03/09/2021

Gustavo Grinsteins Planchart
ASEN 5050
HW6

Problem 1:
Assumptions:
- Relative 2 Body Problem Assumptions
  - Mass of satellite is negligible compared to attracting body
  - Coordinate system is inertial
  - Satellite and attracting body are treated as point masses
  - No other forces in the system except for gravitational forces
- Gm_{Sun} = 1.32712428 × 1011$km2/s2$
- 1 AU = 149,597,870.7 km

Design a transfer for a spacecraft to travel from the Earth to Venus.

Given:

Earth Sun-Centered inertial coordinates: (At a Julian date of 2459528.5 TDB)

$$\bar{r}_1 = 1.00078x10^8\hat{X} + 1.09250x10^8\hat{Y} - 5.29404x10^3\hat{Z}\ km$$
$$\bar{v}_1 = -22.46086\hat{X} + 20.00474\hat{Y} - 1.79921x10^{-4}\hat{Z}\ km/s$$

Venus Sun-Centered inertial coordinates: (At a Julian date of 2459640.5 TDB)

$$\bar{r}_2 = -1.05048x10^8\hat{X} - 2.37576x10^7\hat{Y} + 5.73539x10^6\hat{Z}\ km$$
$$\bar{v}_2 = 7.49784\hat{X} - 34.31464\hat{Y} - 0.90369\hat{Z}\ km/s$$

a) Write a script to implement an iterative numerical method solving Lambert's equation to calculate a semi-major axis for a transfer with a specified time of flight. Write this method only for arcs along elliptical orbits. Describe the setup of your numerical method, the stopping condition/s and initial guess used – in your own words. Attach your code. (Hint: fsolve is a useful root-finding function in Matlab)

The numerical procedures follow the following steps:

1. Calculate the transfer angle given the current information

$$\Delta\theta^* = \cos^{-1}\left(\frac{\bar{r}_1\bar{r}_2}{|\bar{r}_1||\bar{r}_2|}\right)\left(\frac{180}{\pi}\right) = \pm 145.12° = 145.12°$$

Keeping in mind that the transfer arc $\Delta\theta^* < 180°$. Therefore, we select the positive value.

2. Calculate geometric quantities

Calculating the space triangle chord length:

$$c = \sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\Delta\theta^*)} = 2.4454x10^8\ km$$

Calculating the space triangle semi-perimeter:

$$s = \left(\frac{1}{2}\right)(r_1 + r_2 + c) = 2.5028x10^8 km$$

3. Checking that the expected TOF occurs within an ellipse trajectory

Calculating TOF$_P$ (the time of flight for a parabolic trajectory within the given geometry):

$$TOF_p = \left(\frac{1}{3}\right)\sqrt{\frac{2}{\mu_{sun}}}\left(s^{\frac{3}{2}} + (s-c)^{\frac{3}{2}}\right) = 5.1413x10^6\ seconds = 59.5057\ Days$$

Calculating the expected Venus to Earth TOF using the given Julian Dates:

$$TOF = Venus\ Julian\ Date - Earth\ Julian\ Date = 112\ Days$$

Since TOF$_p$ < TOF the Earth to Venus transfer arc will be elliptical.

4. Calculate minimum energy transfer arc values to adequately define α & β

$$a_m = \left(\frac{S}{2}\right) = 1.2514x10^8 km$$

$$n_m = \sqrt{\frac{\mu_{sun}}{a_m^3}} = 2.6024x10^{-7}\ rad/sec$$

$$\alpha_m = \pi\ Radians$$

$$\beta_{m,0} = 2\sin^{-1}\left(\sqrt{\frac{s-c}{s}}\right) = \pm\ 0.309\ Radians = -0.309\ Radians$$

Select the negative value of $\beta_{m,0}$ since $\Delta\theta^* < 180°$

Calculating TOF$_{min}$ (The time of flight for a minimum energy elliptical transfer orbit):

$$TOF_{Min} = \left(\frac{1}{n_m}\right)\left((\alpha_m - \beta_m) - (\sin(\alpha_m) - \sin(\beta_m))\right) = 1.2054x10^7 seconds = 139.5 Days$$

The algorithm will use this value to adequality define α for lambert's equation.

5. Use MATLAB's fsolve function to iteratively calculate the semi-major axis for the transfer arc.

Note: Please refer to the attached script "EllOrbitLambertEqSolve.m" for the details of the topics described below. The Main script running this function is labeled "Grinsteins_HW6_Code.m"

The defined function takes in the values previously given/calculated on the steps before (s,c,a_m,TOF,TOFmin,mu_sun) with additional Boolean information regarding $\Delta\theta^*$ to correctly define β.

The initial condition for the semi-major axis is set to the semi-major axis of the minimum energy elliptical transfer arc with an added delta value of 150 km given the distances within the Earth and Venus environment.

$$a_{init} = a_m + 150\ km$$

The fsolve accesses the function "LambertEqt.m" which contains the definition of lambert's equation. Conditional statements in this function define alpha/beta through Boolean parameters that contain information about the transfer angle and TOF_{min} before applying the equation definition.

The fsolve function is wrapped inside a while loop that has three build-in stopping conditions:

- The first one (the while loop Boolean condition), continuously checks the difference value between the specified TOF vs the Iteration's TOF. The while loop stops iterating when this value falls below a defined tolerance (currently set to 5 decimal digit accuracy).
- The second stopping condition checks for divergent behavior while finding the semi-major axis. If the difference between the TOF values of interest starts increasing unexpectedly, the function will break from the while loop and set the semi-major axis to the value before the calculations started diverging
- The third stopping condition sets a limit to the number of iterations that can go through before the code breaks from the while loop (currently set to 300). If this threshold is reached, the code will exit giving the user a relevant warning.

b) Use Lambert's problem to design a transfer with a transfer angle of less than 180 degrees and connects these two position vectors at the specified epochs. Find the values of a, e, and Ɛ along the transfer relative to the Sun. Except for using the script from part a) to solve Lambert's equation, you must show all your working by hand or typed up with mathematical notation and discuss the procedure you use to solve this problem.

Using the script defined above we obtain the following semi-major axis for the elliptical transfer arc:

$$\boldsymbol{a_t = 1.29797x10^8 km} < --$$

Double checking this is an acceptable value by recalculating the TOF:

$$n_t = \sqrt{\frac{\mu_{sun}}{a_t^3}} = 2.4635x10^{-7}\ rad/sec$$

$$\alpha_0 = 2\sin^{-1}\left(\sqrt{\frac{s}{2a_t}}\right) = 2.7604\ Radians$$

Since TOF < TOF$_{min}$ $\alpha_t = \alpha_0$

$$\beta_0 = 2\sin^{-1}\left(\sqrt{\frac{s-c}{2a_t}}\right) = 0.29840\ Radians$$

Since $\Delta\theta^* < 180°$ $\beta_t = \beta_0$

$$TOF_{check} = \left(\frac{1}{n_t}\right)\left((\alpha_t - \beta_t) - (\sin(\alpha_t) - \sin(\beta_t))\right) = 9.67679x10^6 seconds$$

$$TOF - TOF_{check} = 2.04891x10^{-8}\ seconds$$

Accuracy to the 8th decimal digit is a good enough approximation.

Finding eccentricity via the semi-latus rectum:

$$p = \frac{4a(s-r_1)(s-r_2)}{c^2}\sin^2\left(\frac{\alpha_t + \beta_t}{2}\right) = 1.2605x10^8 km$$

$$\boldsymbol{e = \sqrt{1 - \frac{p}{a_t}} = 0.16981} < --$$

Calculating specific mechanical energy:

$$\boldsymbol{\varepsilon = -\frac{\mu_{sun}}{2a_t} = -511.24\frac{km^2}{s^2}} < --$$

c) What is the lower bound on the semi-major axis for an elliptical transfer connecting these two position vectors?

The lowest possible semi-major axis that can be used to solve lambert's problem is the elliptical minimum energy semi-major axis. Solutions are not possible for a semi-major axis that is less than a$_{min}$. Therefore, the lower bound on the semi-major axis is:

$$\boldsymbol{a_m} = \left(\frac{\boldsymbol{s}}{\boldsymbol{2}}\right) = \boldsymbol{1.2514x10^8 km} < --$$

d) At both the beginning and end of this transfer, calculate the true anomaly and velocity vectors in a Sun-centered inertial coordinate system. Use these velocities to calculate the maneuver magnitudes Δv_1 and Δv_2 required for the spacecraft to simply match the velocities of Earth and Venus at the beginning and end of the transfer.

Finding the true anomalies at the beginning and end of the transfer arc:

$$\theta_1^* = \cos^{-1}\left(\frac{1}{e}\left(\frac{p}{r_1} - 1\right)\right)\left(\frac{180}{\pi}\right) = \pm\ 151.46°$$

$$\theta_2^* = \cos^{-1}\left(\frac{1}{e}\left(\frac{p}{r_2} - 1\right)\right)\left(\frac{180}{\pi}\right) = \pm 6.36°$$

Given that:

$$\Delta\theta^* = \theta_2^* - \theta_1^* = 145.12°$$

Therefore, the only combination of θ_2^* & θ_1^* that yields this result:

$$\boldsymbol{\theta_1^* = -151.46°} < --$$

$$\boldsymbol{\theta_2^* = -6.36°} < --$$

Using f and g functions to obtain the velocities at the beginning and end of the transfer orbit:

$$\bar{r}_2 = f\bar{r}_1 + g\bar{v}_{1,f}$$
$$\bar{v}_{2,i} = \dot{f}\bar{r}_1 + \dot{g}\bar{v}_{1,f}$$

$$\bar{v}_{1,f} = velocity\ at\ the\ start\ of\ the\ transfer\ arc\ after\ maneuver$$

$$\bar{v}_{2,i} = velocity\ at\ the\ end\ of\ the\ transfer\ arc\ before\ maneuver$$

Where our f and g functions are defined as:

$$f = 1 - \left(\frac{r_2}{p}\right)(1 - \cos(\Delta\theta^*)) = -0.55748$$
$$g = \frac{r_2 r_1 \sin(\Delta\theta^*)}{\sqrt{\mu_{sun} p}} = 2.2343x10^6\ seconds$$

$$\dot{f} = \sqrt{\frac{\mu_{sun}}{p}} \tan\left(\frac{\Delta\theta^*}{2}\right)\left(\frac{1 - \cos(\Delta\theta^*)}{p} - \frac{1}{r_2} - \frac{1}{r_1}\right) = -1.6324x10^{-7}\ s^{-1}$$
$$\dot{g} = 1 - \left(\frac{r_1}{p}\right)(1 - \cos(\Delta\theta^*)) = -1.1395$$

Using these values, we find the velocity vectors/magnitudes of interest using the following equations:

$$\bar{v}_{1,f} = \frac{\bar{r}_2 - f\bar{r}_1}{g} = -22.0454\hat{X} + 16.627\hat{Y} + 2.5656\hat{Z}\ km/s$$
$$v_{1,f} = \left\|\bar{v}_{1,f}\right\| = 27.731\ km/s$$

$$\bar{v}_{2,i} = \dot{f}\bar{r}_1 + \dot{g}\bar{v}_{1,f} = 8.7845\hat{X} - 36.7806\hat{Y} - 2.9228\hat{Z}\ km/s$$

$$v_{2,i} = \left\|\bar{v}_{2,i}\right\| = 37.928\ km/s$$

Calculating the first change in velocity:

$$\Delta\bar{v}_1 = \bar{v}_{1,f} - \bar{v}_1 = 0.41550\hat{X} - 3.3782\hat{Y} + 2.5658\hat{Z}\ km/s$$

$$\|\Delta\boldsymbol{\bar{v}_1}\| = \mathbf{4.2624}\frac{\boldsymbol{km}}{\boldsymbol{s}} < --$$

Calculating the second change in velocity:

$$\Delta\bar{v}_2 = \bar{v}_2 - \bar{v}_{2,i} = -1.2867\hat{X} + 2.4660\hat{Y} + 2.0191\hat{Z}\ km/s$$

$$\|\Delta\boldsymbol{\bar{v}_2}\| = \mathbf{3.4370}\frac{\boldsymbol{km}}{\boldsymbol{s}} < --$$

Problem 2:

<u>Assumptions:</u>

- Relative 2 Body Problem Assumptions
    - Mass of satellite is negligible compared to attracting body
    - Coordinate system is inertial
    - Satellite and attracting body are treated as point masses
    - No other forces in the system except for gravitational forces
- Gm_{Sun} = 1.32712428 × 1011$km2/s2$
- 1 AU = 149,597,870.7 km

Calculate trajectories that deliver a spacecraft from Earth to Mars using provided ephemerides text files.

a) Convert the procedure that you used to solve Problem 1 into a numerical script that you can run to produce transfers with a transfer angle that is less than 180 degrees given the following inputs: the initial and final state vectors and the time of flight. Attach your code.

Note: Please refer to the attached script "EllOrbitLambertEqSolve2.m" for the details of the topics described below. The Main script running this function is labeled "Grinsteins_HW6_Code.m"

This new procedure is similar in structure as the procedure in problem 1. The difference comes from the parameters being passed in to the function: R1,R2,V1,V2,TOF,mu_sun and less180. Correspondingly, the ephemerides data, the calculated TOF for a particular combination of earth and mars trajectories, the sun's gravitational constant, and a Boolean to perform calculations for a transfer angle of less than 180 degrees. Using this information, we can follow the numerical procedure from problem 1 but this time apply most calculations within the EllOrbitLambertEqSolve2 function.

Since all the transfer design calculations are contained within this function, a nested for loop can be built around it (visible in "Grinsteins_HW6_Code.m") in order to iterate through all possible combinations of the given Earth and Mars state vectors. Note that the given ephemerides data was parsed through and modified in MATLAB to optimize this procedure.

There are some combinations of the given values that result in non-elliptical transfer orbits (that is $TOF<TOF_P$). Therefore, the code was adapted in order to define V-infinity values that fall under this category as NaN (not a number) values in MATLAB in order to not graph these values. Parabolic and Hyperbolic transfer orbits have different numerical procedures to calculate their properties so using elliptical orbits procedures would not be accurate.

b) For every possible combination of the provided initial and final states and epochs for the Earth and Mars (where the final epoch occurs after the initial epoch, of course), use your script to design a transfer from the Earth to Mars with a transfer angle that is less than 180 degrees. For each transfer, calculate the v-infinity magnitude at each of Earth departure and Mars arrival; discuss in your writeup how you calculated this quantity. Then, create two porkchop plots: 1) displaying the v-infinity at Earth departure and 2) displaying the v-infinity at Mars arrival. For each porkchop plot, display the epoch associated with the Earth's state vector on the horizontal axis (i.e., the departure date) and display the epoch associated with Mars' state vector on the vertical axis (i.e., the arrival date).

Another adaptation that differentiates EllOrbitLambertEqSolve.m vs EllOrbitLambertEqSolve2.m is the calculation of v-infinity for the earth departure and mars arrival velocities.

In order to calculate these v-infinity values, the code applies f and g functions (as used in P1 part d) after the iterative solver for the semi-major axis converges to a solution.

$$\bar{v}_{1,f} = \frac{\bar{r}_2 - f\bar{r}_1}{g}$$

$$\bar{v}_{2,i} = \dot{f}\bar{r}_1 + \dot{g}\bar{v}_{1,f}$$

Where in this particular problem:

$$\bar{r}_1 = Earth's\ heliocentric\ position\ vector$$

$$\bar{r}_2 = Mars'\ heliocentric\ position\ vector$$

$$\bar{v}_{1,f} = Velocity\ of\ S/C\ at\ the\ start\ of\ the\ transfer$$

$$\bar{v}_{2,f} = Velocity\ of\ S/C\ at\ the\ end\ of\ the\ transfer$$

After obtaining these values, V-infinity can be calculated using the formula provided in lecture and the velocity vectors for Earth and Mars provided by the ephemerides ($\bar{v}_1\ and\ \bar{v}_2$):

$$v_{\infty,Earth} = \left\|\bar{v}_{\infty,Earth}\right\| = \bar{v}_{1,f} - \bar{v}_1$$

$$v_{\infty,Mars} = \left\|\bar{v}_{\infty,Mars}\right\| = \bar{v}_{2,f} - \bar{v}_2$$

Where:

$$v_{\infty,Earth} = v_\infty\ at\ Earth's\ Departure$$

$$v_{\infty,Mars} = v_\infty\ at\ Mars'\ Arrival$$

Prorkchop Plots:

Both of these arrangements have selected value marked contours on the left and non-value marked contours towards the right. This way a descriptive visual data set is shown without cluttering values on one screen.

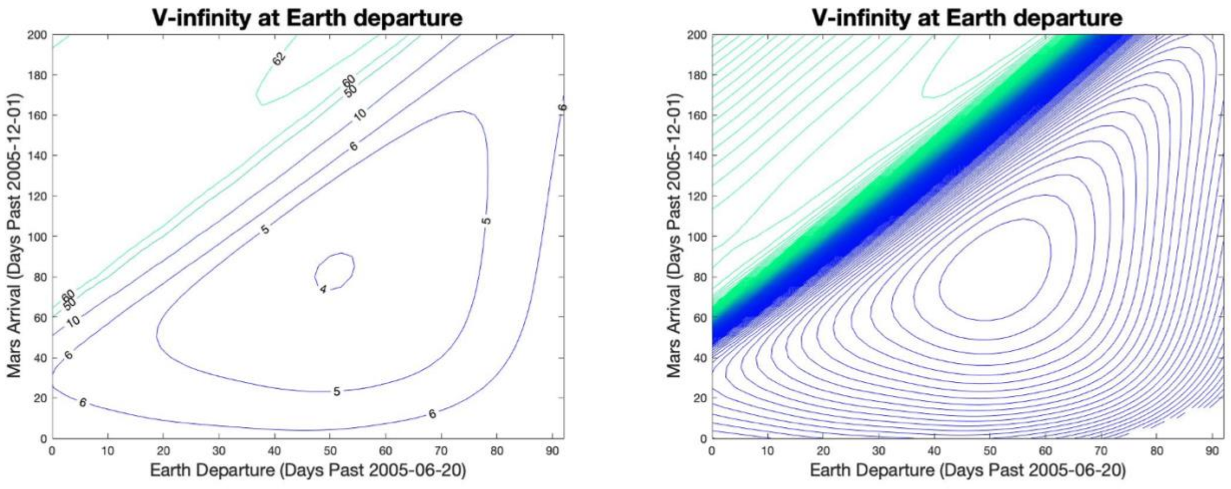


Figure 1 - V-Infinity Porkchop Plot for Earth's Departure

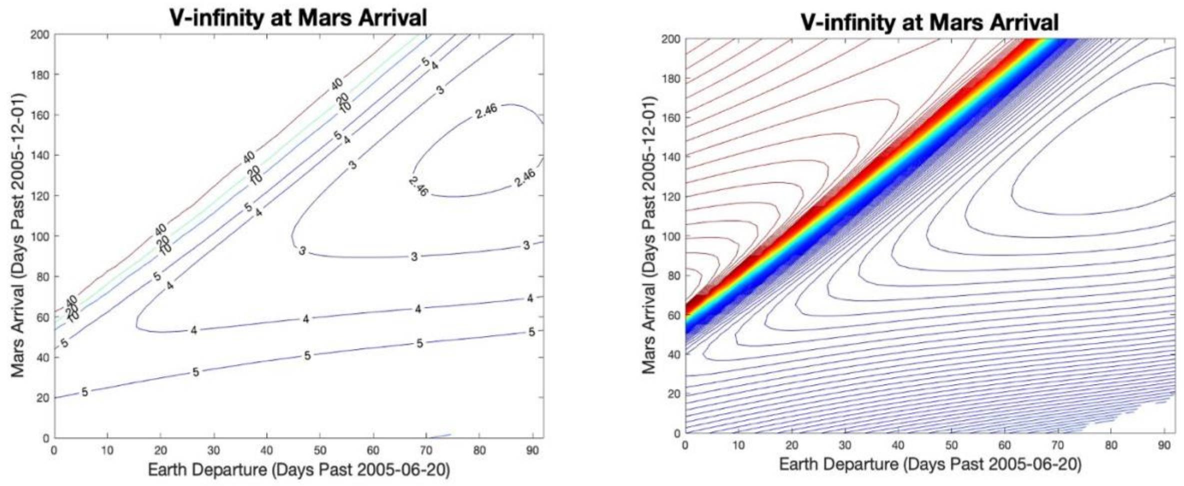


Figure 2 - V-Infinity Porkchop Plot Mars Arrival

c) Analyze and discuss the recovered subset of the transfer design space for trajectories from the Earth to Mars across the provided range of departure and arrival dates using the porkchop plots you have constructed. How could this transfer design space be expanded further in future work?

Analysis and Discussion:

The first note about these graphs is that all these values are for transfers were $\Delta\theta^* < 180°$. There are still another set of solutions not calculated where $\Delta\theta^* > 180°$.

Values to note are the v-inifinity values for both plots at the local minima. For earth departure v-infinity is about 4 km/s around 1.5 months after the initial departure dat. For the Mars arrival, V-infinity is around 2.4 km/s around 2.5 months after the initial departure date. These values are important because they allow the trajectory engineers to optimize their launch dates in order to reduce the amount of energy expense transferring from Earth to Mars.

Another interesting section of the plots is the rapid increase in v-infinity values starting around 1 month after the initial Mars arrival date and the apparent linear pattern as the earth departure dates are incremented. This rapid increase in values might be due to the relative motion of Mars with respect to Earth. Since Earth orbits faster around the sun than Mars there will be a section along their relative orbit where transfers between the planets become more energy extensive causing this sharp rise in v-infinity values for both arrival and departure as seen in the graphs. Therefore, it is advantageous to wait for the right relative positions to launch spacecrafts and minimize the energy expensed in the process.

As mentioned in P2 part a, there were several calculated values were $TOF<TOF_P$ making the transfer arc non-elliptic. These values are graphically denoted at the bottom right of both v-infinity non-value marked contour graphs (it looks like someone took a bite out of the graph). Since these departure dates and arrival dates are closer in comparison to the other combinations of transfers, these epochs will require higher energy parabolic/hyperbolic arcs in order to meet the TOF design constraint.

Further evaluation on the design space:

To evaluate the design space further we can include in the porkchop plot:

- TOF information
- Initial and Final maneuvers information
- Geometry and orientation of departure and arrival time

Superimposing these values on the data would help engineers and analysts to justify their design solutions at pivotal team meetings.

Problem 3:

a) Use the information you calculated at the beginning and end of the transfer designed in Problem 1b) to report the $\Delta\bar{v}_1$ and $\Delta\bar{v}_2$ in the Sun-centered inertial coordinate system. Also report the time of flight along the transfer.

Reiterating the values calculated in Problem 1 part b and part d:

Time of flight between Earth and Venus:

$$TOF = Venus\ Julian\ Date - Earth\ Julian\ Date = 112\ Days = 9.6777x10^6 seconds$$

The calculated $\Delta\bar{v}_1$ and $\Delta\bar{v}_2$ in the Sun-centered inertial coordinate system:

$$\Delta\bar{v}_1 = \bar{v}_{1,f} - \bar{v}_1 = 0.41550\hat{X} - 3.3782\hat{Y} + 2.5658\hat{Z}\ km/s$$

$$\|\Delta\bar{v}_1\| = 4.2624\frac{km}{s}$$

$$\Delta\bar{v}_2 = \bar{v}_2 - \bar{v}_{2,i} = -1.2867 + 2.4660\hat{Y} + 2.0191\hat{Z}\ km/s$$

$$\|\Delta\bar{v}_2\| = 3.4370\frac{km}{s}$$

b) Provide a screenshot of the transfer, along with the orbits of Earth and Venus, in a heliocentric view looking down on the ecliptic. Indicate the direction of motion along each arc using arrows added to the screenshot or state in words the direction of motion.

In figure 1 we can observe the heliocentric view looking down on the ecliptic. The specific angular momentum is pointing out of the page and therefore the direction of motion is depicted by the orange arrows.

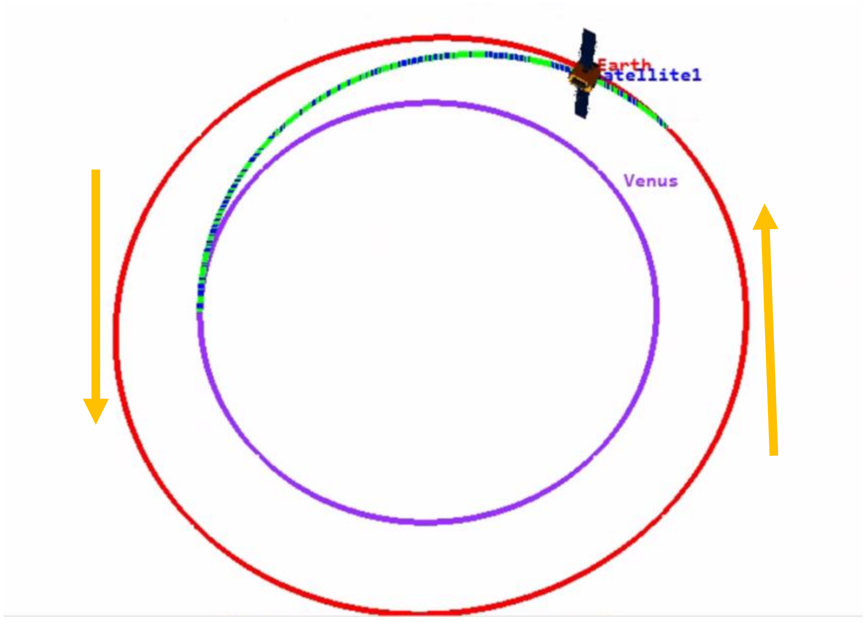


Figure 3- Heliocentric view of Earth And Venus - looking down on the ecliptic

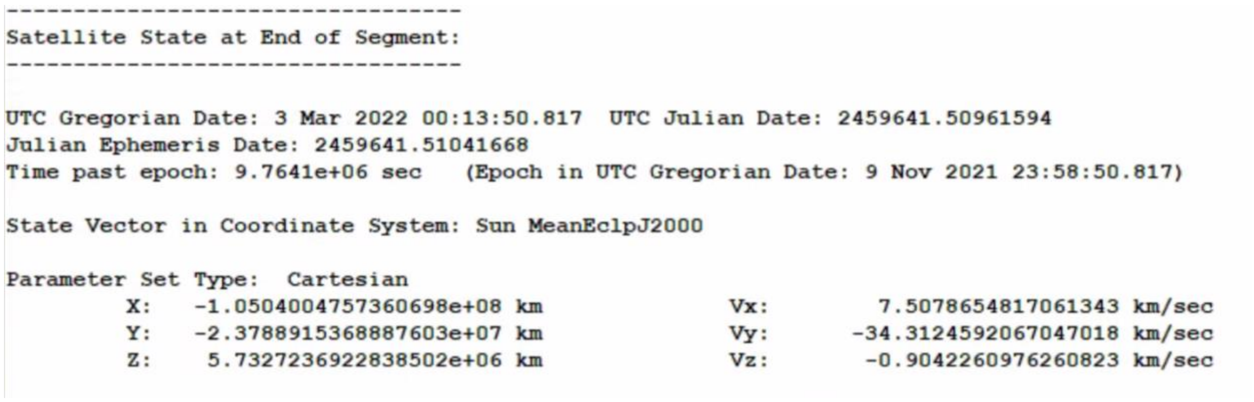
c) Use the summary or reporting function described in the lab instructions to list the state of the spacecraft in the Sun-centered inertial coordinate system after the second maneuver has been applied. Compare the state components to the $R2$ and $V2$ vectors provided in Problem 1. If there are any differences between the quantities, discuss a potential reason for this difference.

State vectors from problem 1:

Venus Sun-Centered inertial coordinates: (At a Julian date of 2459640.5 TDB)

$$\bar{r}_2 = -1.05048x10^8\hat{X} - 2.37576x10^7\hat{Y} + 5.73539x10^6\hat{Z}\ km$$
$$\bar{v}_2 = 7.49784\hat{X} - 34.31464\hat{Y} - 0.90369\hat{Z}\ km/s$$

State vectors from the STK results:



```
---------------------------------
Satellite State at End of Segment:
---------------------------------

UTC Gregorian Date: 3 Mar 2022 00:13:50.817  UTC Julian Date: 2459641.50961594
Julian Ephemeris Date: 2459641.51041668
Time past epoch: 9.7641e+06 sec   (Epoch in UTC Gregorian Date: 9 Nov 2021 23:58:50.817)

State Vector in Coordinate System: Sun MeanEclpJ2000

Parameter Set Type:  Cartesian
         X:   -1.0504004757360698e+08 km             Vx:        7.5078654817061343 km/sec
         Y:   -2.3788915368887603e+07 km             Vy:      -34.3124592067047018 km/sec
         Z:    5.7327236922838502e+06 km             Vz:       -0.9042260976260823 km/sec
```

Figure 4 - STK State vectors at the end of second maneuver

Comparing both set of values we can see all the values close within some rounding errors.

d) Provide a screenshot of the converged transfer, along with the orbits of Earth and Venus, in a heliocentric view looking down on the ecliptic. Indicate the direction of motion along each arc using arrows. Use the summary or reporting function described in the lab instructions to list the state of the spacecraft in the Sun-centered inertial coordinate system after the second maneuver has been applied – does this state vector lie within the specified tolerance of the desired state vector? Also list the $\Delta\bar{v}_1$ and $\Delta\bar{v}_2$ in the Sun-centered inertial coordinate system as computed by the targeter and compare these vectors to the quantities you reported in Problem 3a). How many iterations did the targeter require to recover this solution?

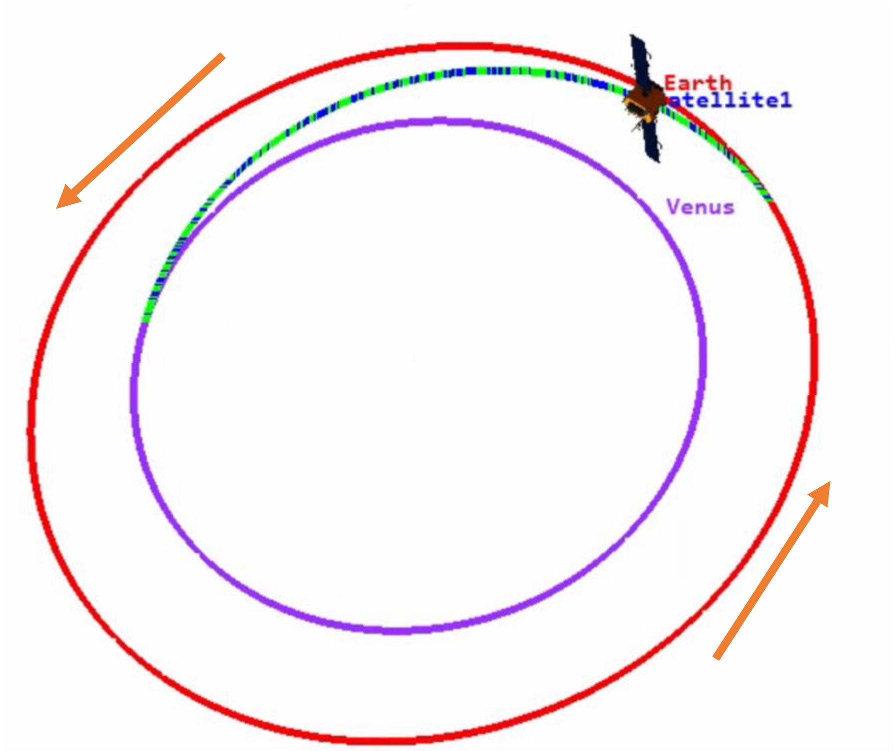
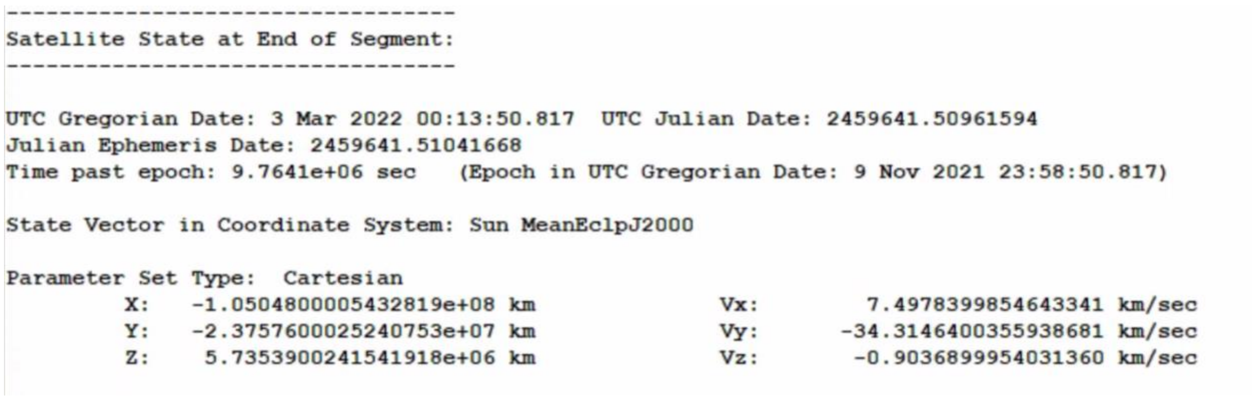


Figure 5 - Heliocentric view of Earth and Venus - looking down on the ecliptic - After the STK targeter converged



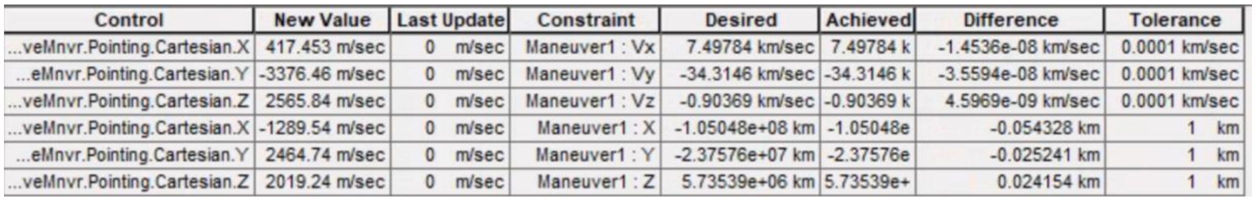
```
---------------------------------
Satellite State at End of Segment:
---------------------------------

UTC Gregorian Date: 3 Mar 2022 00:13:50.817  UTC Julian Date: 2459641.50961594
Julian Ephemeris Date: 2459641.51041668
Time past epoch: 9.7641e+06 sec   (Epoch in UTC Gregorian Date: 9 Nov 2021 23:58:50.817)

State Vector in Coordinate System: Sun MeanEclpJ2000

Parameter Set Type:  Cartesian
         X:   -1.0504800005432819e+08 km             Vx:        7.4978399854643341 km/sec
         Y:   -2.3757600025240753e+07 km             Vy:      -34.3146400355938681 km/sec
         Z:    5.7353900241541918e+06 km             Vz:       -0.9036899954031360 km/sec
```

Figure 6 - STK State vectors at the end of second maneuver after the STK targeter converged

Venus Sun-Centered inertial coordinates: (At a Julian date of 2459640.5 TDB)

$$\bar{r}_2 = -1.05048x10^8\hat{X} - 2.37576x10^7\hat{Y} + 5.73539x10^6\hat{Z}\ km$$
$$\bar{v}_2 = 7.49784\hat{X} - 34.31464\hat{Y} - 0.90369\hat{Z}\ km/s$$

The numbers are within tolerance by observing the above quantities. Also, in the given Targeter report after it converges:



Control	New Value	Last Update	Constraint	Desired	Achieved	Difference	Tolerance
...veMnvr.Pointing.Cartesian.X	417.453 m/sec	0 m/sec	Maneuver1 : Vx	7.49784 km/sec	7.49784 k	-1.4536e-08 km/sec	0.0001 km/sec
...eMnvr.Pointing.Cartesian.Y	-3376.46 m/sec	0 m/sec	Maneuver1 : Vy	-34.3146 km/sec	-34.3146 k	-3.5594e-08 km/sec	0.0001 km/sec
...veMnvr.Pointing.Cartesian.Z	2565.84 m/sec	0 m/sec	Maneuver1 : Vz	-0.90369 km/sec	-0.90369 k	4.5969e-09 km/sec	0.0001 km/sec
...veMnvr.Pointing.Cartesian.X	-1289.54 m/sec	0 m/sec	Maneuver1 : X	-1.05048e+08 km	-1.05048e	-0.054328 km	1 km
...eMnvr.Pointing.Cartesian.Y	2464.74 m/sec	0 m/sec	Maneuver1 : Y	-2.37576e+07 km	-2.37576e	-0.025241 km	1 km
...veMnvr.Pointing.Cartesian.Z	2019.24 m/sec	0 m/sec	Maneuver1 : Z	5.73539e+06 km	5.73539e+	0.024154 km	1 km

Figure 7 - STK Targeter convergence summary

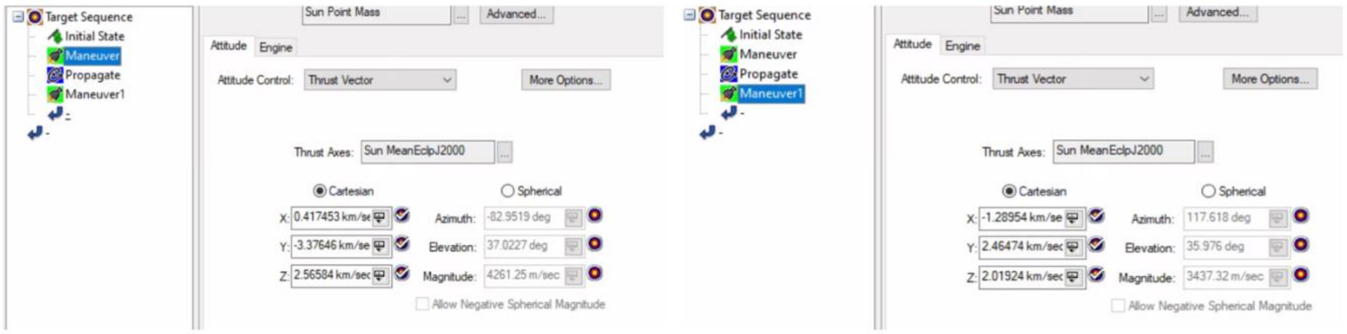
<u>Maneuvers $\Delta\bar{v}$ comparison:</u>

Calculate delta-vs results from P3 a:

$$\Delta\bar{v}_1 = \bar{v}_{1,f} - \bar{v}_1 = 0.41550\hat{X} - 3.3782\hat{Y} + 2.5658\hat{Z}\ km/s$$

$$\Delta\bar{v}_2 = \bar{v}_2 - \bar{v}_{2,i} = -1.2867 + 2.4660\hat{Y} + 2.0191\hat{Z}\ km/s$$

Targeter delta-v values after convergence:



Target Sequence: Initial State, Maneuver, Propagate, Maneuver1

Sun Point Mass | Advanced...

Attitude | Engine

Attitude Control: Thrust Vector | More Options...

Thrust Axes: Sun MeanEclpJ2000

[✓ Cartesian] Spherical

X: 0.417453 km/se | Azimuth: -82.9519 deg
Y: -3.37646 km/se | Elevation: 37.0227 deg
Z: 2.56584 km/sec | Magnitude: 4261.25 m/sec

Allow Negative Spherical Magnitude

Target Sequence: Initial State, Maneuver, Propagate, Maneuver1

Sun Point Mass | Advanced...

Attitude | Engine

Attitude Control: Thrust Vector | More Options...

Thrust Axes: Sun MeanEclpJ2000

[✓ Cartesian] Spherical

X: -1.28954 km/se | Azimuth: 117.618 deg
Y: 2.46474 km/sec | Elevation: 35.976 deg
Z: 2.01924 km/sec | Magnitude: 3437.32 m/sec

Allow Negative Spherical Magnitude

Figure 8 - STK Targeter delta-v values after convergence

The values are close with the greatest difference existing in the third decimal digit. This small change in values might be the result of rounding procedures in the algorithm carried over different calculations.

<u>How many iterations did the targeter require to recover this solution?</u>

The targeter converged in 14 Iterations.

```
%Gustavo Grinsteins
%ASEN 5050
%HW6 - Defining Lamberts Equation

function F = LambertEqt(mu,a,s,c,TOF,less180,greatThanTOFmin)

alpha = 2*asin(sqrt((s)/(2*a)));
beta = 2*asin(sqrt((s-c)/(2*a)));
n = sqrt(mu/a^3);

%perform sign checks
if less180
    %beta = beta;
else
    beta = -beta;
end

if greatThanTOFmin
    alpha = 2*pi - alpha;
else
    %alpha = alpha;
end

%define the entire equation set to zero
F = (1/n)*((alpha-beta)-(sin(alpha)-sin(beta)))-TOF;
```

```
%Gustavo Grinsteins
%ASEN 5050
%HW6
%Solving lambert's equation

function aCalc = EllOrbitLambertEqSolve(s,c,a_m,TOF,TOFmin,mu,less180)
    %STEP 5: Usin Fsolve
    %Initial Guess for a
    a_initial = a_m+150;%how to defend delta_a?
    %Calc alpha boolean
    if TOF > TOFmin
        greatThanTOFmin = true;
    else
        greatThanTOFmin = false;
    end
    diff = 1000; %initial value for stopping condition
    diff2 = inf; %initial value for second stopping condition
    a_new = 0;
    iterations = 0;
    tolerance = 10^-5; %5 digit accuracy is desired
    while diff > tolerance %Convergence stopping condition
        %implement Fsolve function
        %define the anonymous function handle
        options = optimoptions('fsolve','Display','off');
        a_bef = a_new;
        a_new = fsolve(@(a)LambertEqt(mu,a,s,c,TOF,less180,greatThanTOFmin),a_initial,↙
options);
        %Recalculate values
        n_new = sqrt(mu/(a_new^3));
        alpha_0 = 2*asin(sqrt((s)/(2*a_new)));
        beta_0 = 2*asin(sqrt((s-c)/(2*a_new)));
        if less180
            beta = beta_0;
        else
            beta = -beta_0;
        end
        if TOF > TOFmin
            alpha = 2*pi - alpha_0;
        else
            alpha = alpha_0;
        end
        TOF_new = (1/n_new)*((alpha-beta)-(sin(alpha)-sin(beta)));
        diff = abs(TOF_new-TOF);
        %Divergence stopping condition
        if diff > diff2
            fprintf('Calculations started Diverging - Stopping iterations \n')
            a_new = a_bef;
            break
        end
        %Too many Iterations stopping condition
        if iterations > 300
            fprintf('Too many iterations reached, adapt your algorithm for the problem↙
\n')
            break
        end
        diff2 = diff;
        fprintf('diff = %0.12f \n',diff)
        a_initial = a_new + 150;
        iteration = iterations +1;
    end
    aCalc = a_new;
end
```

```
%Gustavo Grinsteins
%ASEN 5050
%HW6
%Solving lambert's equation v 2

function [DepVinf,ArrivVinf] = EllOrbitLambertEqSolve2(R1,R2,V1,V2,TOF,mu,less180)
    r1 = norm(R1);
    r2 = norm(R2);
    %STEP 1: Find the transfer angle (theta<180))
    if less180
        delta_ThetaStar = abs(acos(dot(R1,R2)/(r1*r2)));
    end
    %STEP 2: Calculate Geometric quantities
    c = sqrt(r1^2+r2^2-2*(r1*r2)*cos(delta_ThetaStar));
    s = (1/2)*(r1+r2+c);
    %STEP 3: Determine if the TOF is for an ellipse
    TOFp = (1/3)*sqrt(2/mu)*((s^(3/2))+((s-c)^(3/2)));
    if TOF > TOFp
        fprintf('The transfer orbit is elliptical \n')
        %STEP 4: Determine correct alpha and beta
        a_m = (s/2);
        n_m = sqrt(mu/(a_m^3));
        alpha_m = pi;
        beta_m_0 = 2*asin(sqrt((s-c)/(s)));
        less180 = true;
        if less180
            beta_m = beta_m_0;%(theta<180)
        else
            beta_m = -beta_m_0;%(theta>180)
        end
        TOFmin = (1/n_m)*((alpha_m-beta_m)-(sin(alpha_m)-sin(beta_m)));
        %STEP 5: Usin Fsolve
        %Initial Guess for a
        a_initial = a_m+150;%how to defend delta_a?
        %Calc alpha boolean
        if TOF > TOFmin
            greatThanTOFmin = true;
        else
            greatThanTOFmin = false;
        end
        diff = 1000; %initial value for stopping condition
        diff2 = inf; %initial value for second stopping condition
        a_new = 0;
        iterations = 0;
        tolerance = 10^-5; %5 digit accuracy is desired
        while diff > tolerance %Convergence stopping condition
            %implement Fsolve function
            %define the anonymous function handle
            options = optimoptions('fsolve','Display','off');
            a_bef = a_new;
            a_new = fsolve(@(a)LambertEqt(mu,a,s,c,TOF,less180,greatThanTOFmin),↙
a_initial,options);
            %Recalculate values
            n_new = sqrt(mu/(a_new^3));
            alpha_0 = 2*asin(sqrt((s)/(2*a_new)));
            beta_0 = 2*asin(sqrt((s-c)/(2*a_new)));
            if less180
                beta = beta_0;
            else
                beta = -beta_0;
            end
            if TOF > TOFmin
                alpha = 2*pi - alpha_0;
            else
                alpha = alpha_0;
            end
            TOF_new = (1/n_new)*((alpha-beta)-(sin(alpha)-sin(beta)));
            diff = abs(TOF_new-TOF);
            %Divergence stopping condition
            if diff > diff2
                fprintf('Calculations started Diverging - Stopping iterations \n')
                a_new = a_bef;
                break
            end
            %Too many Iterations stopping condition
            if iterations > 300
                fprintf('Too many iterations reached, adapt your algorithm for the↙
problem \n')
                break
            end
            diff2 = diff;
            fprintf('diff = %0.12f \n',diff)
            a_initial = a_new + 150;
            iterations = iterations +1;
        end
        %calculating v infinity
        at = a_new;
        alpha_0 = 2*asin(sqrt((s)/(2*at)));
        beta_0 = 2*asin(sqrt((s-c)/(2*at)));
        if less180
            beta = beta_0;
        else
            beta = -beta_0;
        end
        if TOF > TOFmin
            alpha = 2*pi - alpha_0;
        else
            alpha = alpha_0;
        end
        p =((4*at*(s-r1)*(s-r2))/(c^2))*sin((alpha+beta)/(2)).^2;
        %Calculating transfer speeds using f and G functions
        f = 1- (r2/p)*(1-cos(delta_ThetaStar));
        g = (r2*r1*sin(delta_ThetaStar))/(sqrt(mu*p));
        f_dot = sqrt(mu/p)*tan(delta_ThetaStar/2)*(((1-cos(delta_ThetaStar))/(p))-(1/r2)↙
-(1/r1));
        g_dot = 1-(r1/p)*(1-cos(delta_ThetaStar));
        V1_f = (1/g)*(R2-f*R1);
        DepVinf = norm(V1_f-V1);%Earth Departing Vinf
        V2_i = (f_dot)*R1+g_dot*V1_f;
        ArrivVinf = norm(V2_i-V2);%Mars Arriving Vinf

    else
        fprintf('The transfer orbit is not elliptical \n')
        DepVinf = NaN;%Earth Departing Vinf
        ArrivVinf = NaN;%Mars Arriving Vinf
    end
```

```
end
```

```
%Gustavo Grinsteins
%ASEN 5050
%HW6

%House Keeping
clc;
clear;
%% Problem 1
% %Given
mu_sun = 1.32712428*10^11;
%earth vectors
R1 = [1.00078*10^8,1.09250*10^8,-5.29404*10^3];
V1 = [-22.46086,20.00474,-1.79921*10^-4];
JulianDateEarth = 2459528.5;
%venus vectors
R2 = [-1.05048*10^8,-2.375576*10^7,5.73539*10^6];
V2 = [7.49784,-34.31464,-0.90369];
r1 = norm(R1);
r2 = norm(R2);
JulianDateVenus = 2459640.5;
%STEP 1: Find the transfer angle (theta<180))
less180 = true;
delta_ThetaStar = abs(acos(dot(R1,R2)/(r1*r2)));
%STEP 2: Calculate Geometric quantities
c = sqrt(r1^2+r2^2-2*(r1*r2)*cos(delta_ThetaStar));
s = (1/2)*(r1+r2+c);
%STEP 3: Determine if the TOF is for an ellipse
TOFp = (1/3)*sqrt(2/mu_sun)*((s^(3/2))+((s-c)^(3/2)));
OrbitDays = abs(JulianDateVenus-JulianDateEarth);
OrbitSeconds = OrbitDays*86400;
if OrbitSeconds > TOFp
    fprintf('The transfer orbit is elliptical \n')
    TOF = OrbitSeconds;
else
    fprintf('The transfer orbit is not elliptical \n')
end
%STEP 4: Determine correct alpha and beta
a_m = (s/2);
n_m = sqrt(mu_sun/(a_m^3));
alpha_m = pi;
beta_m_0 = 2*asin(sqrt((s-c)/(s)));
less180 = true;
if less180
    beta_m = beta_m_0;%(theta<180)
else
    beta_m = -beta_m_0;%(theta>180)
end
TOFmin = (1/n_m)*((alpha_m-beta_m)-(sin(alpha_m)-sin(beta_m)));

at = EllOrbitLambertEqSolve(s,c,a_m,TOF,TOFmin,mu_sun,true);

%Recalc check
n = sqrt(mu_sun/(at^3));
alpha_0 = 2*asin(sqrt((s)/(2*at)));
beta_0 = 2*asin(sqrt((s-c)/(2*at)));
if less180
    beta = beta_0;
else
```

```
    beta = -beta_0;
end
if TOF > TOFmin
    alpha = 2*pi - alpha_0;
else
    alpha = alpha_0;
end
TOF_check = (1/n)*((alpha-beta)-(sin(alpha)-sin(beta)));
p =((4*at*(s-r1)*(s-r2))/(c^2))*sin((alpha+beta)/(2)).^2;
e = sqrt(1-(p/at));
mech_e =-(mu_sun)/(2*at);
%Part D - Calculating true anomalies
thetaS_1 = abs(acos((1/e)*((p/r1)-1)));
thetaS_2 = abs(acos((1/e)*((p/r2)-1)));
%Calculating speeds using f and G functions
f = 1- (r2/p)*(1-cos(delta_ThetaStar));
g = (r2*r1*sin(delta_ThetaStar))/(sqrt(mu_sun*p));
f_dot = sqrt(mu_sun/p)*tan(delta_ThetaStar/2)*(((1-cos(delta_ThetaStar))/(p))-(1/r2)-(1↙
/r1));
g_dot = 1-(r1/p)*(1-cos(delta_ThetaStar));
V1_f = (1/g)*(R2-f*R1);
v1_f = norm(V1_f);
V2_i = (f_dot)*R1+g_dot*V1_f;
v2_i = norm(V2_i);
delta_V1 = V1_f - V1;
delta_V2 = V2 - V2_i;

%% Problem 2
%Load imported data
%JD X Y Z Vx Vy Vz (columns)
load('EarthEphemMatrix.mat');
load('MarsEphemMatrix.mat');
mu_sun = 1.32712428*10^11;
less180 = true;

%Create a nested for loop to access the imported data
for i = 1:length(HW6EphemEarth1)
    EarthJD = HW6EphemEarth1(i,1);
    R1 = [HW6EphemEarth1(i,2),HW6EphemEarth1(i,3),HW6EphemEarth1(i,4)];
    V1 = [HW6EphemEarth1(i,5),HW6EphemEarth1(i,6),HW6EphemEarth1(i,7)];
    for j = 1:length(HW6EphemMars)
       MarsJD = HW6EphemMars(j,1);
       TOF = (MarsJD - EarthJD)*86400; %days to seconds
       R2 = [HW6EphemMars(j,2),HW6EphemMars(j,3),HW6EphemMars(j,4)];
       V2 = [HW6EphemMars(j,5),HW6EphemMars(j,6),HW6EphemMars(j,7)];
       [EarthVinf,MarsVinf] = EllOrbitLambertEqSolve2(R1,R2,V1,V2,TOF,mu_sun,less180);
       EarthDepartingVelocities(i,j) = EarthVinf;
       MarsArravingVelocities(i,j) = MarsVinf;
    end
end

%creating the pork chop plot for earth departing
figure(1)
X = 0:2:(HW6EphemEarth1(end,1)-HW6EphemEarth1(1,1));
Y = 0:5:(HW6EphemMars(end,1)-HW6EphemMars(1,1));
%peaks = EarthDepartingVelocities(:);
%peaks = fix(peaks);
%peaks = unique(peaks);
```

```
%peaks = peaks(1:2:end);
%peaks = peaks(~isnan(peaks));
peaks = [3.9811 4.5 5 5.5 6 10 50 60 62 63 64 65 66];
%peaks = fix(peaks);
%peaks = unique(peaks);
contour(X,Y,EarthDepartingVelocities.',peaks,'ShowText','on','LabelSpacing',300)
colormap winter
title('V-infinity at Earth departure','FontSize',20)
xlabel('Earth Departure (Days Past 2005-06-20)','FontSize',15)
ylabel('Mars Arrival (Days Past 2005-12-01)','FontSize',15)

figure(2)
X = 0:2:(HW6EphemEarth1(end,1)-HW6EphemEarth1(1,1));
Y = 0:5:(HW6EphemMars(end,1)-HW6EphemMars(1,1));
contour(X,Y,EarthDepartingVelocities.',400)
colormap winter
title('V-infinity at Earth departure','FontSize',20)
xlabel('Earth Departure (Days Past 2005-06-20)','FontSize',15)
ylabel('Mars Arrival (Days Past 2005-12-01)','FontSize',15)

%creating the pork chop plot for Mars Arrival
figure(3)
X = 0:2:(HW6EphemEarth1(end,1)-HW6EphemEarth1(1,1));
Y = 0:5:(HW6EphemMars(end,1)-HW6EphemMars(1,1));
peaks = MarsArravingVelocities(:);
peaks = fix(peaks);
peaks = unique(peaks);
%peaks = peaks(1:2:end);
%peaks = peaks(~isnan(peaks));
%peaks = fix(peaks);
%peaks = unique(peaks);
contour(X,Y,MarsArravingVelocities.',200)
colormap jet
title('V-infinity at Mars Arrival','FontSize',20)
xlabel('Earth Departure (Days Past 2005-06-20)','FontSize',15)
ylabel('Mars Arrival (Days Past 2005-12-01)','FontSize',15)

%creating the pork chop plot for Mars Arrival
figure(4)
X = 0:2:(HW6EphemEarth1(end,1)-HW6EphemEarth1(1,1));
Y = 0:5:(HW6EphemMars(end,1)-HW6EphemMars(1,1));
contour(X,Y,MarsArravingVelocities.',[2.36 3 4 5 10 20↙
40],'ShowText','on','LabelSpacing',100)
colormap jet
title('V-infinity at Mars Arrival','FontSize',20)
xlabel('Earth Departure (Days Past 2005-06-20)','FontSize',15)
ylabel('Mars Arrival (Days Past 2005-12-01)','FontSize',15)
```