorrection = Off;
GMAT MarsTwoBody_ForceModel.ErrorControl = RSSStep;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.Degree = 4;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.Order = 4;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.StmLimit = 100;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.PotentialFile = 'Mars50c.cof';
GMAT MarsTwoBody_ForceModel.GravityField.Mars.TideModel = 'None';

%----------------------------------------
%---------- Propagators
%----------------------------------------

Create Propagator MarsTwoBody;
GMAT MarsTwoBody.FM = MarsTwoBody_ForceModel;
GMAT MarsTwoBody.Type = RungeKutta89;
GMAT MarsTwoBody.InitialStepSize = 60;
GMAT MarsTwoBody.Accuracy = 9.999999999999999e-12;
GMAT MarsTwoBody.MinStep = 0.001;
GMAT MarsTwoBody.MaxStep = 2700;
GMAT MarsTwoBody.MaxStepAttempts = 50;
GMAT MarsTwoBody.StopIfAccuracyIsViolated = true;

%----------------------------------------
%---------- Burns
%----------------------------------------

Create ImpulsiveBurn ImpulsiveBurn1;
GMAT ImpulsiveBurn1.CoordinateSystem = Local;
GMAT ImpulsiveBurn1.Origin = Mars;
GMAT ImpulsiveBurn1.Axes = VNB;
GMAT ImpulsiveBurn1.Element1 = 0.4296;
GMAT ImpulsiveBurn1.Element2 = 0;
GMAT ImpulsiveBurn1.Element3 = -0.4156;
GMAT ImpulsiveBurn1.DecrementMass = false;
GMAT ImpulsiveBurn1.Isp = 300;
GMAT ImpulsiveBurn1.GravitationalAccel = 9.81;

Create ImpulsiveBurn ImpulsiveBurn2;
GMAT ImpulsiveBurn2.CoordinateSystem = Local;
GMAT ImpulsiveBurn2.Origin = Mars;
GMAT ImpulsiveBurn2.Axes = VNB;
GMAT ImpulsiveBurn2.Element1 = 0.0964;
GMAT ImpulsiveBurn2.Element2 = 0;
GMAT ImpulsiveBurn2.Element3 = 0.5568;
GMAT ImpulsiveBurn2.DecrementMass = false;
GMAT ImpulsiveBurn2.Isp = 300;
```

```
GMAT ImpulsiveBurn2.GravitationalAccel = 9.81;

%----------------------------------------
%---------- Coordinate Systems
%----------------------------------------

Create CoordinateSystem Mars2000Eq;
GMAT Mars2000Eq.Origin = Mars;
GMAT Mars2000Eq.Axes = MJ2000Eq;

%----------------------------------------
%---------- Subscribers
%----------------------------------------

Create OrbitView MarsOrbit;
GMAT MarsOrbit.SolverIterations = Current;
GMAT MarsOrbit.UpperLeft = [ 0 0 ];
GMAT MarsOrbit.Size = [ 0.8 0.85 ];
GMAT MarsOrbit.RelativeZOrder = 15;
GMAT MarsOrbit.Maximized = false;
GMAT MarsOrbit.Add = {sat1, Mars};
GMAT MarsOrbit.CoordinateSystem = Mars2000Eq;
GMAT MarsOrbit.DrawObject = [ true true ];
GMAT MarsOrbit.DataCollectFrequency = 1;
GMAT MarsOrbit.UpdatePlotFrequency = 50;
GMAT MarsOrbit.NumPointsToRedraw = 0;
GMAT MarsOrbit.ShowPlot = true;
GMAT MarsOrbit.MaxPlotPoints = 20000;
GMAT MarsOrbit.ShowLabels = true;
GMAT MarsOrbit.ViewPointReference = Mars;
GMAT MarsOrbit.ViewPointVector = [ 30000 0 0 ];
GMAT MarsOrbit.ViewDirection = Mars;
GMAT MarsOrbit.ViewScaleFactor = 1;
GMAT MarsOrbit.ViewUpCoordinateSystem = Mars2000Eq;
GMAT MarsOrbit.ViewUpAxis = Z;
GMAT MarsOrbit.EclipticPlane = Off;
GMAT MarsOrbit.XYPlane = On;
GMAT MarsOrbit.WireFrame = Off;
GMAT MarsOrbit.Axes = On;
GMAT MarsOrbit.Grid = Off;
GMAT MarsOrbit.SunLine = Off;
GMAT MarsOrbit.UseInitialView = On;
GMAT MarsOrbit.StarCount = 7000;
GMAT MarsOrbit.EnableStars = Off;
GMAT MarsOrbit.EnableConstellations = On;


%----------------------------------------
%---------- Mission Sequence
%----------------------------------------

BeginMissionSequence;
Propagate MarsTwoBody(sat1) {sat1.ElapsedDays = 0.56, OrbitColor = [255 0 0]};
Maneuver ImpulsiveBurn1(sat1);
Propagate MarsTwoBody(sat1) {sat1.Mars.TA = -150.0214, OrbitColor = [0 0 255]};
Maneuver ImpulsiveBurn2(sat1);
Propagate MarsTwoBody(sat1) {sat1.ElapsedDays = 1.27, OrbitColor = [255 128 0]};
```

ps10_p2.script

```
%General Mission Analysis Tool(GMAT) Script
%Created: 2020-11-30 19:49:49


%----------------------------------------
%---------- Spacecraft
%----------------------------------------

Create Spacecraft sat1;
GMAT sat1.DateFormat = UTCGregorian;
GMAT sat1.Epoch = '01 Jan 2000 11:59:28.000';
GMAT sat1.CoordinateSystem = Sun2000Eq;
GMAT sat1.DisplayStateType = Keplerian;
GMAT sat1.SMA = 149597898;
GMAT sat1.ECC = 0;
GMAT sat1.INC = 0;
GMAT sat1.RAAN = 0;
GMAT sat1.AOP = 0;
GMAT sat1.TA = 0;
GMAT sat1.DryMass = 850;
GMAT sat1.Cd = 2.2;
GMAT sat1.Cr = 1.8;
GMAT sat1.DragArea = 15;
GMAT sat1.SRPArea = 1;
GMAT sat1.SPADDragScaleFactor = 1;
GMAT sat1.SPADSRPScaleFactor = 1;
GMAT sat1.NAIFId = -10000001;
GMAT sat1.NAIFIdReferenceFrame = -9000001;
GMAT sat1.OrbitColor = Red;
GMAT sat1.TargetColor = Teal;
GMAT sat1.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0 0 0 0 ; 0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0
1e+70 0 ; 0 0 0 0 0 1e+70 ];
GMAT sat1.CdSigma = 1e+70;
GMAT sat1.CrSigma = 1e+70;
GMAT sat1.Id = 'SatId';
GMAT sat1.Attitude = CoordinateSystemFixed;
GMAT sat1.SPADSRPInterpolationMethod = Bilinear;
GMAT sat1.SPADSRPScaleFactorSigma = 1e+70;
GMAT sat1.SPADDragInterpolationMethod = Bilinear;
GMAT sat1.SPADDragScaleFactorSigma = 1e+70;
GMAT sat1.ModelFile = 'aura.3ds';
GMAT sat1.ModelOffsetX = 0;
GMAT sat1.ModelOffsetY = 0;
GMAT sat1.ModelOffsetZ = 0;
GMAT sat1.ModelRotationX = 0;
GMAT sat1.ModelRotationY = 0;
GMAT sat1.ModelRotationZ = 0;
GMAT sat1.ModelScale = 1;
GMAT sat1.AttitudeDisplayStateType = 'Quaternion';
GMAT sat1.AttitudeRateDisplayStateType = 'AngularVelocity';
GMAT sat1.AttitudeCoordinateSystem = EarthMJ2000Eq;
GMAT sat1.EulerAngleSequence = '321';
```

```
%----------------------------------------
%---------- ForceModels
%----------------------------------------

Create ForceModel MarsTwoBody_ForceModel;
GMAT MarsTwoBody_ForceModel.CentralBody = Mars;
GMAT MarsTwoBody_ForceModel.PrimaryBodies = {Mars};
GMAT MarsTwoBody_ForceModel.Drag = None;
GMAT MarsTwoBody_ForceModel.SRP = Off;
GMAT MarsTwoBody_ForceModel.RelativisticCorrection = Off;
GMAT MarsTwoBody_ForceModel.ErrorControl = RSSStep;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.Degree = 4;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.Order = 4;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.StmLimit = 100;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.PotentialFile = 'Mars50c.cof';
GMAT MarsTwoBody_ForceModel.GravityField.Mars.TideModel = 'None';

Create ForceModel Helio_ForceModel;
GMAT Helio_ForceModel.CentralBody = Sun;
GMAT Helio_ForceModel.PointMasses = {Sun};
GMAT Helio_ForceModel.Drag = None;
GMAT Helio_ForceModel.SRP = Off;
GMAT Helio_ForceModel.RelativisticCorrection = Off;
GMAT Helio_ForceModel.ErrorControl = RSSStep;

%----------------------------------------
%---------- Propagators
%----------------------------------------

Create Propagator MarsTwoBody;
GMAT MarsTwoBody.FM = MarsTwoBody_ForceModel;
GMAT MarsTwoBody.Type = RungeKutta89;
GMAT MarsTwoBody.InitialStepSize = 60;
GMAT MarsTwoBody.Accuracy = 9.999999999999999e-12;
GMAT MarsTwoBody.MinStep = 0.001;
GMAT MarsTwoBody.MaxStep = 2700;
GMAT MarsTwoBody.MaxStepAttempts = 50;
GMAT MarsTwoBody.StopIfAccuracyIsViolated = true;

Create Propagator Helio;
GMAT Helio.FM = Helio_ForceModel;
GMAT Helio.Type = RungeKutta89;
GMAT Helio.InitialStepSize = 60;
GMAT Helio.Accuracy = 9.999999999999999e-12;
GMAT Helio.MinStep = 0.001;
GMAT Helio.MaxStep = 2700;
GMAT Helio.MaxStepAttempts = 50;
GMAT Helio.StopIfAccuracyIsViolated = true;

%----------------------------------------
%---------- Burns
%----------------------------------------

Create ImpulsiveBurn ImpulsiveBurn1;
GMAT ImpulsiveBurn1.CoordinateSystem = Local;
GMAT ImpulsiveBurn1.Origin = Sun;
GMAT ImpulsiveBurn1.Axes = VNB;
GMAT ImpulsiveBurn1.Element1 = 3.5134;
```

```
GMAT ImpulsiveBurn1.Element2 = 0;
GMAT ImpulsiveBurn1.Element3 = 1.6867;
GMAT ImpulsiveBurn1.DecrementMass = false;
GMAT ImpulsiveBurn1.Isp = 300;
GMAT ImpulsiveBurn1.GravitationalAccel = 9.81;

Create ImpulsiveBurn ImpulsiveBurn2;
GMAT ImpulsiveBurn2.CoordinateSystem = Local;
GMAT ImpulsiveBurn2.Origin = Sun;
GMAT ImpulsiveBurn2.Axes = VNB;
GMAT ImpulsiveBurn2.Element1 = 1.1392;
GMAT ImpulsiveBurn2.Element2 = 0;
GMAT ImpulsiveBurn2.Element3 = -5.3007;
GMAT ImpulsiveBurn2.DecrementMass = false;
GMAT ImpulsiveBurn2.Isp = 300;
GMAT ImpulsiveBurn2.GravitationalAccel = 9.81;

%----------------------------------------
%---------- Coordinate Systems
%----------------------------------------

Create CoordinateSystem Mars2000Eq;
GMAT Mars2000Eq.Origin = Mars;
GMAT Mars2000Eq.Axes = MJ2000Eq;

Create CoordinateSystem Sun2000Eq;
GMAT Sun2000Eq.Origin = Sun;
GMAT Sun2000Eq.Axes = MJ2000Eq;

%----------------------------------------
%---------- Subscribers
%----------------------------------------

Create OrbitView OrbitView1;
GMAT OrbitView1.SolverIterations = Current;
GMAT OrbitView1.UpperLeft = [ 0.002352941176470588 0 ];
GMAT OrbitView1.Size = [ 0.8 0.85 ];
GMAT OrbitView1.RelativeZOrder = 76;
GMAT OrbitView1.Maximized = true;
GMAT OrbitView1.Add = {sat1, Sun};
GMAT OrbitView1.CoordinateSystem = Sun2000Eq;
GMAT OrbitView1.DrawObject = [ true true ];
GMAT OrbitView1.DataCollectFrequency = 1;
GMAT OrbitView1.UpdatePlotFrequency = 50;
GMAT OrbitView1.NumPointsToRedraw = 0;
GMAT OrbitView1.ShowPlot = true;
GMAT OrbitView1.MaxPlotPoints = 4000000;
GMAT OrbitView1.ShowLabels = true;
GMAT OrbitView1.ViewPointReference = Sun;
GMAT OrbitView1.ViewPointVector = [ 0 0 30000 ];
GMAT OrbitView1.ViewDirection = Sun;
GMAT OrbitView1.ViewScaleFactor = 1;
GMAT OrbitView1.ViewUpCoordinateSystem = Sun2000Eq;
GMAT OrbitView1.ViewUpAxis = Z;
GMAT OrbitView1.EclipticPlane = Off;
GMAT OrbitView1.XYPlane = On;
GMAT OrbitView1.WireFrame = Off;
GMAT OrbitView1.Axes = On;
```

```
GMAT OrbitView1.Grid = Off;
GMAT OrbitView1.SunLine = Off;
GMAT OrbitView1.UseInitialView = On;
GMAT OrbitView1.StarCount = 7000;
GMAT OrbitView1.EnableStars = Off;
GMAT OrbitView1.EnableConstellations = On;


%----------------------------------------
%---------- Mission Sequence
%----------------------------------------

BeginMissionSequence;
Propagate Helio(sat1) {sat1.ElapsedDays = 365, OrbitColor = [255 0 0]};
Maneuver ImpulsiveBurn1(sat1);
Propagate Helio(sat1) {sat1.Sun.TA = 134.2200, OrbitColor = [0 0 255]};
Maneuver ImpulsiveBurn2(sat1);
Propagate Helio(sat1) {sat1.ElapsedDays = 1000, OrbitColor = [255 128 0]};
```

ps10_p3.script

```
%General Mission Analysis Tool(GMAT) Script
%Created: 2020-11-30 19:49:49


%----------------------------------------
%---------- Spacecraft
%----------------------------------------

Create Spacecraft sat1;
GMAT sat1.DateFormat = UTCGregorian;
GMAT sat1.Epoch = '01 Jan 2000 11:59:28.000';
GMAT sat1.CoordinateSystem = Sun2000Eq;
GMAT sat1.DisplayStateType = Keplerian;
GMAT sat1.SMA = 149597898;
GMAT sat1.ECC = 0;
GMAT sat1.INC = 0;
GMAT sat1.RAAN = 0;
GMAT sat1.AOP = 0;
GMAT sat1.TA = 0;
GMAT sat1.DryMass = 850;
GMAT sat1.Cd = 2.2;
GMAT sat1.Cr = 1.8;
GMAT sat1.DragArea = 15;
GMAT sat1.SRPArea = 1;
GMAT sat1.SPADDragScaleFactor = 1;
GMAT sat1.SPADSRPScaleFactor = 1;
GMAT sat1.NAIFId = -10000001;
GMAT sat1.NAIFIdReferenceFrame = -9000001;
GMAT sat1.OrbitColor = Red;
GMAT sat1.TargetColor = Teal;
GMAT sat1.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0 0 0 0 ; 0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0
1e+70 0 ; 0 0 0 0 0 1e+70 ];
GMAT sat1.CdSigma = 1e+70;
GMAT sat1.CrSigma = 1e+70;
GMAT sat1.Id = 'SatId';
GMAT sat1.Attitude = CoordinateSystemFixed;
```

```
GMAT sat1.SPADSRPInterpolationMethod = Bilinear;
GMAT sat1.SPADSRPScaleFactorSigma = 1e+70;
GMAT sat1.SPADDragInterpolationMethod = Bilinear;
GMAT sat1.SPADDragScaleFactorSigma = 1e+70;
GMAT sat1.ModelFile = 'aura.3ds';
GMAT sat1.ModelOffsetX = 0;
GMAT sat1.ModelOffsetY = 0;
GMAT sat1.ModelOffsetZ = 0;
GMAT sat1.ModelRotationX = 0;
GMAT sat1.ModelRotationY = 0;
GMAT sat1.ModelRotationZ = 0;
GMAT sat1.ModelScale = 1;
GMAT sat1.AttitudeDisplayStateType = 'Quaternion';
GMAT sat1.AttitudeRateDisplayStateType = 'AngularVelocity';
GMAT sat1.AttitudeCoordinateSystem = EarthMJ2000Eq;
GMAT sat1.EulerAngleSequence = '321';


%----------------------------------------
%---------- ForceModels
%----------------------------------------

Create ForceModel MarsTwoBody_ForceModel;
GMAT MarsTwoBody_ForceModel.CentralBody = Mars;
GMAT MarsTwoBody_ForceModel.PrimaryBodies = {Mars};
GMAT MarsTwoBody_ForceModel.Drag = None;
GMAT MarsTwoBody_ForceModel.SRP = Off;
GMAT MarsTwoBody_ForceModel.RelativisticCorrection = Off;
GMAT MarsTwoBody_ForceModel.ErrorControl = RSSStep;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.Degree = 4;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.Order = 4;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.StmLimit = 100;
GMAT MarsTwoBody_ForceModel.GravityField.Mars.PotentialFile = 'Mars50c.cof';
GMAT MarsTwoBody_ForceModel.GravityField.Mars.TideModel = 'None';

Create ForceModel Helio_ForceModel;
GMAT Helio_ForceModel.CentralBody = Sun;
GMAT Helio_ForceModel.PointMasses = {Sun};
GMAT Helio_ForceModel.Drag = None;
GMAT Helio_ForceModel.SRP = Off;
GMAT Helio_ForceModel.RelativisticCorrection = Off;
GMAT Helio_ForceModel.ErrorControl = RSSStep;

%----------------------------------------
%---------- Propagators
%----------------------------------------

Create Propagator Helio;
GMAT Helio.FM = Helio_ForceModel;
GMAT Helio.Type = RungeKutta89;
GMAT Helio.InitialStepSize = 60;
GMAT Helio.Accuracy = 9.999999999999999e-12;
GMAT Helio.MinStep = 0.001;
GMAT Helio.MaxStep = 2700;
GMAT Helio.MaxStepAttempts = 50;
GMAT Helio.StopIfAccuracyIsViolated = true;

%----------------------------------------
```

```
%---------- Burns
%----------------------------------------

Create ImpulsiveBurn ImpulsiveBurn1;
GMAT ImpulsiveBurn1.CoordinateSystem = Local;
GMAT ImpulsiveBurn1.Origin = Sun;
GMAT ImpulsiveBurn1.Axes = VNB;
GMAT ImpulsiveBurn1.Element1 = 10.9361;
GMAT ImpulsiveBurn1.Element2 = 0;
GMAT ImpulsiveBurn1.Element3 = -16.6353;
GMAT ImpulsiveBurn1.DecrementMass = false;
GMAT ImpulsiveBurn1.Isp = 300;
GMAT ImpulsiveBurn1.GravitationalAccel = 9.81;

Create ImpulsiveBurn ImpulsiveBurn2;
GMAT ImpulsiveBurn2.CoordinateSystem = Local;
GMAT ImpulsiveBurn2.Origin = Sun;
GMAT ImpulsiveBurn2.Axes = VNB;
GMAT ImpulsiveBurn2.Element1 = -18.6871;
GMAT ImpulsiveBurn2.Element2 = 0;
GMAT ImpulsiveBurn2.Element3 = -16.3831;
GMAT ImpulsiveBurn2.DecrementMass = false;
GMAT ImpulsiveBurn2.Isp = 300;
GMAT ImpulsiveBurn2.GravitationalAccel = 9.81;

%----------------------------------------
%---------- Coordinate Systems
%----------------------------------------

Create CoordinateSystem Mars2000Eq;
GMAT Mars2000Eq.Origin = Mars;
GMAT Mars2000Eq.Axes = MJ2000Eq;

Create CoordinateSystem Sun2000Eq;
GMAT Sun2000Eq.Origin = Sun;
GMAT Sun2000Eq.Axes = MJ2000Eq;

%----------------------------------------
%---------- Subscribers
%----------------------------------------

Create OrbitView OrbitView1;
GMAT OrbitView1.SolverIterations = Current;
GMAT OrbitView1.UpperLeft = [ 0 0 ];
GMAT OrbitView1.Size = [ 0.8 0.85 ];
GMAT OrbitView1.RelativeZOrder = 142;
GMAT OrbitView1.Maximized = false;
GMAT OrbitView1.Add = {sat1, Sun};
GMAT OrbitView1.CoordinateSystem = Sun2000Eq;
GMAT OrbitView1.DrawObject = [ true true ];
GMAT OrbitView1.DataCollectFrequency = 1;
GMAT OrbitView1.UpdatePlotFrequency = 50;
GMAT OrbitView1.NumPointsToRedraw = 0;
GMAT OrbitView1.ShowPlot = true;
GMAT OrbitView1.MaxPlotPoints = 4000000;
GMAT OrbitView1.ShowLabels = true;
GMAT OrbitView1.ViewPointReference = Sun;
GMAT OrbitView1.ViewPointVector = [ 0 0 30000 ];
```

```
GMAT OrbitView1.ViewDirection = Sun;
GMAT OrbitView1.ViewScaleFactor = 1;
GMAT OrbitView1.ViewUpCoordinateSystem = Sun2000Eq;
GMAT OrbitView1.ViewUpAxis = Z;
GMAT OrbitView1.EclipticPlane = Off;
GMAT OrbitView1.XYPlane = On;
GMAT OrbitView1.WireFrame = Off;
GMAT OrbitView1.Axes = On;
GMAT OrbitView1.Grid = Off;
GMAT OrbitView1.SunLine = Off;
GMAT OrbitView1.UseInitialView = On;
GMAT OrbitView1.StarCount = 7000;
GMAT OrbitView1.EnableStars = Off;
GMAT OrbitView1.EnableConstellations = On;

Create ReportFile ReportFile1;
GMAT ReportFile1.SolverIterations = Current;
GMAT ReportFile1.UpperLeft = [ 0 0 ];
GMAT ReportFile1.Size = [ 0 0 ];
GMAT ReportFile1.RelativeZOrder = 0;
GMAT ReportFile1.Maximized = false;
GMAT ReportFile1.Filename = 'C:\Users\Tomo\Desktop\studies\2020-Fall\AAE532\gmat\ps10\report_p3.txt';
GMAT ReportFile1.Precision = 4;
GMAT ReportFile1.Add = {sat1.Sun.RMAG, sat1.Sun.ECC};
GMAT ReportFile1.WriteHeaders = true;
GMAT ReportFile1.LeftJustify = On;
GMAT ReportFile1.ZeroFill = Off;
GMAT ReportFile1.FixedWidth = true;
GMAT ReportFile1.Delimiter = ' ';
GMAT ReportFile1.ColumnWidth = 23;
GMAT ReportFile1.WriteReport = true;


%----------------------------------------
%---------- Mission Sequence
%----------------------------------------

BeginMissionSequence;
Propagate Helio(sat1) {sat1.ElapsedDays = 365, OrbitColor = [255 0 0]};
Maneuver ImpulsiveBurn1(sat1);
Propagate Helio(sat1) {sat1.Sun.TA = 78.6993, OrbitColor = [0 0 255]};
Maneuver ImpulsiveBurn2(sat1);
Propagate Helio(sat1) {sat1.ElapsedDays = 700, OrbitColor = [255 128 0]};
```