Silent Horizon – Mission Report

Author: Rushi Faldu
Aerospace Engineering Undergraduate, Raleigh, NC, 27607

This report indicates the mission desing project for the class of intro to space flight. The report discusses about the complete mission desing of a robotic spacecraft launching from earth to an asteroid to explore it and study it. The overall design project was a success despite minor errors. The final result was accurate and precise.

I. Introduction

Space is vast and contains several things that are unknown to the mankind. From bright stars to the mysterious dark matter, its secrets are incomprehensible. That is why it is important for us to explore it and learn more about it. The Silent Horizon mission is one such mission that focuses on investigating and gathering intel about asteroid Halion in our Solar System. In this mission design project, a robotic spacecraft is launched from Earth and is tasked to execute a rendezvous maneuver reaching Halion. Then the spacecraft is designed to conduct a reconnaissance operation at Halion.

Mission Silent Horizon is divided into two different parts. The first part of the mission is the heliocentric transfer from Earth to Halion where the spacecraft will rendezvous with the asteroid to begin a bounded orbit around it. The next part of the mission is the science phase which is comprised of three different parts in itself. During the first part of the science phase, the spacecraft assumes a terminator orbit directly after the heliocentric transfer using a rendezvous maneuver. This terminator orbit is situated at a minimum altitude of 1 km above Halion's surface. This distance is large enough for the asteroid's gravity to be neglected. During this part, the spacecraft orbits Halion 10 times conducting reconnaissance and checking the components of the spacecraft for next parts. Then comes part two where the spacecraft transfers to a second terminator orbit situated at a minimum height of 50m above Halion's surface. The spacecraft then conducts reconnaissance and investigates the asteroid further for 15 orbits. The third part of the science phase are targeted passes over five difference latitudes on Halion's surface mapping it and analyzing it for future missions.

Mission Silent Horizon is designed to launch, transfer, rendezvous, and perform reconnaissance on Halion's surface within a defined interval of Jan 1, 2030, to Jan 1 ,2040. This mission design project is intended to simulate realistic mission that goes through the above-mentioned phases. Hence, this report contains a full mission design for Silent Horizon that utilizes MATLAB for realistic simulation and trajectory optimization.

II. Methodology

Before the start of the mission, a few characteristics of Earth, Sun, Halion, and the robotic spacecraft are already known. With the help of these characteristics, a simulation will be conducted in MATLAB leading to a fully designed mission report. The known characteristics for the planetary bodies are as follows:

Body	Gravitational Parameter [km^3/s^2]	Equatorial Radius [km]
Earth	3.986e5	6378
Sun	1.327124e11	696000
Halion	5e-9	0.3

Table (1): Planetary Body Characteristics

Moreover, Halion has a J2 of 2.1e-2. However, in this mission design project it is assumed that Halion has zero obliquity. Moving on to the robotic spacecraft, it has a dry mass of 850 kg and thrusters that have a specific impulse, I_{sp} of 250 seconds. To keep the spacecraft safe from crashing, this mission design makes sure that it always remains at least 50m above Halion's surface at any given time. However, the spacecraft has a constraint that it can only operate at distances between 0.5 and 2 AU from the Sun due to the limitations of the hardware onboard the spacecraft. It is also assumed in this mission design that while calculating change in mass of the spacecraft, $g = 9.81\ m/s^2$. Lastly, all calculations and simulations for this mission design are conducted in MATLAB.

The mission design begins with the science phase of Silent Horizon. This is because the spacecraft is required to conduct multiple maneuvers throughout the whole mission and the only known characteristic about

the spacecraft is its dry mass. Dry mass of 850 kg means excluding the fuel, the spacecraft will have a mass of 850 kg which includes all of its structure, hardware, and payload. Due to this we need to calculate and simulate all the orbits in science phase first to obtain the amount of propellant required during the that phase. This is will be necessary when simulating the heliocentric transfer for the spacecraft.

The design starts by calculating magnitude of radius of the first terminator orbit around asteroid. This magnitude is then used to calculate the velocity of the spacecraft during the first terminator orbit. The orbit is assumed to be a circular orbit making the calculations and simulations simpler and easy to understand. Next, the vectors for radius and velocity for the first terminator orbit are created. These vectors and magnitude represent the spacecraft's initial velocity and radius at the start of the first terminator orbit. Asteroid centered Hill frame is used for all simulations and calculations for the science phase of the mission. Lastly, using the radius magnitude, time period for one orbit around the terminator orbit is calculated. It is then multiplied by 10 to calculate the total time span for the first part of the science phase. Similar calculations are conducted then for second terminator of the science phase.

After calculating the initial cartesian vectors for both terminator orbits, the characteristics of the Hohmann transfer that occurs from terminator orbit one to two are calculated. Velocity at the pericenter and apocenter of the Hohmann transfer are calculated using the principles of Hohmann transfer. These along with the initial position and velocity, when entering the transfer orbit, are stored as vectors in MATLAB. Next, time period for the transfer orbit is calculated which is halved and stored as the time span for transfer as it is only half an orbit long. The delta v1 and v2 for Hohmann transfer are calculated and stored as vector.

Lastly, as the end of first and second parts of the science phase, gradient of J2 perturbation is calculated for the asteroid. This gradient along with the with the initial positions and velocities for both the terminator orbits is then plugged into the ode45 function and orbitJ2.m. The output of this is a simulated orbit with both position and velocity as time passes for 10 orbits of first terminator orbit and 15 orbits of the second terminator orbit. The tolerance for this simulation is set to 1e-12 to ensure as much accuracy in the results as possible.

Diving into part 3, of the science, the passes are conducted over the targeted latitudes of positive 60, positive 30, 0, negative 30, and negative 60 degrees. In this part the space will transfer at the first node from the second terminator orbit to the desired passing orbit. Then at the second node, the spacecraft will transfer back to the second terminator orbit marking the end of the passing orbit. The altitude of these passes is 30m. Next, the altitude at pericenter for these passes is calculated by adding the altitude of passing orbits and the radius of the asteroid. The pericenter of the passing orbits is located at 90 degrees from the nodes facing the Sun.

To calculate the characteristics of these transfers and passes, initial properties of the second terminator orbit are a must. To calculate the initial orbital elements for the second terminator orbit, the cartesian coordinates of the initial position and velocity of the spacecraft in that orbit are used. Cartesian2Orbital.m is used to find the orbital elements. Next, the true anomaly at the first node and the second node is stored in a variable and is used along with other orbital elements from the second terminator orbit in Orbital2Cartesian.m to find the cartesian vectors for position and velocity at the initial node and final node of the passes.

Eccentricity and semi-major axis for all the passes are calculated using the cartesian position and velocity vectors. In addition to that, time period and the time span for each pass is also calculated using the cartesian position. These will be the same for each pass as the only difference between the passes is the change in inclination of the orbit. Next, the orbital elements at the beginning of each pass are used to calculate the initial cartesian position and velocity vectors for each pass at the first node. The inclination used over here is different for all the passes whereas the other orbital elements are identical to the orbital elements of second terminator orbit as that's where the first node is located.

Using the initial position and velocity vectors passes, delta v1 for each pass is calculated and stored. Delta v1 represents the delta v for the maneuver performed at the first node to transfer from the second terminator orbit to the passing orbit over the desired latitude. Using the initial position and velocity vectors for the passes, again, each passing orbit is simulated using the ode45 and orbit.m function in MATLAB. This results in an output of position and velocity of the spacecraft throughout the whole passing orbit. Using this information, the velocity of the spacecraft at the end of the passing orbit is extracted and used to calculate the delta v2 for the passing orbits. Delta v2 represents the delta v for the maneuver performed at the end node of the passing orbit to transfer from the passing orbit to the second terminator orbit.

At the end of the science phase, these simulated orbits are plotted in 3D in MATLAB along with the asteroid body to showcase the exact trajectory of the spacecraft. This plot makes it easier to understand the spacecraft trajectory in science phase and the locations where the maneuvers occur. Along with the plot, an orbit table and a maneuver table are also created. The orbit table contains all the orbital elements of both the terminator orbits and the passing orbits. The maneuver table contains the time in days from the beginning of the science phase at which all 12 of the maneuvers occur, position of the spacecraft when these maneuvers occur, delta v vector and magnitude for each maneuver, and the mass of the spacecraft after each maneuver.

The time in the maneuver table is calculated using the time periods for the terminator orbits and the passing orbits. Time spent for each part of the science phase is then iterated by adding to the previous time until all the orbits are complete. This results in an array containing time at which each maneuver occurs. The mass after each maneuver is calculated in a similar manner using the equation as shown below.

$$m_o = m_f e^{\left(\frac{\Delta V}{g * I_{sp}}\right)} \quad (1)$$

In the above equation, m_o stands for initial mass and m_f stands for final mass. One thing to note in the equation is that letter e stands for exponent and not eccentricity. To calculate mass at each maneuver, we start from the last maneuver iterating backwards using the equation to the first maneuver calculating the initial masses for all the delta v maneuvers along the way.

Lastly, before ending the science phase, two checks are conducted to ensure the safety of the spacecraft in the terminator orbits. A simulation of the spacecraft in terminator orbit one and two is done over a time span of one month using the ode45 and orbitJ2.m function. This simulates the orbit of the spacecraft in these two orbits over a month determining the effects of J2 perturbation on the spacecraft. Next, the position data from the results is extracted and normalized to find the magnitude of position at each time step. At last, the radius of the asteroid is subtracted from this magnitude resulting in a vector of altitude over the surface of the asteroid which is then plotted for each step. The plots show whether the spacecraft altitude dips below 1 km for terminator orbit one and 50 m for terminator orbit two. If it dips, the spacecraft is considered to be in a dangerous situation at a risk of crashing.

The next step in the mission design project is to create porkchop plots. For creating the porkchop plot, a launch date range and an arrival date range for the space craft is estimated from the provided window of the whole mission. Then using datetime and juliandate functions in MATLAB, these ranges are made readable to MATLAB allowing further using in calculations. To create a porkchop plot, we first need to calculate delta v, v infinity, and c3 for the trajectories that are feasible. To do that, nested for loops are used for launch date range and arrival date range to match each launch date with each arrival date. Between the for loops, a test is present to determine the issues in the code. Next, to get rid of too long or too short launch and arrival date combinations, a limit for time of flight is estimated. Time of flight for each launch and arrival date combination is calculated and if the time of flight is between those limits, then only it is considered valid.

AsteroidOE.m and EarthOE.m functions are utilised to calculate the orbital elements for the asteroid and Earth at each launch arrival and launch date respectively. The mean anomaly found from these functions is then converted to true anomaly using Mean2True.m function. Using all the orbital elements, true anomaly, and the gravitational parameter of the Sun, cartesian coordinates are found for position and velocity of the spacecraft at Earth and at the asteroid for the respective launch and arrival dates. Solving lamberts problem using universal variable method, velocity vectors for both short way and long way transfers are determined leaving Earth and reaching the asteroid. Then, utilising the knowledge from patched conics, v inf leaving Earth and c3 values are found for both short and long way transfers.

Next, the frame is rotated to become an equatorial frame where the v inf reaching the asteroid is calculated along with delta v required to rendezvous with the asteroid. This is done using the theory of patched conics. All of this process is repeated for both long and short transfers. After going through all combinations, the loop ends.

Before, plotting the porkchop plots, reference dates are determined by subtracting the start of the mission window from the launch and arrival date range. This results in an array of number of days after the mission window start at the which the trajectory is taking place. Lastly, best and worst case scenarios for c3 and delta v are identified using a combination of both delta v and c3. This minimum and maximum values for this combination is calculated for both long and short transfers resulting in indexes of best and worst dates for c3

and delta v. Using the ind2sub MATLAB function and the indexes, the best and worst launch and arrival dates are found for both short and long transfers. These date values are converted to days using the reference days for the porkchop plots.

The main porkchop plot for the selected launch and arrival date range is then plotted along with a marker for best C3 and delta v locations for both long and short transfers. Next, 2 more porkchop plots are plotted that are zoomed in versions of the original porkchop plot. One for each short and long transfers. The ratio of zoom is determined by the launch windows that are desired for both types of transfers.

After plotting the porkchop plot and determining the best and worst launch and arrival days along with the launch windows, the next step is to move on to the heliocentric phase of the mission. Using the best and worst days, best and worst c3 and delta v values are found by indexing into the previously calculated data for both. Next, to determine the rockets for the mission, the wet mass of the spacecraft needs to be calculated. The wet mass is calculated using equation (1) where the final mass is the initial mass at start of the science phase. Similarly, delta v in equation (1) is the total delta v of all science phase maneuvers plus the rendezvous maneuver for the asteroid. As a result of the equation, we get the wet mass of the spacecraft at the beginning of the mission. This is done for all the best and worst delta v values for both long and short transfers.

Using all this data, an appropriate launch vehicle is found for the mission at:
https://elvperf.ksc.nasa.gov/Pages/Query.aspx

After that, launch and arrival dates are found and stored in a variable that can later be used to print values for future reference. Using these launch and arrival dates for best and worst scenarios, time of flight for each scenario is found in terms of days. Along with the ideal and worst dates, the launch window states are also found along with the launch window width.

Finally, a 3d plot for best short and long transfer is plotted. The plot is Sun centered in EME 2000 frame. It contains Sun, Earth, and asteroid along with orbits or asteroid and Earth around the Sun. Apart from this, trajectories of short and long transfers are also plotted for reference. The trajectory for Earth and asteroid's orbit can be found using ode45 function and the orbital elements of both the bodies at desired dates. Lastly, from this data, the v inf for departure from Earth directly into the heliocentric trajectory is calculated along with the delta v vector required for rendezvous with the asteroid at the end of the trajectory.

This marks the end of the mission as we plot the summary of the heliocentric phase in the heliocentric table which contains the short and long transfer scenarios, best and worst c3s, nominal v inf vector, nominal wet mass for the spacecraft, desired launch vehicle, ideal launch and arrival dates, ideal time of flight, minimum and maximum launch dates for the launch window, launch window width, and the delta v vector and magnitude required for the space craft to rendezvous with the asteroid at the end of the heliocentric transfer.

III. Results & Discussion

The results for this mission design project for the Silent Horizon mission are all the graphs and tables we created as we simulated and calculated the trajectories for the spacecraft in MATLAB.

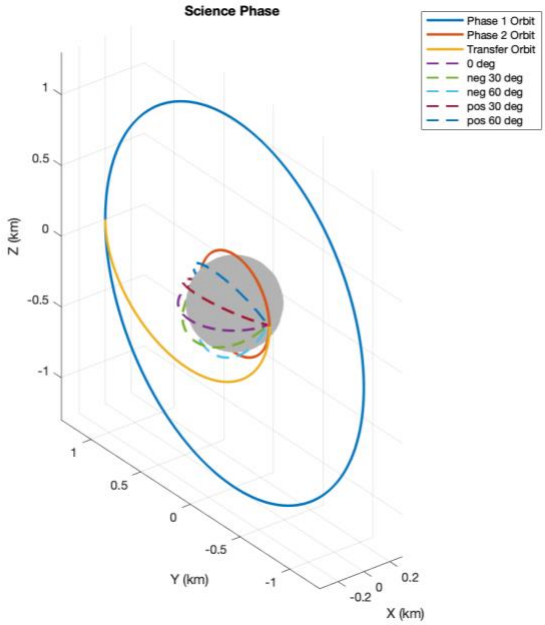


Figure (1): Science Phase Trajectories

Figure (1) shows all the trajectories for the science phase of the Silent Horizon mission. It shows how the outer terminator orbit is placed at an inclination of 90 degrees with respected to the asteroid. It also shows how the latitude passes will be facing the Sun. It mentions the transfer trajectory for the Hohmann transfer that occurs when the spacecraft transfers from terminator orbit 1 to 2. Lastly, we see that the trajectories are placed in a LVLH frame.

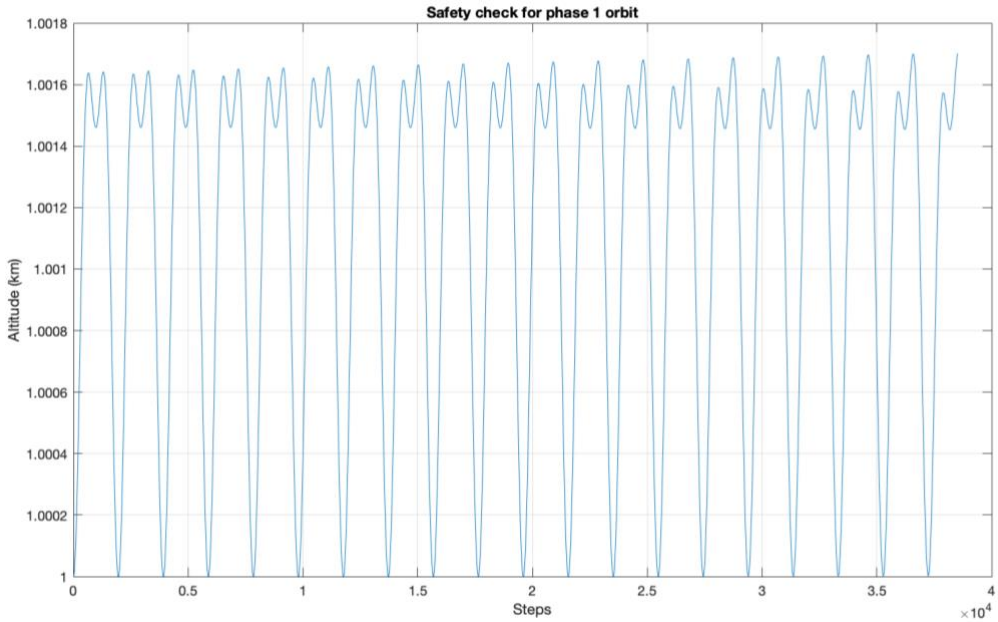


Figure (2): Safety Check for terminator orbit 1

In figure (2) we can see the safety check orbit graph. It is a graph simulated for a time span of 1 month in the termiantor orbit one. It is perturbed by J2 of the asteroid. In above graph, the altitude of the spacecraft for termiantor orbit 1 always stays above the minimum altitude for terminator orbit one which is 1 km. This indicates that the spacecraft is safe in the terminator orbit and there I no risk of crashing during the actual terminator one phase. However, the graph is not smooth indicating some error in the calcualtions while plotting the graph.

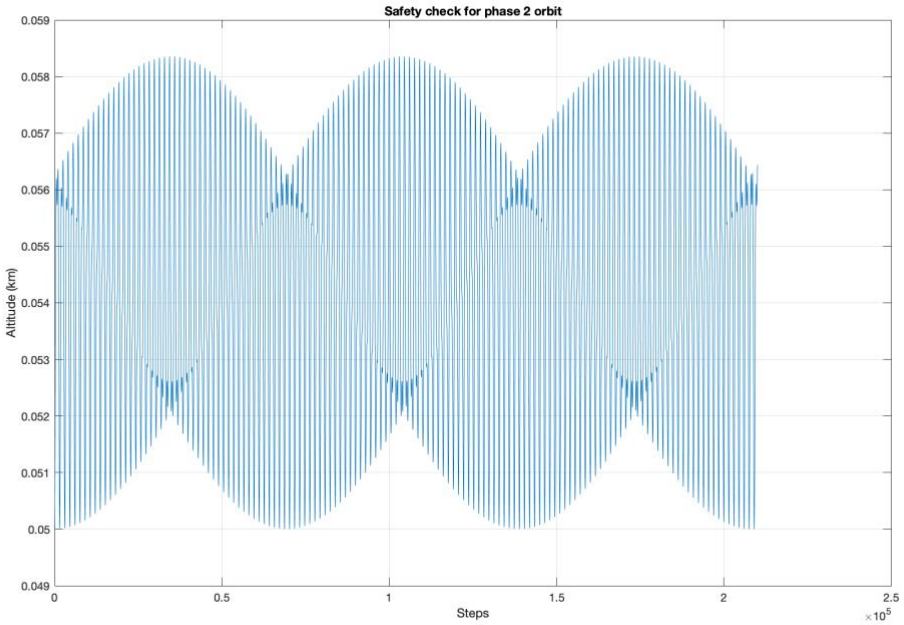


Figure (3): Safety Check for terminator orbit 2

Similar as figure (2), in figure (3) it is apparent that the spacecraft always remains above 0.05 km of altitude at all times during the one-month simulation. This indicates that the spacecraft is safe in terminator orbit 2. However, similar as figure (2), figure (3) is also not smooth indicating some error in the MATLAB code. But despite error the overall trend of both the graphs is correct.

Orbit	SMA (km)	eccentricity	Inclination (deg)	AOP (deg)	Raan (deg)	True Anomaly (deg)
Terminator Orbit 1	1.3	0	90	0	270	0
Terminator Orbit 2	0.35	0	90	0	270	0
Recon pos 60 deg	0.34	0.029411765	60	0	270	180
Recon pos 30 deg	0.34	0.029411765	30	0	270	180
Recon 0 deg	0.34	0.029411765	0	0	270	180
Recon neg 30 deg	0.34	0.029411765	-30	0	270	180
Recon neg 60 deg	0.34	0.029411765	-60	0	270	180

Table (2): Orbit Table

The orbit table gives a brief of all the orbital elements during the science phase of the mission. Important thing to notice in the table is the eccentricity. It is zero for terminator orbits meaning they are completely circular whereas the passing orbits are close to circular but not completely circular. Another important aspect is that the inclination is the only thing that is changing for the passing orbits. It starts at 90 deg indicating that the terminator orbits are polar orbits. Furthermore, the AOP and RAAN do not change at all whereas the true anomaly goes from 0 to 180 deg for the passing orbit. That is because at 180 deg, the first node of the passing phase is located.

Phase	Time (days)	Position (km)		
Terminator Orbit 1 to trasnfer	15.2439021	0	1.3	0
Transfer to Terminator Orbit 2	15.6292315	0	-0.35	0
Terminator Orbit 2 to pos 60 deg	18.8235222	6.43E-17	0.35	4.29E-17
Pos 60 deg to Terminator Orbit 2	18.9254681	-6.43E-17	-0.35	0
Terminator Orbit 2 to pos 30 deg	19.0319444	6.43E-17	0.35	4.29E-17
Pos 30 deg to Terminator Orbit 2	19.1338903	-6.43E-17	-0.35	0

Terminator Orbit 2 to 0 deg	19.2403666	6.43E-17	0.35	4.29E-17
0 deg to Terminator Orbit 2	19.3423125	-6.43E-17	-0.35	0
Terminator Orbit 2 to neg 30 deg	19.4487888	6.43E-17	0.35	4.29E-17
Neg 30 deg to Terminator Orbit 2	19.5507347	-6.43E-17	-0.35	0
Terminator Orbit 2 to neg 60 deg	19.657211	6.43E-17	0.35	4.29E-17
Neg 60 deg to Terminator Orbit 2	19.7591569	-6.43E-17	-0.35	0

Table (3): Maneuver Table

Delta V vector (km/s)			**Delta V (km/s)**	**Mass after delta v (kg)**
0	0	2.16E-05	2.16E-05	850.561466
0	0	-3.05E-05	3.05E-05	850.550883
-5.89E-05	1.10E-20	1.75E-05	6.14E-05	850.529577
-6.24E-05	-1.19E-16	1.14E-05	6.35E-05	850.507566
-0.000102	1.90E-20	6.06E-05	0.00011865	850.466421
-0.0001082	-1.18E-16	5.71E-05	0.00012229	850.424014
-0.0001178	2.19E-20	0.00011952	0.00016778	850.365835
-0.0001249	-1.20E-16	0.00011952	0.00017287	850.305899
-0.000102	1.90E-20	0.0001784	0.00020549	850.234657
-0.0001082	-1.18E-16	0.00018197	0.00021168	850.161274
-5.89E-05	1.10E-20	0.0002215	0.00022919	850.081828
-6.24E-05	-1.19E-16	0.00022768	0.00023609	850

Table (3): Maneuver Table

In the maneuver table above, we see how the mass of the spacecraft changes after it performs each delta v maneuver. However, it only changes by a small amount for the science phase due to the small size of the orbit and small delta v requirements. Furthermore, we notice that the terminator orbit 1 phase is about 15 days long and terminator orbit 2 phase is 3 days long. All the other passing orbits are less than a day long.

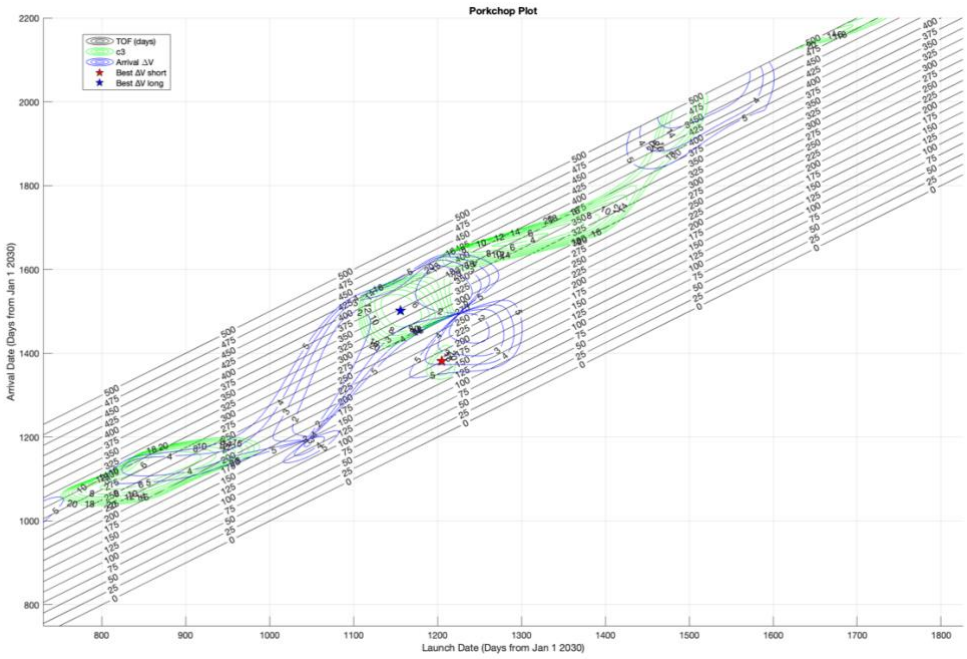


Figure (4): Complete porkchop plot

In the figure (4) above we see the complete porkchop plot. The launch date range for the plot is Jan 1, 2032 to Jan 1, 2035 and the arrival date range for it is Jan 1, 2032 to Jan 1, 2037. The reason for choosing these dates is because giving a 2 year gap in the launch dates from the mission window allows for the building and testing of the rocket and the spacecraft itself. Whereas the 3 year gap in the end is to ensure that there is ample time if something were to go wrong with the spacecraft while it is on its mission. The time of flight range is from 25 to 500 days making the mission not to long or too short. Also in the plot are 2 best dates plotted for best C3 in long way and short way transfer.

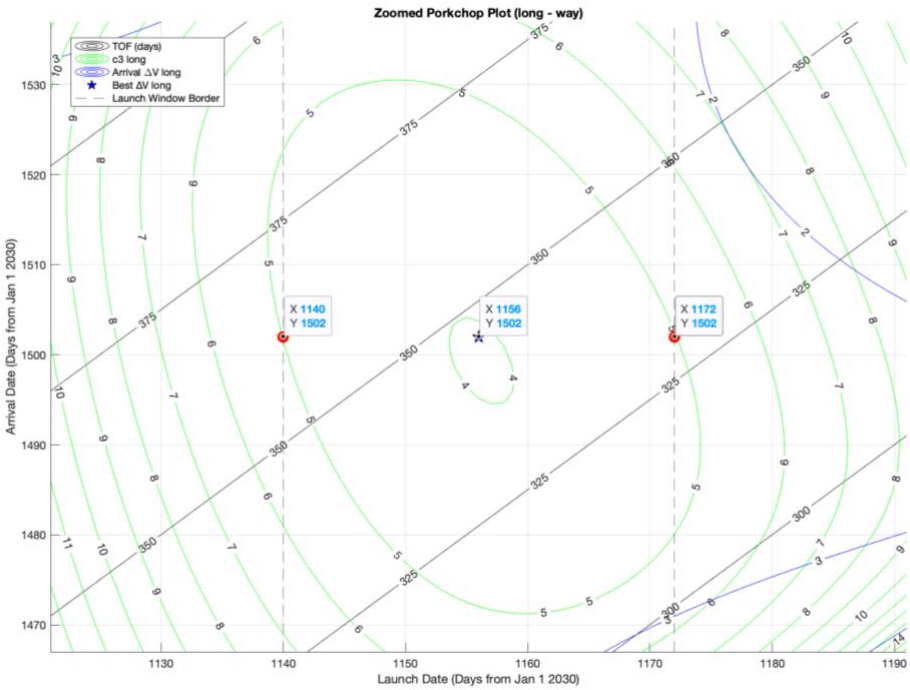


Figure (5): Porkchop plot (long way trasnfer)

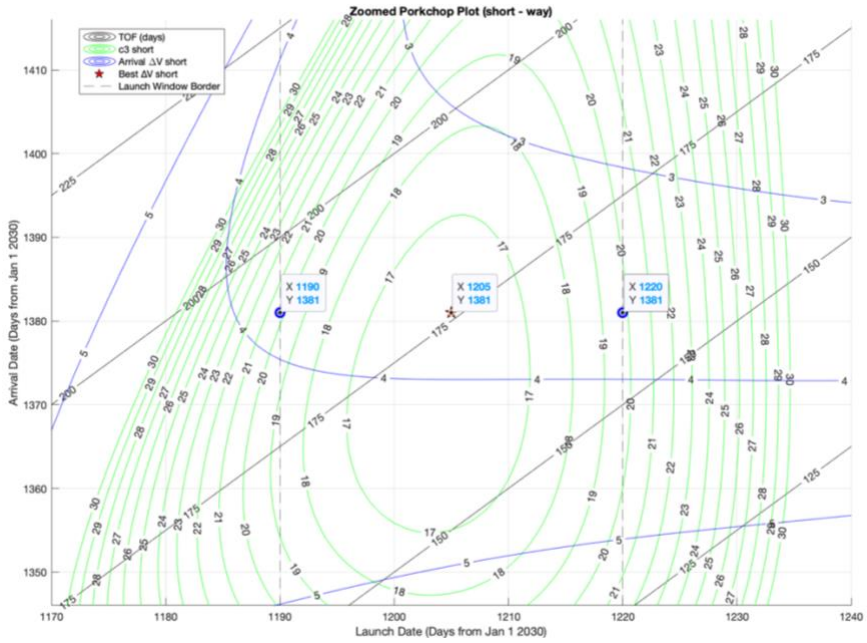


Figure (6): Porkchop plot (short way trasnfer)

For both figure (5) and (6), the launch window is in total 32 and 30 days apart from each other. This is because if the launch window is more wide, the c3 for the trajectory will change and will increase immensely. This will result in a bigger rocket requirement and will cost more making it inefficient.

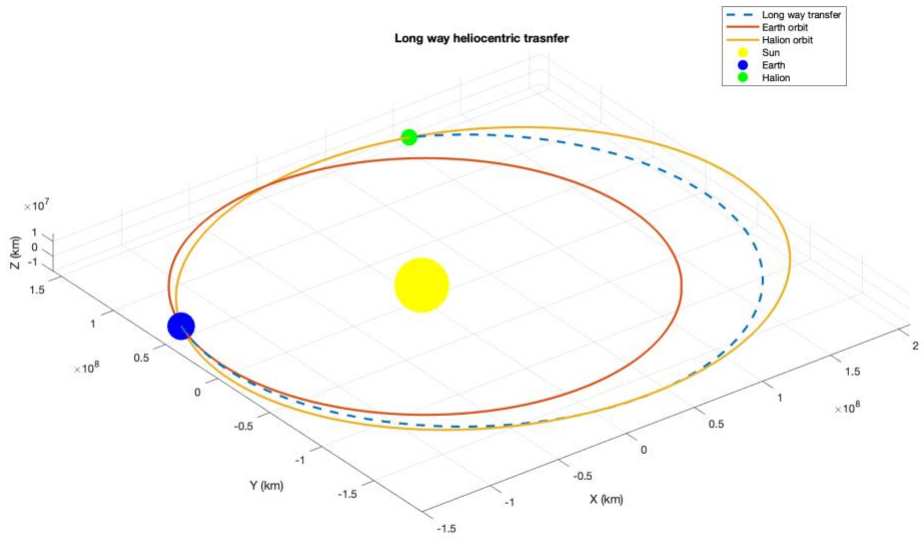


Figure (7): Long Way transfer heliocentric trajectory

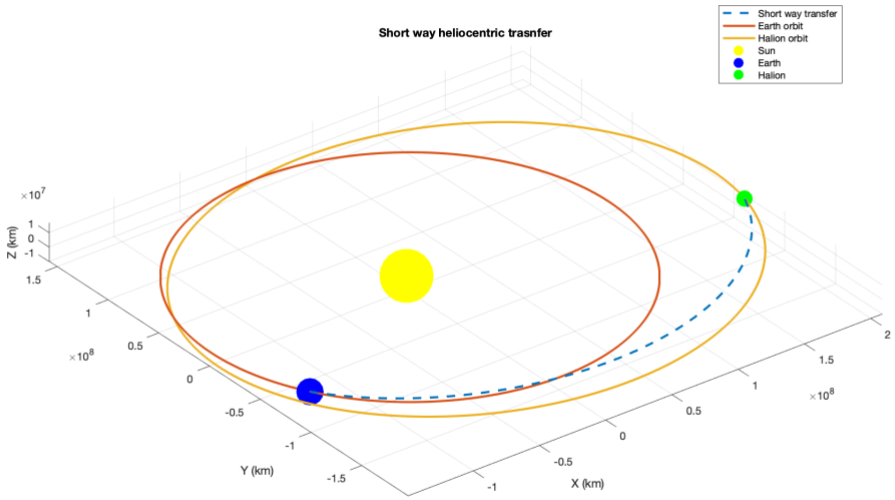


Figure (8): Short Way transfer heliocentric trajectory

Both figure (7) and (8) showcase the heliocentric long way and short way transfer trajectories. This is for the analysis and the reference.

Scenario	Nominal c3	worst c3	Nominal v inf vec		
Short Transfer	16.3112763	28074.28156	-0.0711577	-2.9172897	2.79206626
Long Transfer	3.98591755	9656.471098	-0.7055087	-1.8571198	0.19819492

Table (4): Heliocentric Table

Nominal wet mass (kg)	Launch vehicle	ideal launch date	ideal arrival date	Ideal TOF (days)	Min. launch date
3861.179316	Vulcan (VC2)	20-Apr-33	13-Oct-33	176	05-Apr-33
2242.498298	Falcon 9 (Full Thrust, ASDS)	02-Mar-33	11-Feb-34	346	14-Feb-33

Table (4): Heliocentric Table

Max. launch date	Launch window width (days)	Rendezvous Delta v vector (km/s)			Delta V (km/s)
05-May-33	30	3.55755529	-0.9220578	-0.4974505	3.70855596
18-Mar-33	32	0.95352384	1.14501094	1.8506733	2.37591135

Table (4): Heliocentric Table

According to the final heliocentric table, it is more optimal to choose long way transfer heliocentric phase for the mission. This is because long way transfer not only carries less mass reducing the cost per kilogram, but it is also using the Falcon 9 (Full thrust, ASDS) launch vehicle. That vehicle has reusable engines making it cost effective.

IV. Conclusion

In conclusion, the overall mission design project went well with accurate results. The graphs as well as the tables along with the final decision to pick the long way transfer was excellent. However, there were a few small errors with the safety orbit. Despite that, the overall mission design project was a success.

V. Appendix

I. MATLAB Code attached

Table of Contents


```
clear; clc; close all

% Mission name
mission_name = 'SILENT HORIZON';
asteroid_name = 'Halion';

% Planetary Bodies' Parameters
mu_sun = 1.327124e11; % km^3/s^2
mu_earth = 3.986e5; % km^3/s^2
mu_halion = 5e-9; % km^3/s^2
R_sun = 696000; % km
R_earth = 6378; % km
R_halion = 0.3; % km

J2_halion = 2.1e-2;

% Spacecrafts Paramaters
m_dry = 850; % kg
Isp = 250; % seconds
g = 9.81; % m/s^2
alt_safe = 0.05; % km

% Mission Constraints
AU = 149.6e6; % km
dist_min = 0.5 * AU; % km
dist_max = 2 * AU; % km
```

Science Phase

Phase 1

```
alt_high = 1; % km
num_orbits_high = 10;

r0_high = R_halion + alt_high; % km
v0_high = sqrt(mu_halion/r0_high); % km/s
r0_high_vec = [0,r0_high,0]; % km
v0_high_vec = [0,0,-v0_high]; % km
y0_high = [r0_high_vec,v0_high_vec];
```

```
Period_high = 2*pi*sqrt(r0_high^3/mu_halion);
tspan_phase1_calc = [0,Period_high*num_orbits_high];

% Phase 2
alt_low = 0.05; % km
num_orbits_low = 15;

r0_low = R_halion + alt_low; % km
v0_low = sqrt(mu_halion/r0_low); % km/s
r0_low_vec = [0,-r0_low,0]; % km
v0_low_vec = [0,0,v0_low]; % km
y0_low = [r0_low_vec,v0_low_vec];

Period_low = 2*pi*sqrt(r0_low^3/mu_halion); % s
tspan_phase2_calc = [0,Period_low*num_orbits_low]; % s

% Transfer between terminator orbits
at_terminator = (r0_high + r0_low)/2;
Periodt_terminator = pi*sqrt(at_terminator^3/mu_halion); % s
tspant_terminator = [0,Periodt_terminator]; % s

vt_terminator_apo = sqrt((2*mu_halion)/r0_high - (mu_halion/at_terminator));
vt_terminator_peri = sqrt((2*mu_halion)/r0_low - (mu_halion/at_terminator));
vt_terminator_apo_vec = [0,0,-vt_terminator_apo];
vt_terminator_peri_vec = [0,0,vt_terminator_peri];

delta_v1_vec = vt_terminator_apo_vec - v0_high_vec;
delta_v2_vec = v0_low_vec - vt_terminator_peri_vec;

delta_v1 = norm(delta_v1_vec);
delta_v2 = norm(delta_v2_vec);

r0t_terminator_vec = [0,r0_high,0];
v0t_terminator_vec = [0,0,-vt_terminator_apo];
y0t_terminator = [r0t_terminator_vec,v0t_terminator_vec];

% Simulating terminator orbits with J2
tol = 1e-12; % abs and rel tolerance

% Calculating J2 gradient for terminator orbits
syms x y z real

r = sqrt(x^2 + y^2 + z^2);
phi = asin(z/r);

RJ2 = -((mu_halion * J2_halion)/ r)*((R_halion / r)^2) *(3/2)*((sin(phi)^2) -
(1/3));

Grad_RJ2 = gradient(RJ2,[x,y,z]);
gradient_val_func = matlabFunction(Grad_RJ2, 'Vars', {x,y,z});

options = odeset('RelTol',tol,'AbsTol',tol);
[t_phase1,orbit_phase1] = ode45(@(t,y_vec)
orbit_J2(y_vec,mu_halion,gradient_val_func),tspan_phase1_calc,y0_high,options)
```

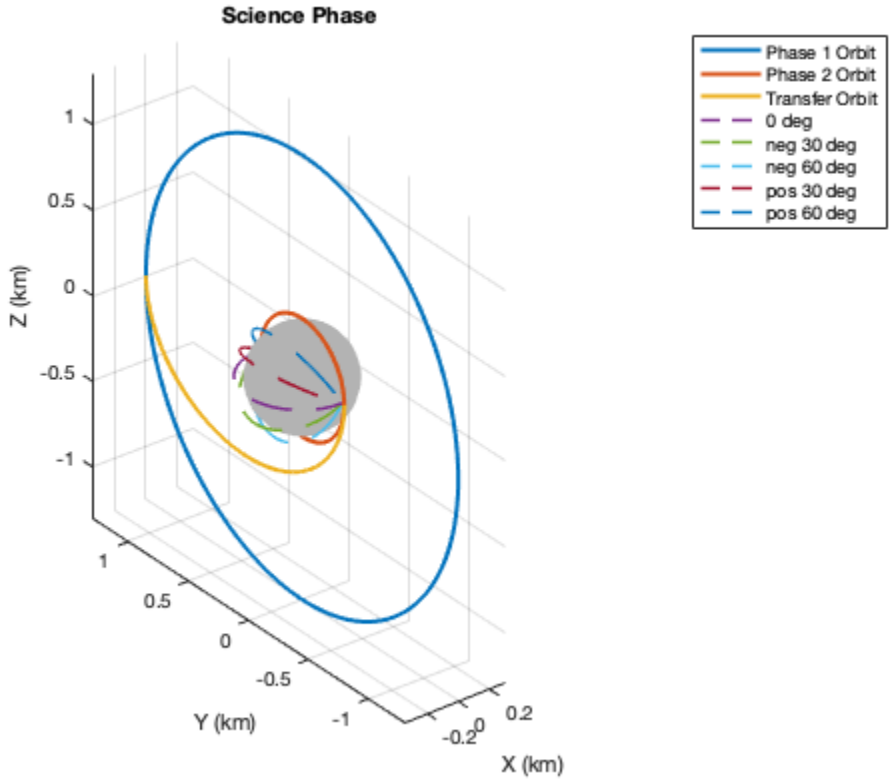
```
;
[t_phase2,orbit_phase2] = ode45(@(t,y_vec)
orbit_J2(y_vec,mu_halion,gradient_val_func),tspan_phase2_calc,y0_low,options);
[t_phase1to2,orbit_phase1to2] = ode45(@(t,y_vec)
orbit_J2(y_vec,mu_halion,gradient_val_func),tspant_terminator,y0t_terminator,o
ptions);

% Phase 3
alt_pass = 0.03; % km
rp_pass = R_halion + alt_pass; % km

[a0_low,e0_low,i0_low,w0_low,RAAN0_low,nu0_low] =
Cartesian2Orbital(r0_low_vec,v0_low_vec,mu_halion);
nui_pass = 180; % deg
nuf_pass = 0; % deg

[ri_pass_vec,vi_pass_vec] =
Orbital2Cartesian(i0_low,nui_pass,w0_low,RAAN0_low,e0_low,a0_low,mu_halion);
% Phase 3 start node
[rf_pass_vec,vf_pass_vec] =
Orbital2Cartesian(i0_low,nuf_pass,w0_low,RAAN0_low,e0_low,a0_low,mu_halion);
% Phase 3 end node

ri_pass = norm(ri_pass_vec);
e_pass = (ri_pass - rp_pass)/(rp_pass - ri_pass*cosd(nui_pass));
a_pass = rp_pass/(1 - e_pass);
Period_pass = pi*sqrt(a_pass^3 / mu_halion);
tspan_pass = [0,Period_pass];

% Vectors for passes
[r0_pass_vec_pos60,v0_pass_vec_pos60] = Orbital2Cartesian(i0_low -
30,nui_pass,w0_low,RAAN0_low,e_pass,a_pass,mu_halion);
[r0_pass_vec_pos30,v0_pass_vec_pos30] = Orbital2Cartesian(i0_low -
60,nui_pass,w0_low,RAAN0_low,e_pass,a_pass,mu_halion);
[r0_pass_vec_0,v0_pass_vec_0] = Orbital2Cartesian(i0_low -
90,nui_pass,w0_low,RAAN0_low,e_pass,a_pass,mu_halion);
[r0_pass_vec_neg30,v0_pass_vec_neg30] = Orbital2Cartesian(i0_low -
120,nui_pass,w0_low,RAAN0_low,e_pass,a_pass,mu_halion);
[r0_pass_vec_neg60,v0_pass_vec_neg60] = Orbital2Cartesian(i0_low -
150,nui_pass,w0_low,RAAN0_low,e_pass,a_pass,mu_halion);

% Delta-V1 for passes
delta_v1_pass_pos60_vec = v0_pass_vec_pos60 - vi_pass_vec;
delta_v1_pass_pos30_vec = v0_pass_vec_pos30 - vi_pass_vec;
delta_v1_pass_0_vec = v0_pass_vec_0 - vi_pass_vec;
delta_v1_pass_neg30_vec = v0_pass_vec_neg30 - vi_pass_vec;
delta_v1_pass_neg60_vec = v0_pass_vec_neg60 - vi_pass_vec;

delta_v1_pass_pos60 = norm(delta_v1_pass_pos60_vec);
delta_v1_pass_pos30 = norm(delta_v1_pass_pos30_vec);
delta_v1_pass_0 = norm(delta_v1_pass_0_vec);
delta_v1_pass_neg30 = norm(delta_v1_pass_neg30_vec);
delta_v1_pass_neg60 = norm(delta_v1_pass_neg60_vec);
```

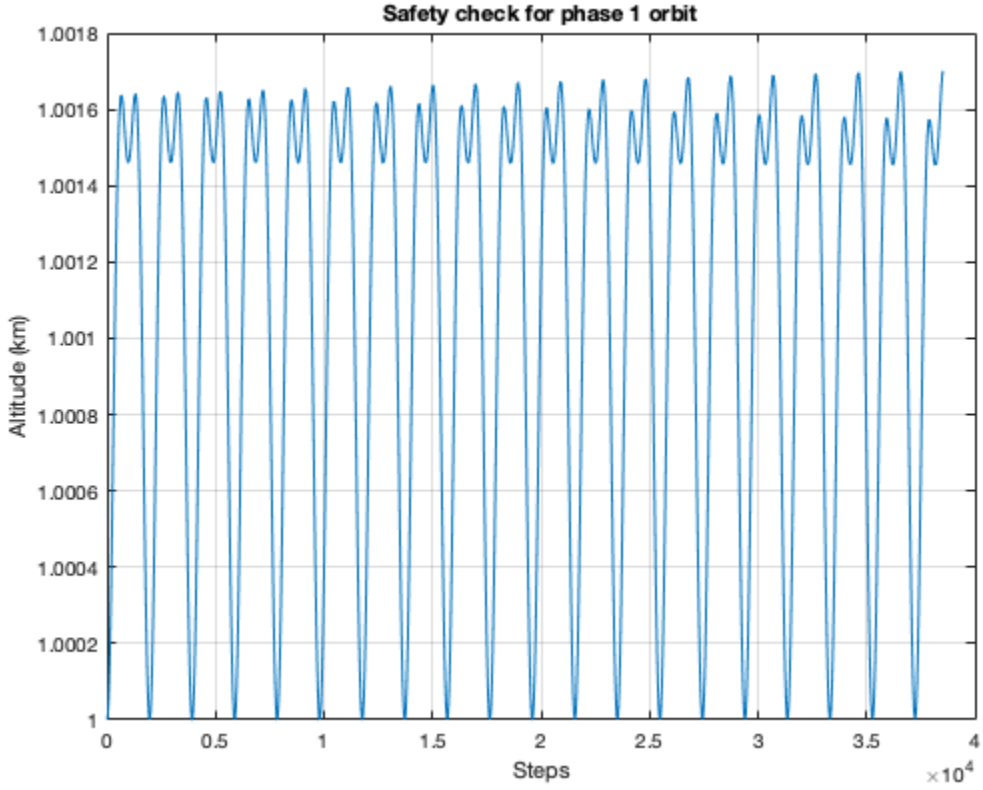
```
% Orbit simulation for passes
y0_pass_pos60 = [r0_pass_vec_pos60,v0_pass_vec_pos60];
y0_pass_pos30 = [r0_pass_vec_pos30,v0_pass_vec_pos30];
y0_pass_0 = [r0_pass_vec_0,v0_pass_vec_0];
y0_pass_neg30 = [r0_pass_vec_neg30,v0_pass_vec_neg30];
y0_pass_neg60 = [r0_pass_vec_neg60,v0_pass_vec_neg60];

[t_phase3_pos60,orbit_phase3_pos60] = ode45(@(t, y_vec) orbit(y_vec,
mu_halion), tspan_pass, y0_pass_pos60,options);
[t_phase3_pos30,orbit_phase3_pos30] = ode45(@(t, y_vec) orbit(y_vec,
mu_halion), tspan_pass, y0_pass_pos30,options);
[t_phase3_0,orbit_phase3_0] = ode45(@(t, y_vec) orbit(y_vec, mu_halion),
tspan_pass, y0_pass_0,options);
[t_phase3_neg30,orbit_phase3_neg30] = ode45(@(t, y_vec) orbit(y_vec,
mu_halion), tspan_pass, y0_pass_neg30,options);
[t_phase3_neg60,orbit_phase3_neg60] = ode45(@(t, y_vec) orbit(y_vec,
mu_halion), tspan_pass, y0_pass_neg60,options);

% Delta-V2 for passes
delta_v2_pass_pos60_vec = vf_pass_vec - (orbit_phase3_pos60(end,4:6))';
delta_v2_pass_pos30_vec = vf_pass_vec - (orbit_phase3_pos30(end,4:6))';
delta_v2_pass_0_vec = vf_pass_vec - (orbit_phase3_0(end,4:6))';
delta_v2_pass_neg30_vec = vf_pass_vec - (orbit_phase3_neg30(end,4:6))';
delta_v2_pass_neg60_vec = vf_pass_vec - (orbit_phase3_neg60(end,4:6))';

delta_v2_pass_pos60 = norm(delta_v2_pass_pos60_vec);
delta_v2_pass_pos30 = norm(delta_v2_pass_pos30_vec);
delta_v2_pass_0 = norm(delta_v2_pass_0_vec);
delta_v2_pass_neg30 = norm(delta_v2_pass_neg30_vec);
delta_v2_pass_neg60 = norm(delta_v2_pass_neg60_vec);

% Plot for science phase
figure(1)
hold on
plot3(orbit_phase1(:,1),orbit_phase1(:,2),orbit_phase1(:,3),'LineWidth',2) %
Phase 1
plot3(orbit_phase2(:,1),orbit_phase2(:,2),orbit_phase2(:,3),'LineWidth',2) %
Phase 2
plot3(orbit_phase1to2(:,1),orbit_phase1to2(:,2),orbit_phase1to2(:,3),'LineWidt
h',2) % Transfer
plot3(orbit_phase3_0(:,1),orbit_phase3_0(:,2),orbit_phase3_0(:,3),'--','LineWi
dth',1.5) % 0 deg
plot3(orbit_phase3_pos30(:,1),orbit_phase3_pos30(:,2),orbit_phase3_pos30(:,3),
'--','LineWidth',1.5) % pos 30 deg
plot3(orbit_phase3_pos60(:,1),orbit_phase3_pos60(:,2),orbit_phase3_pos60(:,3),
'--','LineWidth',1.5) % pos 60 deg
plot3(orbit_phase3_neg30(:,1),orbit_phase3_neg30(:,2),orbit_phase3_neg30(:,3),
'--','LineWidth',1.5) % neg 30 deg
plot3(orbit_phase3_neg60(:,1),orbit_phase3_neg60(:,2),orbit_phase3_neg60(:,3),
'--','LineWidth',1.5) % neg 60 deg
grid on
view(3)

[s1,s2,s3] = sphere(20);
```

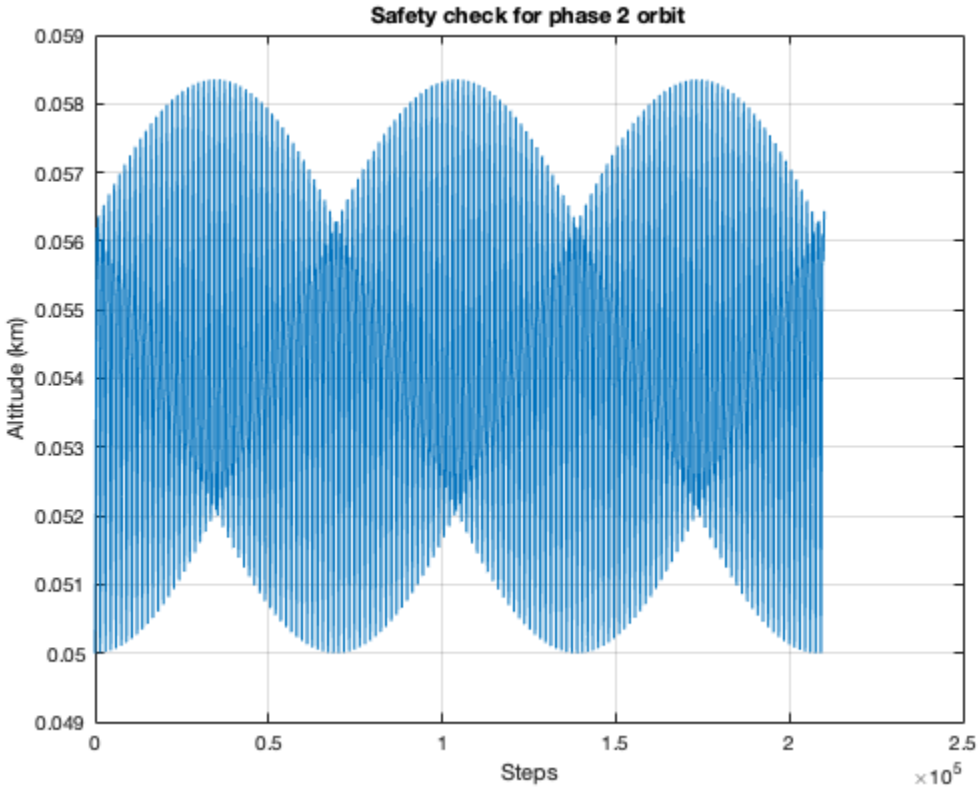
```
surf(0.3*s1,0.3*s2,0.3*s3,'FaceColor',[0.7 0.7 0.7],'EdgeColor','none')
axis equal
legend('Phase 1 Orbit','Phase 2 Orbit','Transfer Orbit','0 deg','neg 30
deg','neg 60 deg','pos 30 deg','pos 60 deg')
title('Science Phase')
xlabel('X (km)')
ylabel('Y (km)')
zlabel('Z (km)')

% Space craft Mass - Starting from last maneuver
delta_Vs_science_phase_vec =
[delta_v1_vec;delta_v2_vec;delta_v1_pass_pos60_vec';delta_v2_pass_pos60_vec';.
..

delta_v1_pass_pos30_vec';delta_v2_pass_pos30_vec';delta_v1_pass_0_vec';delta_v
2_pass_0_vec';...

delta_v1_pass_neg30_vec';delta_v2_pass_neg30_vec';delta_v1_pass_neg60_vec';del
ta_v2_pass_neg60_vec'];

delta_Vs_science_phase =
[delta_v1;delta_v2;delta_v1_pass_pos60;delta_v2_pass_pos60;...

delta_v1_pass_pos30;delta_v2_pass_pos30;delta_v1_pass_0;delta_v2_pass_0;...

delta_v1_pass_neg30;delta_v2_pass_neg30;delta_v1_pass_neg60;delta_v2_pass_neg6
0];

total_delta_v_science_phase = abs(delta_v1) + abs(delta_v2) +
abs(delta_v1_pass_pos60) + abs(delta_v1_pass_pos30) + ...
    abs(delta_v1_pass_0) + abs(delta_v1_pass_neg30) +
abs(delta_v1_pass_neg60) + abs(delta_v2_pass_pos60) + ...
    abs(delta_v2_pass_pos30) + abs(delta_v2_pass_0) +
abs(delta_v2_pass_neg30) + abs(delta_v2_pass_neg60); % km/s

initial_masses = zeros(12,1); % kg

it = 0;
for man_num = 12:-1:1
    temp_delta_v = delta_Vs_science_phase(man_num,1)*1000; % m/s
    if it >= 1
        initial_masses(man_num,1) = temp_mass*exp(abs(temp_delta_v)/(Isp*g));
        temp_mass = initial_masses(man_num,1);
    else
        initial_masses(man_num,1) = m_dry*exp(abs(temp_delta_v)/(Isp*g));
        temp_mass = initial_masses(man_num,1);
    end

    it = it + 1;
end

initial_masses = [initial_masses;m_dry];

% Maneuver Table
```

```
pos_science_phase =
[r0_high_vec;r0_low_vec;ri_pass_vec';rf_pass_vec';ri_pass_vec';rf_pass_vec';..
.

ri_pass_vec';rf_pass_vec';ri_pass_vec';rf_pass_vec';ri_pass_vec';rf_pass_vec']
;

time_science_phase = zeros(12,1);

for man_num = 1:12

    if man_num == 1
       time_science_phase(man_num,1) = Period_high*num_orbits_high;
    elseif man_num == 2
       time_science_phase(man_num,1) = temp_time + Periodt_terminator;
    elseif man_num == 3
       time_science_phase(man_num,1) = temp_time +
(Period_low*num_orbits_low);
    elseif man_num > 3 && rem(man_num,2) == 0
       time_science_phase(man_num,1) = temp_time + Period_pass;
    elseif man_num > 3 && rem(man_num,2) ~= 0
       time_science_phase(man_num,1) = temp_time + Period_low/2;
    end

    temp_time = time_science_phase(man_num,1);
end

time_science_phase = time_science_phase/(3600*24);

Phases = ["Terminator Orbit 1 to trasnfer";'Transfer to Terminator Orbit
2';'Terminator Orbit 2 to pos 60 deg';...
    'Pos 60 deg to Terminator Orbit 2';'Terminator Orbit 2 to pos 30
deg';'Pos 30 deg to Terminator Orbit 2';'Terminator Orbit 2 to 0 deg';'0 deg
to Terminator Orbit 2';'Terminator Orbit 2 to neg 30 deg';...
    'Neg 30 deg to Terminator Orbit 2';'Terminator Orbit 2 to neg 60
deg';'Neg 60 deg to Terminator Orbit 2'];

Maneuver_table =
table(string(Phases),time_science_phase,pos_science_phase,delta_Vs_science_pha
se_vec,delta_Vs_science_phase,initial_masses(2:end,1),...
    'VariableNames',{'Science Phase','Time (days)','Position (km)','Delta V
vector (km/s)','Delta V (km/s)','Mass after Delta V (kg)'});

% Orbit Table
Phases2 = ["Terminator Orbit 1";"Terminator Orbit 2";"Recon pos 60
deg";"Recon pos 30 deg";"Recon 0 deg";"Recon neg 30 deg";"Recon neg 60 deg"];
a_science_phase = [r0_high;r0_low;a_pass;a_pass;a_pass;a_pass;a_pass];
e_science_phase = [0;0;e_pass;e_pass;e_pass;e_pass;e_pass];
i_science_phase =
[90;i0_low;i0_low-30;i0_low-60;i0_low-90;i0_low-120;i0_low-150];
w_science_phase = [0;w0_low;w0_low;w0_low;w0_low;w0_low;w0_low];
RAAN_science_phase =
[RAAN0_low;RAAN0_low;RAAN0_low;RAAN0_low;RAAN0_low;RAAN0_low;RAAN0_low];
nu_science_phase = [0;nu0_low;nui_pass;nui_pass;nui_pass;nui_pass;nui_pass];
```

```
Orbit_table =
table(string(Phases2),a_science_phase,e_science_phase,i_science_phase,w_scienc
e_phase,RAAN_science_phase,nu_science_phase,...
    'VariableNames',{'Orbit','SMA (km)','eccentricity','inlcination
(deg)','AOP (deg)','RAAN (deg)','True Anomaly (deg)'});

% Safety check for terminator orbits
sim_time = [0 ,30*3600*24];
[sim_t1, pos_t1] = ode45(@(t,y_vec)
Safety_orbit_J2(y_vec,mu_halion,J2_halion,R_halion),sim_time,y0_high,options);
pos_t1_sum = sqrt(sum(pos_t1(:,1:3).^2,2));
safety_1 = pos_t1_sum - R_halion;

[sim_t2, pos_t2] = ode45(@(t,y_vec)
Safety_orbit_J2(y_vec,mu_halion,J2_halion,R_halion),sim_time,y0_low,options);
pos_t2_sum = sqrt(sum(pos_t2(:,1:3).^2,2));
safety_2 = pos_t2_sum - R_halion;

% Plot
figure (2)
plot(1:length(safety_1),safety_1)
title('Safety check for phase 1 orbit')
xlabel('Steps')
ylabel('Altitude (km)')
grid on

figure (3)
plot(1:length(safety_2),safety_2)
title('Safety check for phase 2 orbit')
xlabel('Steps')
ylabel('Altitude (km)')
grid on
```


Science Phase
Phase 1 Orbit
Phase 2 Orbit
Transfer Orbit
0 deg
neg 30 deg
neg 60 deg
pos 30 deg
pos 60 deg
Z (km)
Y (km)
X (km)


Safety check for phase 1 orbit
Altitude (km)
Steps
×10^4



Porkchop Plot

```
% Converting to datetime
launch_i = datetime(2032,1,1);
launch_f = datetime(2035,1,1);
arrival_i = datetime(2032,1,1);
arrival_f = datetime(2037,1,1);

% Date Ranges
launch_dates = launch_i:days(1):launch_f;
arrival_dates = arrival_i:days(1):arrival_f;

% Convert to Julian Dates
launch_dates_j = juliandate(launch_dates);
arrival_dates_j = juliandate(arrival_dates);

% Pre-allocation of arrays
num_rows = length(launch_dates_j);
num_col = length(arrival_dates_j);

TOF = NaN(num_rows,num_col);
c3_short = NaN(num_rows,num_col);
c3_long = NaN(num_rows,num_col);
delta_v_short = NaN(num_rows,num_col);
delta_v_long = NaN(num_rows,num_col);
```

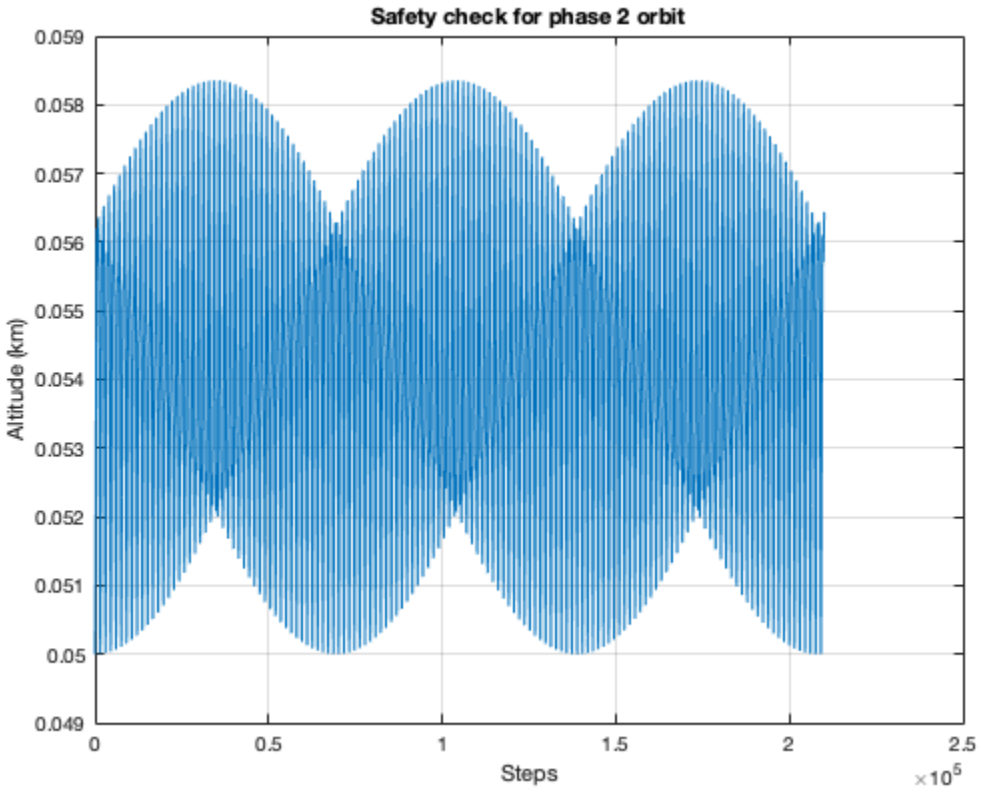
```
% Loop for iterating through each each launch date with each arrival date.
for i_date = 1:num_rows

    % To check if it is stuck in a loop or not, goes till 360. Total launch
    % dates = 366
    if mod(i_date,20) == 0
        fprintf("Processing departure date %d\n", i_date)
    end

    for j_date = 1:num_col

        % Time of flight
        tof = arrival_dates_j(j_date) - launch_dates_j(i_date); % days
        TOF(i_date,j_date) = tof; % days

        % TOF range is from 25 - 500 days
        if ~(tof >= 25 && tof <= 500)
           continue
        end

        % Orbital Elements for the Dates
        [E_a,E_e,E_i,E_Om,E_w,E_M] = EarthOE(launch_dates_j(i_date)); % Earth
        [H_a,H_e,H_i,H_Om,H_w,H_M] = AsteroidOE(arrival_dates_j(j_date)); %
Halion

        % Mean anomaly to True Anomaly
        E_nu = Mean2True(E_M,E_e); % Earth
        H_nu = Mean2True(H_M,H_e); % Halion

        % Orbital to Cartesian
        [r_vec_E, v_vec_E] =
Orbital2Cartesian(E_i,E_nu,E_w,E_Om,E_e,E_a,mu_sun); % Earth
        [r_vec_H, v_vec_H] =
Orbital2Cartesian(H_i,H_nu,H_w,H_Om,H_e,H_a,mu_sun); % Halion Asteroid

        % Lamberts Problem - Short
        [v1_vec,v2_vec,~] = LambertsSolver(r_vec_E,r_vec_H,tof,1,mu_sun);

        v_inf_E = v1_vec - v_vec_E; % Earth V_inf
        c3_short(i_date,j_date) = (norm(v_inf_E))^2;

        % Rotate v_inf_E to equatorial frame
        epsilon = deg2rad(23.44); % Obliquity of the ecliptic
        R = [1 0 0; 0 cos(epsilon) -sin(epsilon); 0 sin(epsilon)
cos(epsilon)];

        v_inf_H = v2_vec - v_vec_H; % Halion V_inf
        v_inf_H_mag = norm(v_inf_H);
        v_p_H = sqrt(2*((v_inf_H_mag^2)/2 + mu_halion/r0_high)); % Halion
velcoity at pericenter
        delta_v_short(i_date,j_date) = abs(v0_high - v_p_H);

        % Lamberts Problem - Long
```

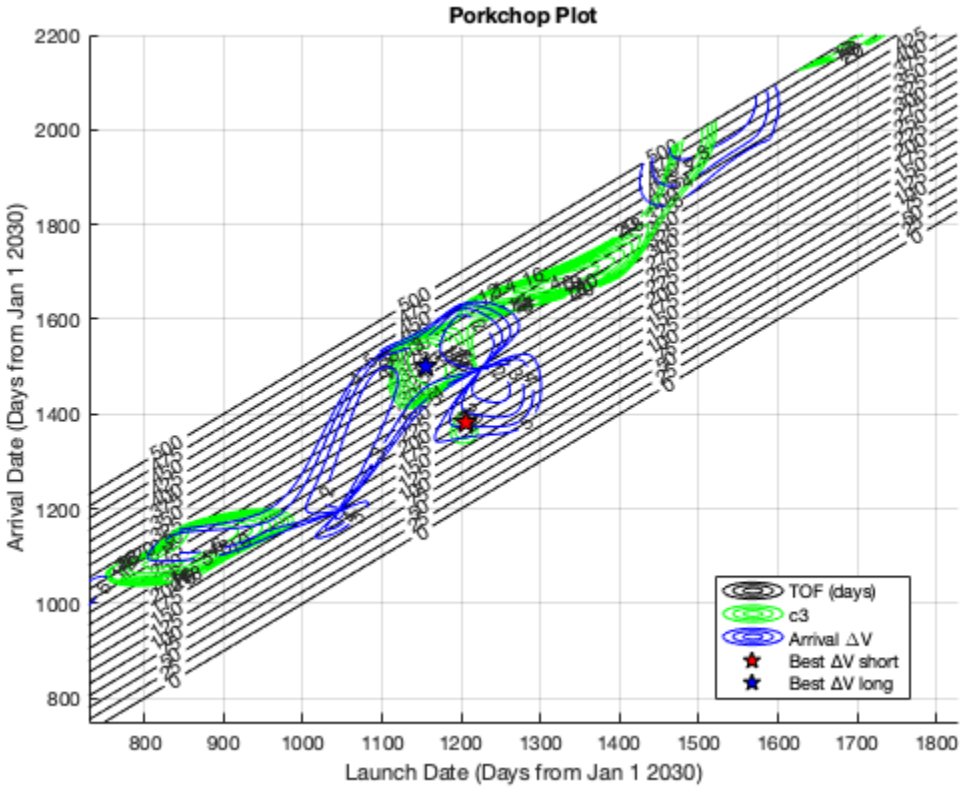
```
        [v1_vec,v2_vec,~] = LambertsSolver(r_vec_E,r_vec_H,tof,-1,mu_sun);

        v_inf_E = v1_vec - v_vec_E;  % Earth V_inf
        c3_long(i_date,j_date) = (norm(v_inf_E))^2;

        v_inf_H = v2_vec - v_vec_H; % Halion V_inf
        v_inf_H_mag = norm(v_inf_H);
        v_p_H = sqrt(2*((v_inf_H_mag^2)/2 + mu_halion/r0_high)); % Halion
velcoity at pericenter
        delta_v_long(i_date,j_date) = abs(v0_high - v_p_H);

    end
end

clc

% Reference Dates
ref_launch = datetime(2030,1,1);
ref_arrival = datetime(2030,1,1);

launch_days = days(launch_dates - ref_launch);
arrival_days = days(arrival_dates - ref_arrival);

% Best and worst case calculation
combined_short = c3_short + delta_v_short;
combined_long = c3_long + delta_v_long;

[best_val_short,index_best_short] = min(combined_short(:),[],'omitnan');
[best_val_long,index_best_long] = min(combined_long(:),[],'omitnan');
[worst_val_short,index_worst_short] = max(combined_short(:),[],'omitnan');
[worst_val_long,index_worst_long] = max(combined_long(:),[],'omitnan');

[launch_best_short,arrival_best_short] = ind2sub(size(combined_short),
index_best_short);
[launch_best_long,arrival_best_long] = ind2sub(size(combined_long),
index_best_long);
[launch_worst_short,arrival_worst_short] = ind2sub(size(combined_short),
index_worst_short);
[launch_worst_long,arrival_worst_long] = ind2sub(size(combined_long),
index_worst_long);

x_best_short = launch_days(launch_best_short);
y_best_short = arrival_days(arrival_best_short);

x_best_long = launch_days(launch_best_long);
y_best_long = arrival_days(arrival_best_long);

% Porkchop plot
figure(4)
hold on

% Plot TOF contours (black)
contour(launch_days, arrival_days, TOF', 0:25:500, '-k', 'ShowText', true);
```

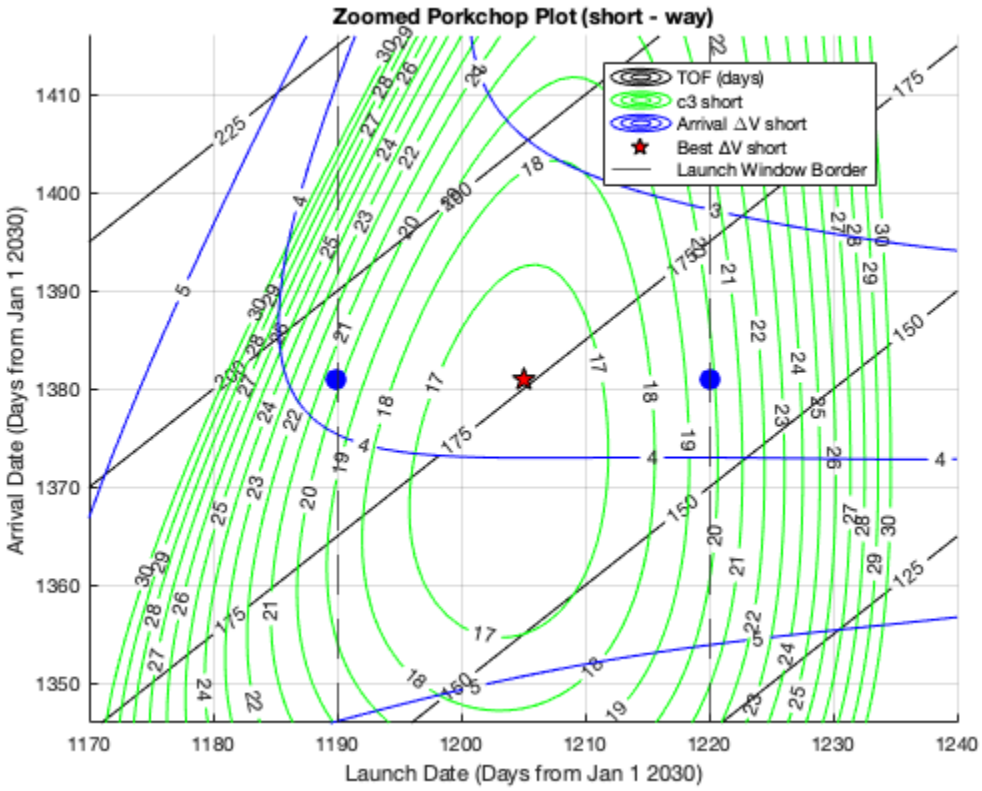
```
% Plot short-way c3 contours (green)
contour(launch_days, arrival_days, c3_short',4:2:20, '-g', 'ShowText', true);

% Plot short-way delta-V contours (blue)
contour(launch_days, arrival_days, delta_v_short',2:1:5, '-b', 'ShowText',
true);

% Plot long-way c3 contours (green)
contour(launch_days, arrival_days, c3_long',4:2:20 ,'-g', 'ShowText', true,
'HandleVisibility','off');

% Plot long-way delta-V contours (blue)
contour(launch_days, arrival_days, delta_v_long',2:1:5, '-b', 'ShowText',
true, 'HandleVisibility','off');

% Plot best values for short and long
plot(x_best_short,  y_best_short,
'kp','MarkerSize',12,'MarkerFaceColor','r','DisplayName','Best ΔV short');
plot(x_best_long,  y_best_long,
'kp','MarkerSize',12,'MarkerFaceColor','b','DisplayName','Best ΔV long');

ylim([750,2200])
xlabel('Launch Date (Days from Jan 1 2030)');
ylabel('Arrival Date (Days from Jan 1 2030)');
title('Porkchop Plot');
legend('TOF (days)', 'c3', 'Arrival \DeltaV ','Best ΔV short','Best ΔV
long' ,'Location', 'Best');
grid on;
hold off;

% zoomed porkchop plot short
figure (5)
hold on

% Plot TOF contours (black)
contour(launch_days, arrival_days, TOF', 0:25:500, '-k', 'ShowText', true);

% Plot short-way c3 contours (green)
contour(launch_days, arrival_days, c3_short',4:1:30, '-g', 'ShowText', true);

% Plot short-way delta-V contours (blue)
contour(launch_days, arrival_days, delta_v_short',2:1:5, '-b', 'ShowText',
true);

% Plot best values for short
plot(x_best_short,
y_best_short,'kp','MarkerSize',12,'MarkerFaceColor','r','DisplayName','Best
ΔV short');
plot(x_best_short - 15,
y_best_short,'bo','MarkerSize',10,'MarkerFaceColor','b','HandleVisibility','of
f')
plot(x_best_short + 15,
y_best_short,'bo','MarkerSize',10,'MarkerFaceColor','b','HandleVisibility','of
f')
```

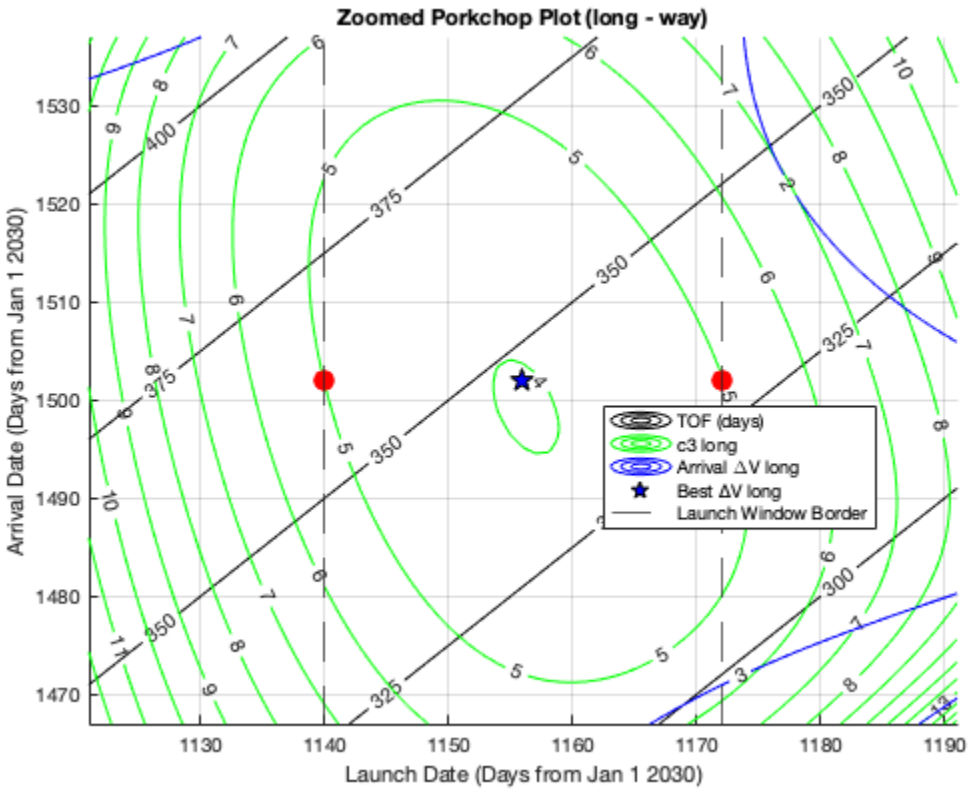
```
xline(x_best_short - 15,'--')
xline(x_best_short + 15,'--')

x_min_short = x_best_short - 35;
x_max_short = x_best_short + 35;
xlim([x_min_short x_max_short])

y_min_short = y_best_short - 35;
y_max_short = y_best_short + 35;
ylim([y_min_short y_max_short])

xlabel('Launch Date (Days from Jan 1 2030)');
ylabel('Arrival Date (Days from Jan 1 2030)');
title('Zoomed Porkchop Plot (short - way)');
legend('TOF (days)', 'c3 short', 'Arrival \DeltaV short','Best ΔV
short','Launch Window Border','Location', 'Best');
grid on;
hold off;

% zoomed porkchop plot long
figure (6)
hold on

% Plot TOF contours (black)
contour(launch_days, arrival_days, TOF', 0:25:500, '-k', 'ShowText', true);

% Plot long-way c3 contours (green)
contour(launch_days, arrival_days, c3_long',4:1 :20 ,'-g', 'ShowText', true);

% Plot long-way delta-V contours (blue)
contour(launch_days, arrival_days, delta_v_long',2:1:5, '-b', 'ShowText',
true);

% Plot best values for long
plot(x_best_long,  y_best_long,
'kp','MarkerSize',12,'MarkerFaceColor','b','DisplayName','Best ΔV long');
plot(x_best_long - 16,
y_best_long,'ro','MarkerSize',10,'MarkerFaceColor','r','HandleVisibility','off
')
plot(x_best_long + 16,
y_best_long,'ro','MarkerSize',10,'MarkerFaceColor','r','HandleVisibility','off
')
xline(x_best_long - 16,'--')
xline(x_best_long + 16,'--')

x_min_long = x_best_long - 35;
x_max_long = x_best_long + 35;
xlim([x_min_long x_max_long])

y_min_long = y_best_long - 35;
y_max_long = y_best_long + 35;
ylim([y_min_long y_max_long])

xlabel('Launch Date (Days from Jan 1 2030)');
```

```
ylabel('Arrival Date (Days from Jan 1 2030)');
title('Zoomed Porkchop Plot (long - way)');
legend('TOF (days)', 'c3 long', 'Arrival \DeltaV long','Best ΔV long','Launch
Window Border','Location', 'Best');
grid on;
hold off;
```




Porkchop Plot
Arrival Date (Days from Jan 1 2030)
Launch Date (Days from Jan 1 2030)
TOF (days)
c3
Arrival ΔV
Best ΔV short
Best ΔV long


Zoomed Porkchop Plot (short - way)
Arrival Date (Days from Jan 1 2030)
Launch Date (Days from Jan 1 2030)
TOF (days)
c3 short
Arrival ΔV short
Best ΔV short
Launch Window Border



Heliocentric Phase

Best Delta V and c3

```
best_delta_v_short = delta_v_short(launch_best_short,arrival_best_short);
best_delta_v_long = delta_v_long(launch_best_long,arrival_best_long);

best_c3_short = c3_short(launch_best_short,arrival_best_short);
best_c3_long = c3_long(launch_best_long,arrival_best_long);

% Worst Delta V and c3
worst_delta_v_short = delta_v_short(launch_worst_short,arrival_worst_short);
worst_delta_v_long = delta_v_long(launch_worst_long,arrival_worst_long);

worst_c3_short = c3_short(launch_worst_short,arrival_worst_short);
worst_c3_long = c3_long(launch_worst_long,arrival_worst_long);

% Finding wet_mass
total_dv_short_best = best_delta_v_short + total_delta_v_science_phase;
total_dv_long_best = best_delta_v_long + total_delta_v_science_phase;

m_wet_short_best = initial_masses(1,1)*exp((total_dv_short_best*1000)/
(Isp*g)); % kg
m_wet_long_best = initial_masses(1,1)*exp((total_dv_long_best*1000)/(Isp*g));
% kg
```

```
total_dv_short_worst = worst_delta_v_short + total_delta_v_science_phase;
total_dv_long_worst = worst_delta_v_long + total_delta_v_science_phase;

m_wet_short_worst = initial_masses(1,1)*exp((total_dv_short_worst*1000)/
(Isp*g)); % kg
m_wet_long_worst = initial_masses(1,1)*exp((total_dv_long_worst*1000)/
(Isp*g)); % kg

% Rockets
short_way_rocket_best = "Vulcan (VC2)";
long_way_rocket_best = "Falcon 9 (Full Thrust, ASDS)";

% Launch Dates
ideal_launch_short = launch_dates(launch_best_short);
ideal_arrival_short = arrival_dates(arrival_best_short);
TOF_short_best = days(ideal_arrival_short - ideal_launch_short); % days

worst_launch_short = launch_dates(launch_worst_short);
worst_arrival_short = arrival_dates(arrival_worst_short);
TOF_short_worst = days(worst_arrival_short - worst_launch_short); % days

ideal_launch_long = launch_dates(launch_best_long);
ideal_arrival_long = arrival_dates(arrival_best_long);
TOF_long_best = days(ideal_arrival_long - ideal_launch_long); % days

worst_launch_long = launch_dates(launch_worst_long);
worst_arrival_long = arrival_dates(arrival_worst_long);
TOF_long_worst = days(worst_arrival_long - worst_launch_long); % days

% Launch Windows
launch_window_start_short = ideal_launch_short - days(15);
launch_window_end_short = ideal_launch_short + days(15);
launch_window_width_short = days(launch_window_end_short -
launch_window_start_short);

launch_window_start_long = ideal_launch_long - days(16);
launch_window_end_long = ideal_launch_long + days(16);
launch_window_width_long = days(launch_window_end_long -
launch_window_start_long);

% Short - way trajectory
[E_a_s,E_e_s,E_i_s,E_Om_s,E_w_s,E_M_s] =
EarthOE(launch_dates_j(launch_best_short)); % Earth
[H_a_s,H_e_s,H_i_s,H_Om_s,H_w_s,H_M_s] =
AsteroidOE(arrival_dates_j(arrival_best_short)); % Halion

E_nu_s = Mean2True(E_M_s,E_e_s); % Earth
H_nu_s = Mean2True(H_M_s,H_e_s); % Halion

[r_vec_Es, v_vec_Es] =
Orbital2Cartesian(E_i_s,E_nu_s,E_w_s,E_Om_s,E_e_s,E_a_s,mu_sun); % Earth
[r_vec_Hs, v_vec_Hs] =
Orbital2Cartesian(H_i_s,H_nu_s,H_w_s,H_Om_s,H_e_s,H_a_s,mu_sun); % Halion
Asteroid
```

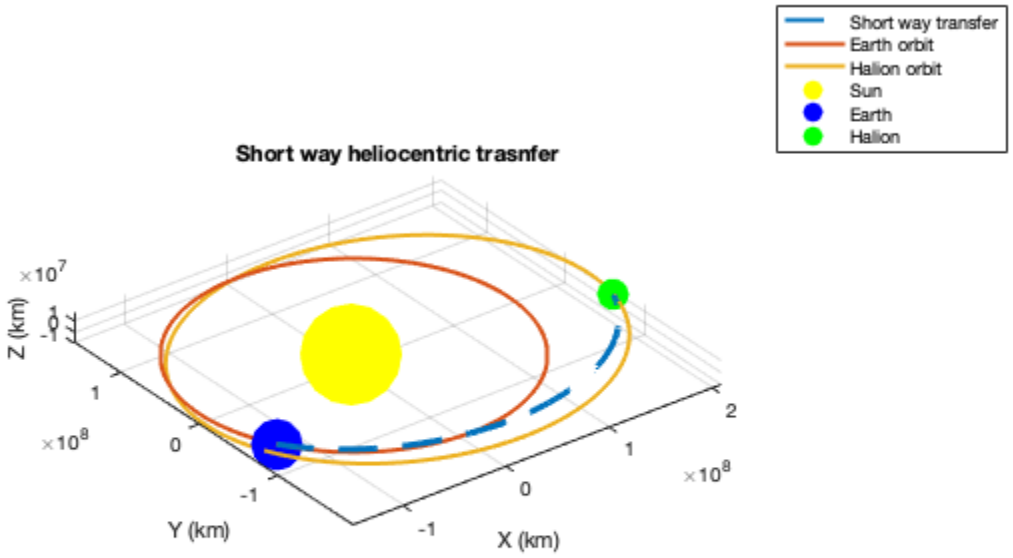
```
[v0_sc_vec_s,v_arr_vec_s,~] =
LambertsSolver(r_vec_Es,r_vec_Hs,TOF_short_best,1,mu_sun);

tspan_short = [0,TOF_short_best*24*3600];
y0_short = [r_vec_Es,v0_sc_vec_s];
[t_short,orbit_short] = ode45(@(t, y_vec) orbit(y_vec, mu_sun), tspan_short,
y0_short,options);

% Long - way trajectory
[E_a_l,E_e_l,E_i_l,E_Om_l,E_w_l,E_M_l] =
EarthOE(launch_dates_j(launch_best_long)); % Earth
[H_a_l,H_e_l,H_i_l,H_Om_l,H_w_l,H_M_l] =
AsteroidOE(arrival_dates_j(arrival_best_long)); % Halion

E_nu_l = Mean2True(E_M_l,E_e_l); % Earth
H_nu_l = Mean2True(H_M_l,H_e_l); % Halion

[r_vec_El, v_vec_El] =
Orbital2Cartesian(E_i_l,E_nu_l,E_w_l,E_Om_l,E_e_l,E_a_l,mu_sun); % Earth
[r_vec_Hl, v_vec_Hl] =
Orbital2Cartesian(H_i_l,H_nu_l,H_w_l,H_Om_l,H_e_l,H_a_l,mu_sun); % Halion
Asteroid

[v0_sc_vec_l,v_arr_vec_l,~] =
LambertsSolver(r_vec_El,r_vec_Hl,TOF_long_best,-1,mu_sun);

tspan_long = [0,TOF_long_best*24*3600];
y0_long = [r_vec_El,v0_sc_vec_l];
[t_long,orbit_long] = ode45(@(t, y_vec) orbit(y_vec, mu_sun), tspan_long,
y0_long,options);

% Earth and Halion orbits
jd_start = juliandate(datetime(2030,1,1));
jd_end = juliandate(datetime(2040,1,1));
jd_range = jd_start : 1 : jd_end;
TOF_EandH_orbits = days(datetime(2040,1,1) - datetime(2030,1,1));

[E_a,E_e,E_i,E_Om,E_w,E_M] = EarthOE(jd_range); % Earth
[H_a,H_e,H_i,H_Om,H_w,H_M] = AsteroidOE(jd_range); % Halion

E_nu = zeros(length(E_M),1);
H_nu = zeros(length(H_M),1);
for it = 1:length(E_M)
    E_nu(it) = Mean2True(E_M(it),E_e(it)); % Earth
    H_nu(it) = Mean2True(H_M(it),H_e(it)); % Halion
end

[r_vec_E, v_vec_E] =
Orbital2Cartesian(E_i(1),E_nu(1),E_w(1),E_Om(1),E_e(1),E_a(1),mu_sun); % Earth
[r_vec_H, v_vec_H] =
Orbital2Cartesian(H_i(1),H_nu(1),H_w(1),H_Om(1),H_e(1),H_a(1),mu_sun); %
Halion Asteroid
```

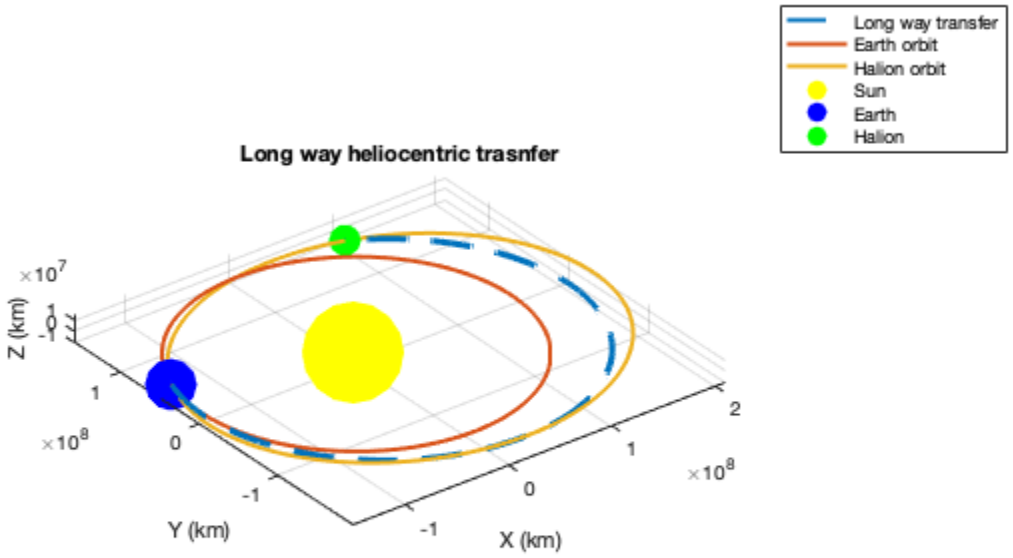
```
tspan_range = [0,TOF_EandH_orbits*24*3600];
y0_earth = [r_vec_E, v_vec_E];
y0_halion = [r_vec_H, v_vec_H];
[t_earth,orbit_earth] = ode45(@(t,y_vec) orbit(y_vec, mu_sun), tspan_range,
y0_earth, options);
[t_halion,orbit_halion] = ode45(@(t,y_vec) orbit(y_vec, mu_sun), tspan_range,
y0_halion, options);

% Vinf vector calculation
v_infs_E = v0_sc_vec_s - v_vec_Es;
v_infl_E = v0_sc_vec_l - v_vec_El;

% Delta V vector
dv_svec = v_arr_vec_s - v_vec_Hs;
dv_lvec = v_arr_vec_l - v_vec_Hl;

% Heliocentric Trajectory Table
Scenario = ["Short Transfer";"Long Transfer"];
Nominal_c3 = [best_c3_short;best_c3_long];
Worst_c3 = [worst_c3_short;worst_c3_long];
Nominal_Vinf_vec = [v_infs_E';v_infl_E'];
Nominal_wet_mass = [m_wet_short_best;m_wet_long_best];
Launch_vehicle = [short_way_rocket_best;long_way_rocket_best];
Ideal_launch_dates = [datestr(ideal_launch_short);datestr(ideal_launch_long)];
Ideal_launch_dates = string(Ideal_launch_dates);
Ideal_arrival_dates =
[datestr(ideal_arrival_short);datestr(ideal_arrival_long)];
Ideal_arrival_dates = string(Ideal_arrival_dates);
Ideal_TOF = [TOF_short_best;TOF_long_best];
Min_launch_dates =
[datestr(launch_window_start_short);datestr(launch_window_start_long)];
Min_launch_dates = string(Min_launch_dates);
Max_launch_dates =
[datestr(launch_window_end_short);datestr(launch_window_end_long)];
Max_launch_dates = string(Max_launch_dates);
Launch_window_width = [launch_window_width_short;launch_window_width_long];
dv_vec_all = [dv_svec';dv_lvec'];
dv_mag = [best_delta_v_short;best_delta_v_long];

Heliocentric_Transfer_table =
table(string(Scenario),Nominal_c3,Worst_c3,Nominal_Vinf_vec,Nominal_wet_mass,.
..

string(Launch_vehicle),string(Ideal_launch_dates),string(Ideal_arrival_dates),
Ideal_TOF,string(Min_launch_dates),...

string(Max_launch_dates),Launch_window_width,dv_vec_all,dv_mag,'VariableNames'
,{'Scenario','Nominal C3','Worst C3',...
    'Nominal Vinf Vector (km/s)','Nominal Wet Mass (kg)','Launch
Vehicle','Ideal Launch Date','Ideal Arrival Date','Ideal TOF (days)',...
    'Minimum Launch Date','Maximum Launch Date','Launch Window Width
(days)',' Rendevouz Delta V Vector (km/s)',' Rendevouz Delta V (km/s)'});
```

Trajectory Plot

```
% Short Way
sun_helio = [0;0;0]; % Sun at origin
earth_helio = r_vec_Es;
halion_helio = r_vec_Hs;

figure(7)
hold on
plot3(orbit_short(:,1),orbit_short(:,2),orbit_short(:,3),'--','LineWidth',2)
% short
plot3(orbit_earth(:,1),orbit_earth(:,2),orbit_earth(:,3),'LineWidth',2) %
Earth orbit
plot3(orbit_halion(:,1),orbit_halion(:,2),orbit_halion(:,3),'LineWidth',2) %
Halion orbit

plot3(sun_helio(1),sun_helio(2),sun_helio(3),'yo','MarkerSize',50,'MarkerFaceC
olor','y','DisplayName','Sun')
plot3(earth_helio(1),earth_helio(2),earth_helio(3),'bo','MarkerSize',25,'Marke
rFaceColor','b','DisplayName','Earth')
plot3(halion_helio(1),halion_helio(2),halion_helio(3),'go','MarkerSize',15,'Ma
rkerFaceColor','g','DisplayName','Halion')
grid on
view(3)
axis equal
legend('Short way transfer','Earth orbit','Halion
orbit','Sun','Earth','Halion')
title('Short way heliocentric trasnfer')
xlabel('X (km)')
ylabel('Y (km)')
zlabel('Z (km)')

% Long Way
earth_helio = r_vec_El;
halion_helio = r_vec_Hl;

figure(8)
hold on
plot3(orbit_long(:,1),orbit_long(:,2),orbit_long(:,3),'--','LineWidth',2) %
long
plot3(orbit_earth(:,1),orbit_earth(:,2),orbit_earth(:,3),'LineWidth',2) %
Earth orbit
plot3(orbit_halion(:,1),orbit_halion(:,2),orbit_halion(:,3),'LineWidth',2) %
Halion orbit

plot3(sun_helio(1),sun_helio(2),sun_helio(3),'yo','MarkerSize',50,'MarkerFaceC
olor','y','DisplayName','Sun')
plot3(earth_helio(1),earth_helio(2),earth_helio(3),'bo','MarkerSize',25,'Marke
rFaceColor','b','DisplayName','Earth')
plot3(halion_helio(1),halion_helio(2),halion_helio(3),'go','MarkerSize',15,'Ma
rkerFaceColor','g','DisplayName','Halion')
grid on
view(3)
```

```
axis equal
legend('Long way transfer','Earth orbit','Halion
orbit','Sun','Earth','Halion')
title('Long way heliocentric trasnfer')
xlabel('X (km)')
ylabel('Y (km)')
zlabel('Z (km)')
```





Printed Data

```
clc

fprintf('Mission: %s',mission_name)
fprintf('\n')
fprintf('Asteroid: %s',asteroid_name)
fprintf('\n')
fprintf('\n')
fprintf('Best C3 and Delta_V for short way are: %.4f km^2/s^2 & %.4f km/
s.',best_c3_short,best_delta_v_short)
fprintf('\n')
fprintf('Wet mass for the best short way is %.4f kg.',m_wet_short_best)
fprintf('\n')
fprintf('Ideal rocket for the short way trasnfer would be
%s.',short_way_rocket_best)
fprintf('\n')
fprintf('Ideal launch date (short way): %s',datestr(ideal_launch_short))
fprintf('\n')
fprintf('Ideal arrival date (short way): %s',datestr(ideal_arrival_short))
fprintf('\n')
fprintf('Time of flight = %.0f days.',TOF_short_best)
fprintf('\n')
fprintf('Worst launch date (short way): %s',datestr(worst_launch_short))
fprintf('\n')
```

```
fprintf('Worst arrival date (short way): %s',datestr(worst_arrival_short))
fprintf('\n')
fprintf('Time of flight = %.0f days.',TOF_short_worst)
fprintf('\n')
fprintf('Minimum launch date (short way):
%s',datestr(launch_window_start_short))
fprintf('\n')
fprintf('Maximum launch date (short way):
%s',datestr(launch_window_end_short))
fprintf('\n')
fprintf('Launch window width (short way): %.0f
days.',launch_window_width_short)
fprintf('\n')
fprintf('\n')
fprintf('Best C3 and Delta_V for long way are: %.4f km^2/s^2 & %.4f km/
s.',best_c3_long,best_delta_v_long)
fprintf('\n')
fprintf('Wet mass for the long way is %.4f kg.',m_wet_long_best)
fprintf('\n')
fprintf('Ideal rocket for the long way trasnfer would be
%s.',long_way_rocket_best)
fprintf('\n')
fprintf('Ideal launch date (long way): %s',datestr(ideal_launch_long))
fprintf('\n')
fprintf('Ideal arrival date (long way): %s',datestr(ideal_arrival_long))
fprintf('\n')
fprintf('Time of flight = %.0f days.',TOF_long_best)
fprintf('\n')
fprintf('Worst launch date (long way): %s',datestr(worst_launch_long))
fprintf('\n')
fprintf('Worst arrival date (long way): %s',datestr(worst_arrival_long))
fprintf('\n')
fprintf('Time of flight = %.0f days.',TOF_long_worst)
fprintf('\n')
fprintf('Minimum launch date (long way):
%s',datestr(launch_window_start_long))
fprintf('\n')
fprintf('Maximum launch date (long way): %s',datestr(launch_window_end_long))
fprintf('\n')
fprintf('Launch window width (long way): %.0f days.',launch_window_width_long)
fprintf('\n')
fprintf('\n')
fprintf('ORBIT TABLE')
fprintf('\n')
fprintf('\n')
disp(Orbit_table)
fprintf('\n')
fprintf('\n')
fprintf('MANEUVERS TABLE')
fprintf('\n')
fprintf('\n')
disp(Maneuver_table)
fprintf('\n')
fprintf('\n')
```

```
fprintf('HELIOCENTRIC TRANSFER TABLE')
fprintf('\n')
fprintf('\n')
disp(Heliocentric_Transfer_table)
```

```
Mission: SILENT HORIZON
Asteroid: Halion

Best C3 and Delta_V for short way are: 16.3113 km^2/s^2 & 3.7086 km/s.
Wet mass for the best short way is 3861.1793 kg.
Ideal rocket for the short way trasnfer would be Vulcan (VC2).
Ideal launch date (short way): 20-Apr-2033
Ideal arrival date (short way): 13-Oct-2033
Time of flight = 176 days.
Worst launch date (short way): 01-Jan-2035
Worst arrival date (short way): 26-Jan-2035
Time of flight = 25 days.
Minimum launch date (short way): 05-Apr-2033
Maximum launch date (short way): 05-May-2033
Launch window width (short way): 30 days.

Best C3 and Delta_V for long way are: 3.9859 km^2/s^2 & 2.3759 km/s.
Wet mass for the long way is 2242.4983 kg.
Ideal rocket for the long way trasnfer would be Falcon 9 (Full Thrust, ASDS).
Ideal launch date (long way): 02-Mar-2033
Ideal arrival date (long way): 11-Feb-2034
Time of flight = 346 days.
Worst launch date (long way): 01-Jan-2032
Worst arrival date (long way): 14-Feb-2032
Time of flight = 44 days.
Minimum launch date (long way): 14-Feb-2033
Maximum launch date (long way): 18-Mar-2033
Launch window width (long way): 32 days.

ORBIT TABLE

            Orbit              SMA (km)    eccentricity    inlcination (deg)
AOP (deg)    RAAN (deg)    True Anomaly (deg)
    ____________________    ________    _____________    _________________
_________    __________    __________________

    "Terminator Orbit 1"       1.3                0
90                0           270                0
    "Terminator Orbit 2"      0.35                0
90                0           270                0
    "Recon pos 60 deg"         0.34         0.029412
60                0           270              180
    "Recon pos 30 deg"         0.34         0.029412
30                0           270              180
    "Recon 0 deg"              0.34         0.029412
0                 0          270               180
    "Recon neg 30 deg"         0.34         0.029412
-30               0           270               180
    "Recon neg 60 deg"         0.34         0.029412
```

-60 0 270 180

MANEUVERS TABLE

Science Phase	Time (days)	Position (km)	Delta V vector (km/s)	Delta V (km/s)	Mass after Delta V (kg)
"Terminator Orbit 1 to trasnfer"	15.244	0 1.3 0	0 0 0	2.1623e-05	2.1623e-05 850.56
"Transfer to Terminator Orbit 2"	15.629	0 -0.35 0	0 0 -3.0513e-05	3.0513e-05	850.55
"Terminator Orbit 2 to pos 60 deg"	18.824	6.4294e-17 0.35 4.2863e-17	-5.8876e-05 1.1035e-20	1.7547e-05	6.1435e-05 850.53
"Pos 60 deg to Terminator Orbit 2"	18.925	-6.4294e-17 -0.35 0	-6.2444e-05 -1.1894e-16 1.1366e-05	6.347e-05	850.51
"Terminator Orbit 2 to pos 30 deg"	19.032	6.4294e-17 0.35 4.2863e-17	-0.00010198 1.8953e-20	6.0647e-05	0.00011865 850.47
"Pos 30 deg to Terminator Orbit 2"	19.134	-6.4294e-17 -0.35 0	-0.00010816 -1.1834e-16 5.7079e-05	0.00012229	850.42
"Terminator Orbit 2 to 0 deg"	19.24	6.4294e-17 0.35 4.2863e-17	-0.00011775 2.1851e-20	0.00011952	0.00016778 850.37
"0 deg to Terminator Orbit 2"	19.342	-6.4294e-17 -0.35 0	-0.00012489 -1.1971e-16 0.00011952	0.00017287	850.31
"Terminator Orbit 2 to neg 30 deg"	19.449	6.4294e-17 0.35 4.2863e-17	-0.00010198 1.8953e-20	0.0001784	0.00020549 850.23
"Neg 30 deg to Terminator Orbit 2"	19.551	-6.4294e-17 -0.35 0	-0.00010816 -1.1834e-16 0.00018197	0.00021168	850.16
"Terminator Orbit 2 to neg 60 deg"	19.657	6.4294e-17 0.35 4.2863e-17	-5.8876e-05 1.1035e-20	0.0002215	0.00022919 850.08
"Neg 60 deg to Terminator Orbit 2"	19.759	-6.4294e-17 -0.35 0	-6.2444e-05 -1.1894e-16 0.00022768	0.00023609	850

HELIOCENTRIC TRANSFER TABLE

```
        Scenario         Nominal C3    Worst C3        Nominal Vinf Vector
(km/s)         Nominal Wet Mass (kg)            Launch Vehicle
Ideal Launch Date    Ideal Arrival Date    Ideal TOF (days)    Minimum Launch
Date    Maximum Launch Date    Launch Window Width (days)     Rendevouz Delta
V Vector (km/s)     Rendevouz Delta V (km/s)
    _________________    __________    _________
_____________________________________    ______________________
_____________________________________    ___________________    ___________________
_________________    ____________________    ____________________
_____________________________    ____________________________________
_________________________

    "Short Transfer"      16.311        28074     -0.071158
-2.9173       2.7921           3861.2            "Vulcan
(VC2)"                       "20-Apr-2033"        "13-Oct-2033"
176              "05-Apr-2033"          "05-May-2033"
30                 3.5576    -0.92206    -0.49745
3.7086
    "Long Transfer"       3.9859       9656.5      -0.70551
-1.8571      0.19819           2242.5            "Falcon 9 (Full Thrust,
ASDS)"      "02-Mar-2033"        "11-Feb-2034"             346
"14-Feb-2033"          "18-Mar-2033"                    32
0.95352       1.145      1.8507              2.3759
```

Functions Used in the code

```
function [a,e,i,RAAN,w,M] = AsteroidOE(jd)

    % Input, date as a Julian Date
    % Outputs in Km or Degrees

    t=(jd-2451545.0);

    AU=149597870;
    muSun=1.32712428e11;

    a=1.2*AU*t.^0;
    e=0.2*t.^0;
    i=4*t.^0;
    RAAN=230*t.^0;
    w=278*t.^0;

    ts=t*24*60*60;
    n=sqrt(muSun/a(1)^3);

    M=wrapTo360(180/pi*n.*ts-52);

end

function [a,e,i,Om,w,M] = EarthOE(jd)
% Given a julian date, calculate the orbital elements of Earth
```

```
% Mark Moretto
% MAE467 F2024
% From Vallado, 4th edition

% OE in km and degrees

t=(jd-2451545.0)/36525;

AU=149597870;

a=1.000001018*AU*t.^0;

e=0.01670862-0.000042037*t-0.0000001236*t.^2+0.00000000004*t.^3;

i=0+0.0130546*t-0.00000931*t.^2-0.000000034*t.^3;

Om=174.873174-0.2410908*t+0.00004067*t.^2-0.000001327*t.^3;

wb=102.937348+0.3225557*t+0.00015026*t.^2+0.000000478*t.^3;

lm=100.466449+35999.3728519*t-0.00000568*t.^2;

w=wb-Om;

M=wrapTo360(lm-wb);

end

function [a0,e0,i,omega,RAAN,nu] = Cartesian2Orbital(r0_vec,v0_vec,mu)

    % Magnitude
    r0 = norm(r0_vec); %km
    v0 = norm(v0_vec); %km

    % Specific Angular Momentum
    h0_vec = cross(r0_vec,v0_vec);
    h0 = norm(h0_vec);

    % Node vector
    z_vec = [0;0;1];
    z = norm(z_vec);
    n_vec = cross(z_vec,h0_vec);
    n = norm(n_vec);
    n_unit = n_vec/n;

    if isnan(n_unit(1))
       n_unit = [0,0,0];
    end

    % Eccentricity vector
    e0_vec = (cross(v0_vec,h0_vec))/mu - r0_vec/r0;
    e0 = norm(e0_vec);
    e0_unit = e0_vec/e0;
```

```
    % Specific Energy
    E0 = v0^2/2 - mu/r0;

    % Semi - latus rectum
    p0 = h0^2/mu; %km

    % Semi-major axis
    a0 = p0/(1 - e0^2);

    % Inclination
    cos_i = dot(h0_vec,z_vec)/(h0*z);
    i = acosd(cos_i);

    % Argument of Pericenter
    cos_omega = dot(e0_unit,n_unit);
    omega = acosd(cos_omega);

    if e0_vec(3) < 0
        omega = 360 - omega;
    end

    % Right ascension of ascending node
    cos_RAAN = n_unit(1);
    RAAN = acosd(cos_RAAN);

    if n_vec(2) < 0
        RAAN = 360 - RAAN;
    end

    % True Anomaly
    cos_nu = dot(r0_vec,e0_vec)/(r0*e0);
    nu = acosd(cos_nu);

    if dot(r0_vec,v0_vec) < 0
        nu = 360 - nu;
    end
end

function [r1_vec,v1_vec] = Orbital2Cartesian(i,nu,omega,RAAN,e0,a0,mu)
    % Degrees to radians
    i_rad = deg2rad(i);
    nu_rad = deg2rad(nu);
    omega_rad = deg2rad(omega);
    RAAN_rad = deg2rad(RAAN);

    % Position Perifocal frame
    rpf_vec = [((a0*(1-e0^2))/(1+e0*cos(nu_rad)))*cos(nu_rad);...
        ((a0*(1-e0^2))/(1+e0*cos(nu_rad)))*sin(nu_rad);0];

    % Velocity Preifocal Frame
    vpf_vec = mu/sqrt(mu*(a0*(1-e0^2))) * [-sin(nu_rad);e0+cos(nu_rad);0];

    % Perifocal to General Inertial Frame forming IP
    IP = zeros(3,3);
```

```
    IP(1,1) = cos(RAAN_rad)*cos(omega_rad) -
sin(RAAN_rad)*sin(omega_rad)*cos(i_rad);
    IP(1,2) = -cos(RAAN_rad)*sin(omega_rad) -
sin(RAAN_rad)*cos(i_rad)*cos(omega_rad);
    IP(1,3) = sin(RAAN_rad)*sin(i_rad);
    IP(2,1) = sin(RAAN_rad)*cos(omega_rad) +
cos(RAAN_rad)*sin(omega_rad)*cos(i_rad);
    IP(2,2) = -sin(RAAN_rad)*sin(omega_rad) +
cos(RAAN_rad)*cos(omega_rad)*cos(i_rad);
    IP(2,3) = -cos(RAAN_rad)*sin(i_rad);
    IP(3,1) = sin(i_rad)*sin(omega_rad);
    IP(3,2) = sin(i_rad)*cos(omega_rad);
    IP(3,3) = cos(i_rad);

    % Back to original position and velocity
    r1_vec = IP*rpf_vec;
    v1_vec = IP*vpf_vec;

end

function [v1_vec,v2_vec, c3] = LambertsSolver(r1_vec,r2_vec,tof,tm,mu)

    r1_mag = norm(r1_vec);
    r2_mag = norm(r2_vec);

    tof = tof*24*3600; % In seconds

    % Calculations
    cos_delta_nu = dot(r1_vec,r2_vec)/(r1_mag*r2_mag);
    A = tm*sqrt(r1_mag*r2_mag*(1 + cos_delta_nu));

    if A == 0
        fprintf('Error')
    end

    % Initializing
    sai_n = 0;
    c2 = 1/2;
    c3 = 1/6;

    sai_up = 4*pi^2;
    sai_low = -4*pi;

    tol = 1e-6;
    delta_tn = 0;
    iteration = 0;

    % While loop
    while (abs(delta_tn - tof) > tol) && (iteration < 1000)

        yn = r1_mag + r2_mag + A * (sai_n*c3 - 1)/sqrt(c2);

        % IF statement
        if A > 0 && yn < 0
```

```
            sai_low = sai_n;
        end

        xn = sqrt(yn/c2);
        delta_tn = (xn^3 * c3 + A*sqrt(yn))/sqrt(mu);

        if delta_tn <= tof
            sai_low = sai_n;
        else
            sai_up = sai_n;
        end

        sai_n1 = (sai_low + sai_up)/2;
        sai_n = sai_n1;

        if sai_n > tol
            c2 = (1 - cos(sqrt(sai_n)))/sai_n;
            c3 = (sqrt(sai_n) - sin(sqrt(sai_n)))/sqrt(sai_n^3);

        elseif sai_n < -tol
            c2 = (1 - cosh(sqrt(-sai_n)))/sai_n;
            c3 = (sinh(sqrt(-sai_n)) - sqrt(-sai_n))/sqrt((-sai_n)^3);

        else
            c2 = 1/2;
            c3 = 1/6;

        end

        iteration = iteration + 1;
    end

    f = 1 - (yn/r1_mag);
    gdot = 1 - (yn/r2_mag);
    g = A*sqrt(yn/mu);

    v1_vec = (r2_vec - f*r1_vec)/g;
    v2_vec = (gdot*r2_vec - r1_vec)/g;

end

function [nu] = Mean2True(M,e)

    M = deg2rad(M);

    if M>pi
        E0 = M - e/2;
    elseif M< pi
        E0 = M + e/2;
    else
        E0 = M;
    end

    tol = 1e-8; % Radians
```

```
    iterations = 0;
    E_values=E0;

    while true
        E1 = E0 - (E0 - e*sin(E0) - M)/(1 - e*cos(E0));
        iterations = iterations + 1;
        E_values = [E_values;E1];
        if abs(E1 - E0) > tol
           E0 = E1;
        elseif abs(E1 - E0) < tol
            break;
        end
    end

    E = E1; % radians
    E = rad2deg(E); % Degrees

    nu = 2*(atan(sqrt((1+e)/(1-e))*tan(E1/2))); % radians
    nu = rad2deg(nu); % Degrees

end

function ydot = orbit(y,mu)

    % Extracts radius and velocity initially
    r_vec = y(1:3);
    r = norm(r_vec);

    v_vec = y(4:6);

    % Calculating accelration due to gravity
    a = -mu*r_vec/r^3;

    ydot = [v_vec;a];

end

function ydot = orbit_J2(y,mu,grad_val_func)

    % Extracts radius and velocity initially
    r_vec = y(1:3);
    r = norm(r_vec);

    v_vec = y(4:6);

    % Calculating accelration due to gravity
    a_g = -mu*r_vec/r^3;

    % Perturbation J2
    a_j2 = grad_val_func(r_vec(1),r_vec(2),r_vec(3));

    a = a_g + a_j2;

    ydot = [v_vec;a];
```

```
end

function ydot = Safety_orbit_J2(y_vec,mu,J2,R)

    % Extracts radius and velocity initially
    r_vec = y_vec(1:3);
    r = norm(r_vec);

    x = r_vec(1);
    y = r_vec(2);
    z = r_vec(3);

    v_vec = y_vec(4:6);

    % Calculating accelration due to gravity
    a_g = -mu*r_vec/r^3;

    a_j2x = (3*J2*R^2*mu*x*(x^2 + y^2 - 4*z^2))/(2*(x^2 + y^2 +z^2)^(7/2));
    a_j2y = (3*J2*R^2*mu*y*(x^2 + y^2 - 4*z^2))/(2*(x^2 + y^2 +z^2)^(7/2));
    a_j2z = (3*J2*R^2*mu*z*(3*x^2 + 3*y^2 - 2*z^2))/(2*(x^2 + y^2
+z^2)^(7/2));

    a_j2 = [a_j2x; a_j2y; a_j2z];

    a = a_g + a_j2;

    ydot = [v_vec;a];

end

Processing departure date 20
Processing departure date 40
Processing departure date 60
Processing departure date 80
Processing departure date 100
Processing departure date 120
Processing departure date 140
Processing departure date 160
Processing departure date 180
Processing departure date 200
Processing departure date 220
Processing departure date 240
Processing departure date 260
Processing departure date 280
Processing departure date 300
Processing departure date 320
Processing departure date 340
Processing departure date 360
Processing departure date 380
Processing departure date 400
Processing departure date 420
Processing departure date 440
Processing departure date 460
```

```
Processing departure date 480
Processing departure date 500
Processing departure date 520
Processing departure date 540
Processing departure date 560
Processing departure date 580
Processing departure date 600
Processing departure date 620
Processing departure date 640
Processing departure date 660
Processing departure date 680
Processing departure date 700
Processing departure date 720
Processing departure date 740
Processing departure date 760
Processing departure date 780
Processing departure date 800
Processing departure date 820
Processing departure date 840
Processing departure date 860
Processing departure date 880
Processing departure date 900
Processing departure date 920
Processing departure date 940
Processing departure date 960
Processing departure date 980
Processing departure date 1000
Processing departure date 1020
Processing departure date 1040
Processing departure date 1060
Processing departure date 1080
```

Published with MATLAB® R2024b