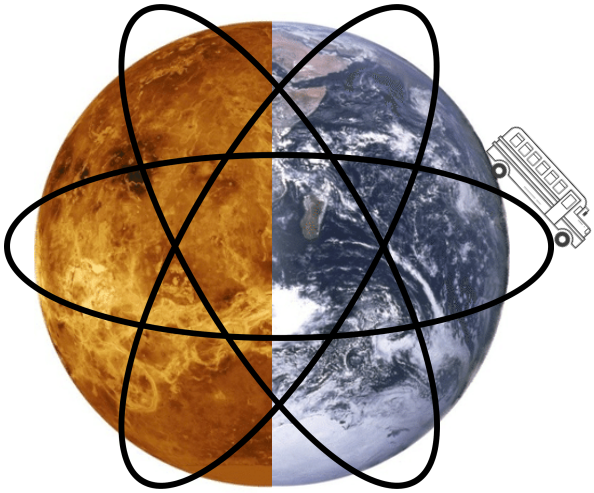
Iowa State University

Final Project: Aer E 351 Astrodynamics I
Fall 2023

Professor:
Dr. Simone Servadio

HBSB: Hell Bound School Bus

Group 11



Plantes image credit:Jamie Graff

Authors:

Surname	Name
Hassan	Martin
Elamin	Mayar
Steffensmeier	Karson

Submitted Date: 12.03.23

Teacher Comments:

Table of Contents

1. Introduction

In the vast expanse of interplanetary exploration, the Hell Bound School Bus (HBSB) project emerges as an ambitious testament to human ingenuity. As we set our sights beyond the familiar confines of our home planet, HBSB envisions a transformative interplanetary journey, casting the traditional space mission in the role of a "school bus." This imaginative approach not only aims to take us on a physical journey but also encourages us to start a mental exploration, diving into the concepts of efficiency and sustainable space travel.

Earth has a rich variety of ecosystems filled with life, water, and an atmosphere that supports us. On the flip side, Venus, our mysterious planetary sibling, has dense clouds of sulfuric acid, creating a harsh environment where surface temperatures reach blistering extremes.

It's crucial to grasp the basic ideas that guide these space journeys. First, orbital parameters are celestial measurements that describe how a spacecraft moves around a celestial body, creating the path for a specific mission, in this case, the HBSB's mission. These parameters, such as perigee radius, eccentricity, inclination, and true anomaly, offer a mathematical guide for the spacecraft's journey.

The HBSB journey resembles navigating a celestial highway between Earth and Venus, using a strategic orbital move known as the Hohmann transfer. This technique is known to be energy conserving which will enhance the journey's efficiency. Another essential element is impulse velocity which can be described as bursts of energy driving the spacecraft along its path. These bursts occur at specific moments.

The HBSB mission begins with a careful and detailed launch from Earth, ensuring the spacecraft is equipped with the required power for a safe and efficient departure. This phase involves two main burns: the first to escape Earth, and the second to enter the final orbit around Venus. Additionally, there will be other intermediate impulses throughout the trip.

2. Orbit i: Initial Orbit Characterization

2.1 Orbit i Characterization

The initial position and velocity vectors of the school bus are specified below:

$r_i = [-6221.9877999428 \quad -1974.43766107265 \quad 3693.02907379578]$

$v_i = [0.31648144597233 \quad -6.63125993907987 \quad -3.01212512711149]$

$R_i = 7500\ km$

$a_i = R_i = 7500\ km$

$e_i = 0$ (circular orbit)

$i_i = 40°$

$\omega_i = 100°$

$\Omega_i = 60°$

$\theta_i = 30°$

2.2 Discussion and Analysis

In its initial orbital phase, the school bus spacecraft operates in a circular orbit around Earth. This is called the parking orbit. As we see from the data above, the initial orbit is chosen to be circular (i.e. with eccentricity 0) for simplicity purposes. Since the orbit is circular, the semi-major axis is the radius of 7500 km of this orbit. Another important thing to note is that the RAAN (Ω) is set to be 0 degrees which makes the vernal equinox vector ($\widehat{I}$) and the line of nodes vector ($\widehat{N}$) align. Before any impulses are completed, the HBSB stays on the initial orbit for 1.3467 hours. After this flight, an impulse burn takes place that transfers the school bus to Orbit 1. This impulse is equal to 4.2423 km/s.

2.3 Orbit i Images

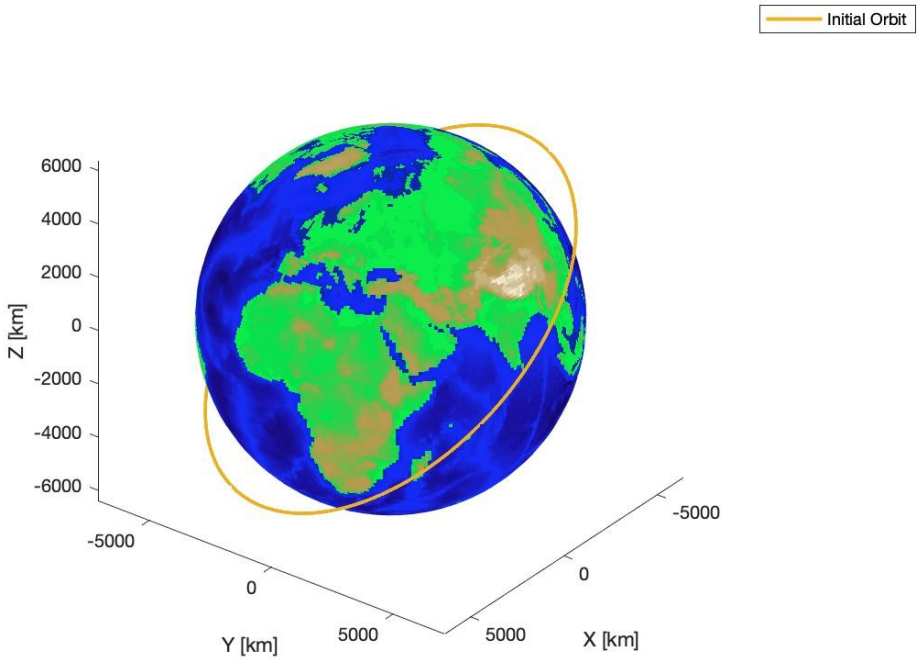


Figure 1. This figure shows the initial orbit around Earth

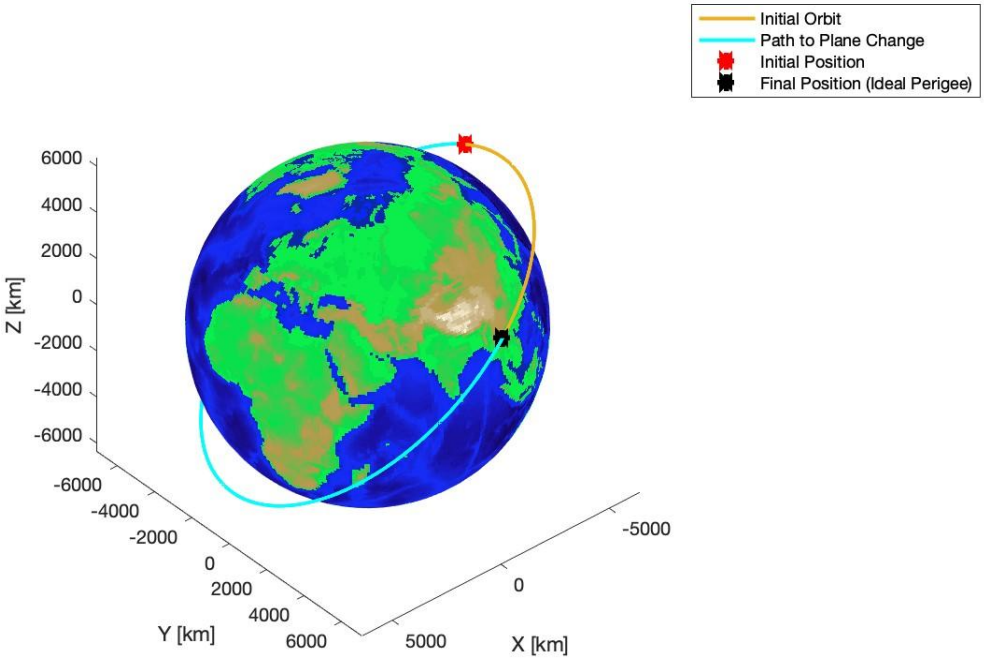


Figure 2. This figure shows the time of flight path on the initial orbit.

3. Orbit f: Final orbit characterization

3.1 Final Orbit Characterization

The final speed and velocity vectors for our school bus are as follows:

$$r_f = [-3924.95965183005 \quad -4625.84454210004 \quad -1881.34059032884]$$

$$v_f = [-1.86760920164055 \quad 4.20506244814649 \quad -6.44310470705568]$$

The orbital characterization of the final orbit in this mission are specified as:

$$a_f = 8202\ km$$

$$e_f = 0.2256$$

$$i_f = 120°$$

$$\omega_f = 200°$$

$$\Omega_f = 60°$$

$$\theta_f = 40°$$

3.2 Discussion and Analysis

This is the final orbit that our school bus arrives at around Venus. This final orbit has an inclination of 120 degrees as we can see in Figure 3.

3.3 Orbit f Image

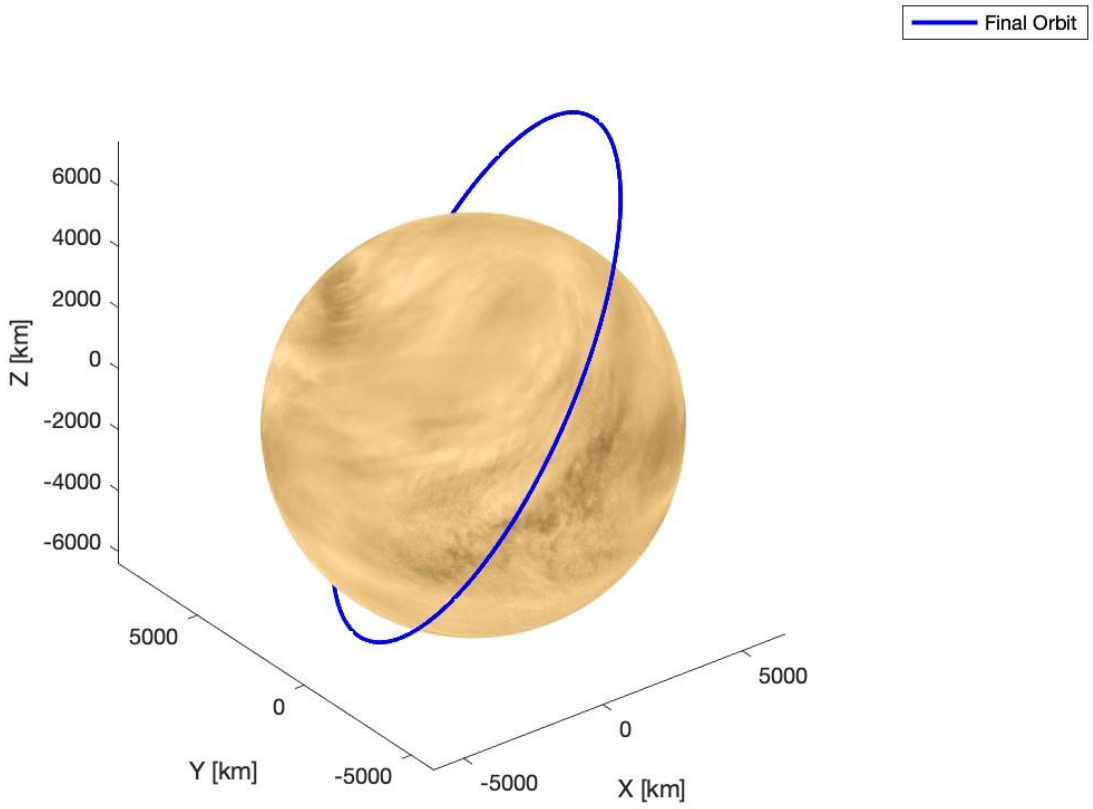


Figure 3. This figure shows the final orbit around Venus

4. Transfer trajectory definition and analysis

4.1 Orbit 1

4.1a Orbit 1 Characterization

This orbit has the same semi-major axis, eccentricity, and true anomaly as orbit i. However, the inclination changes to 23.4° and RAAN is set to zero, from the plane change impulse going from orbit i to orbit 1. The argument of periapsis (ω) is in this case is $\omega_1 = \beta + 90°$. This is so that our impulse to escape via orbit h1 is parallel with the velocity of earth, a “pro-grade” departure. Prograde means we are departing in the same direction the planet rotates.

4.1b Discussion and Analysis

This is the orbit we will escape from to the hyperbola. At this orbit, an important impulse maneuver transitions the spacecraft to a second circular orbit around the Earth. However, this time, the bus is aligned with the plane of the sun.

As the spacecraft progresses in orbit 1, an escape maneuver is executed with precision, pushing the spacecraft into Hyperbolic orbit 1. This escape velocity (V_∞), or excess velocity, is 2.5520 km/s. This happens at the perigee of the hyperbolic orbit.

4.1c Orbit 1 Image

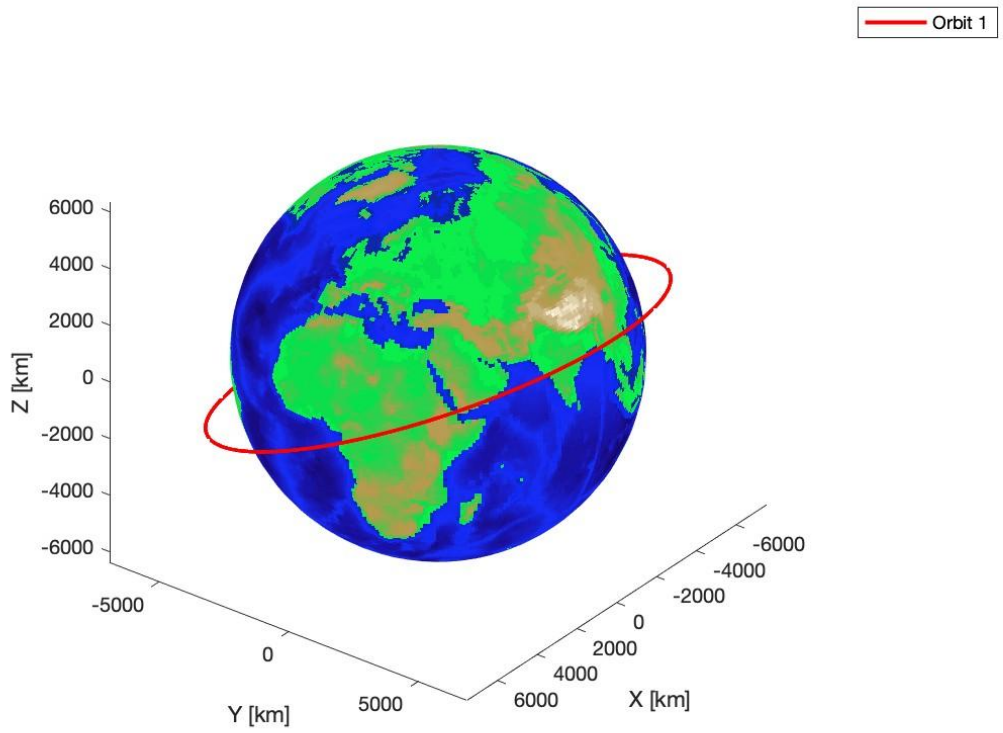


Figure 4. This figure shows the trajectory of Orbit 1.

4.2 Orbit h1: Hyperbolic Orbit 1, Departure Orbit From Earth

4.2a Orbit h1 Characterization

The characterization of this first hyperbolic trajectory, are summarized below:

$a_{h1} = -\ 6.1202\ \times 10^{4}\ km$

$e_{h1} = \ 1.225$

For this trajectory, the inclination, RAAN, argument of perigee, and true anomaly stay the same as orbit 1.

4.2b Discussion and Analysis

After applying an impulse of 3.3308 km/s we can start to travel on orbit h1, departing orbit 1 and Earth. The hyperbolic orbit 1 has a speed of 10.6210 km/s. This orbit is the beginning of the interplanetary Hohmann transfer to Venus stage. It is responsible for taking the bus to orbit 2, which is the stage when the bus starts orbiting the sun and travels to Venus. We move relatively constant until we reach the SOI of Venus. This orbit is of course equivalent to orbit 2 and orbit h2, but differing in SOI.

4.2c Orbit h1 Image

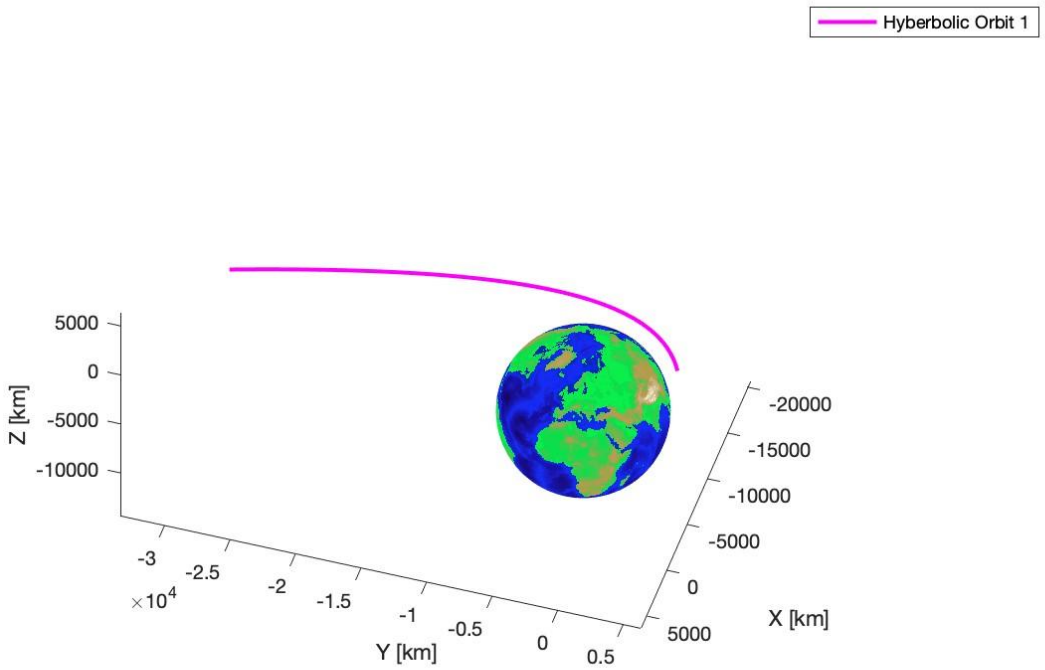


Figure 5. This figure shows the trajectory of orbit h1.

4.3 Orbit 2: Interplanetary Hohmann Transfer

4.3a Orbit 2 Characterization

The orbital parameters of orbit 2 are the following:

$a_2 = 1.30518 \times 10^8 \, km$

$e_2 = 0.1653$

$i_{sun} = 0°$

$\omega_2 = 200°$

$\Omega_2 = 60°$

$\theta_2 = 180°$

One thing to mention here is the sudden change from the inclination of 23.4°. This is because the earth's celestial orbital plane is inclined to the sun's ecliptic plane by that exact amount. So when we reference the sun's SOI, it becomes an orbit of inclination 0°.

4.3b Discussion and Analysis

We have now left the sphere of influence (SOI) of earth and entered the SOI of the Sun. Entering the interplanetary transfer orbit (orbit 2) referenced to the sun, the spacecraft follows a trajectory that positions it for a rendezvous with Venus. We leave the Earth, which is the apogee of orbit 2 and arrive at Venus, the perigee.

4.3c Orbit 2 Image

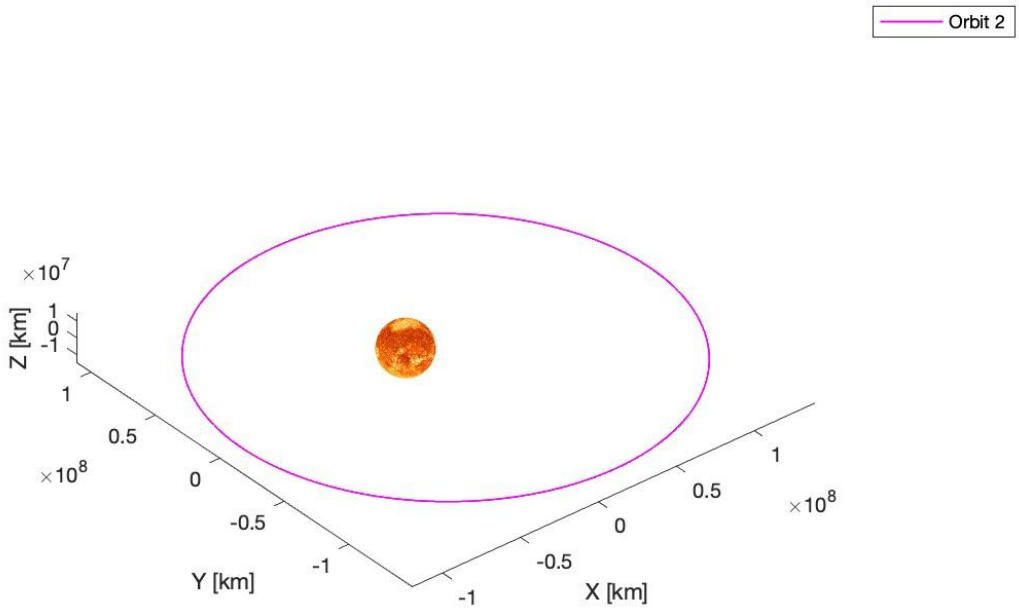


Figure 6. This figure shows the trajectory of orbit 2.

4.4 Orbit h2: Hyperbolic Orbit 2, SOI of Venus

4.4a Orbit h2 Characterization

It should be noted that both orbits h1 & h2 are essentially the same orbits, along with orbit 2. The only difference is the varying differences in spheres of influence. Orbit

h1,h2, and orbit2 are altered based on the Earths, Venus's and the sun's SOI respectively.

The details of this orbit are:

$a_{h2} = -4.219 \times 10^{4}\, km$

$e_{h2} = 1.1506$

$i_2 = 3.39°$

$\omega_2 = \beta_2 + 270°$

$\Omega_2 = 0°$

$\theta_2 = 0°$

4.4b Discussion and Analysis

When we enter the SOI of Venus, the hyperbolic orbit parameters change. The inclination is changed to the inclination Venus is to the Sun's ecliptic plane, it becomes 3.39º when entering Venus's SOI. This new orbit is when the spacecraft moves into a hyperbolic orbit around Venus (Hyperbolic orbit 2). In this case $\omega_2 = \beta_2 + 270°$, because like explained in the orbit 1 section, we have to have our impulses parallel with the velocity of the planet we are departing or arriving at. In this particular case, we are adding 270 degrees, because we are arriving retrograde.

4.4c Orbit h2 Image

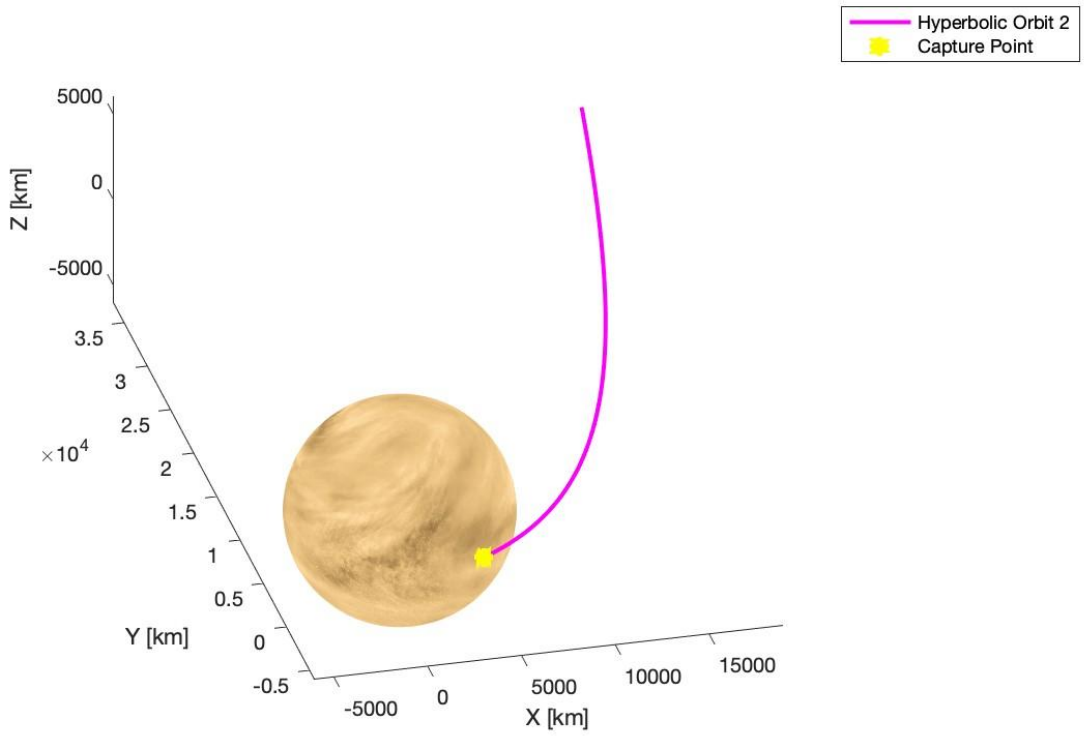


Figure 7. This figure shows the trajectory of orbit h2.

4.5 Orbit 3: Capture Orbit at Venus

4.5a Orbit 3 Characterization

First thing to note here is that this orbit is assumed to be circular so as explained before, we can set RAAN = 0 and allow us to calculate the position where orbit h2 connects to orbit 3.

The following parameters characterize the circular orbit around Venus:

$a_3 = 6352\ km$

$e_3 = 0\ (circular\ orbit)$

The inclination, RAAN, true anomaly, and argument of perigee all stay the same as orbit h2.

4.5b Discussion and Analysis

We have finally arrived at the SOI of Venus and are now orbiting in a circular orbit. The bus escapes the arrival hyperbolic trajectory and enters this orbit by slowing down. It is important to note that the bus arrives in a retrograde state.

4.5c Orbit 3 Image

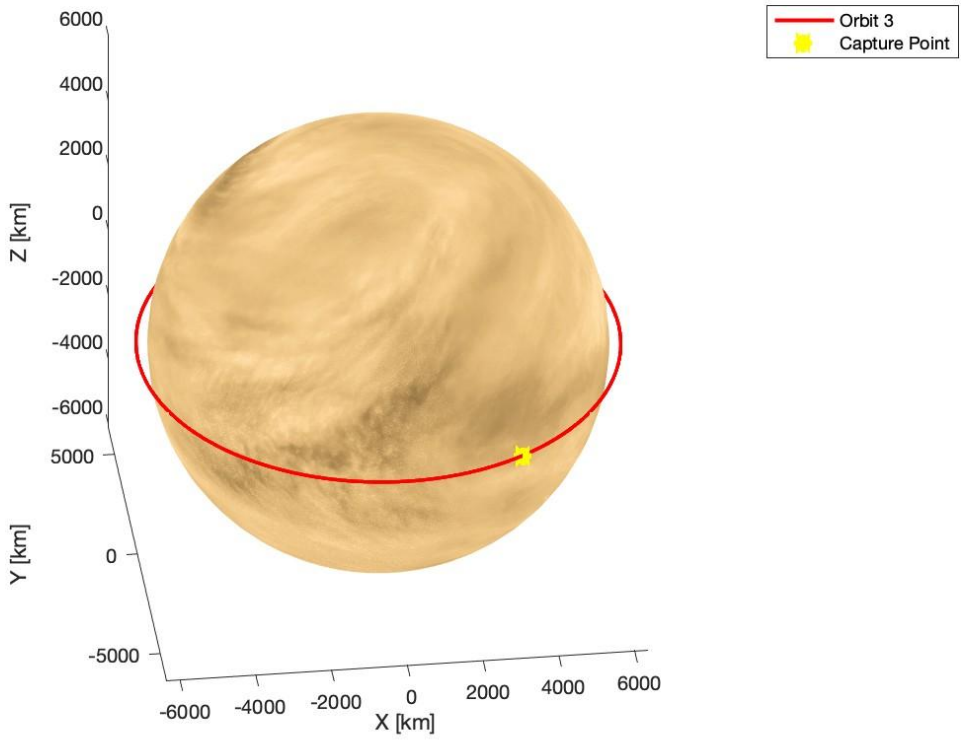


Figure 8. This figure shows the trajectory of orbit 3.

4.6 Orbit 4: Change Orbital Plane of Orbit 3 to Final Plane

4.6a Orbit 4 Characterization

This orbit is equivalent to orbit 3 in both size, dimension, and orientation. All that is different between them is that orbit 4 lies on the plane of our final orbit.
With: $i_f = 120°$ and $\Omega_f = 60°$.

4.6.b Discussion and Analysis

All that is done at this orbit is to get orbit 3 onto our final orbits plane. Once this is done, a simple size/dimension and reorientation can be applied to arrive at the final orbit.

4.6c Orbit 4 Image

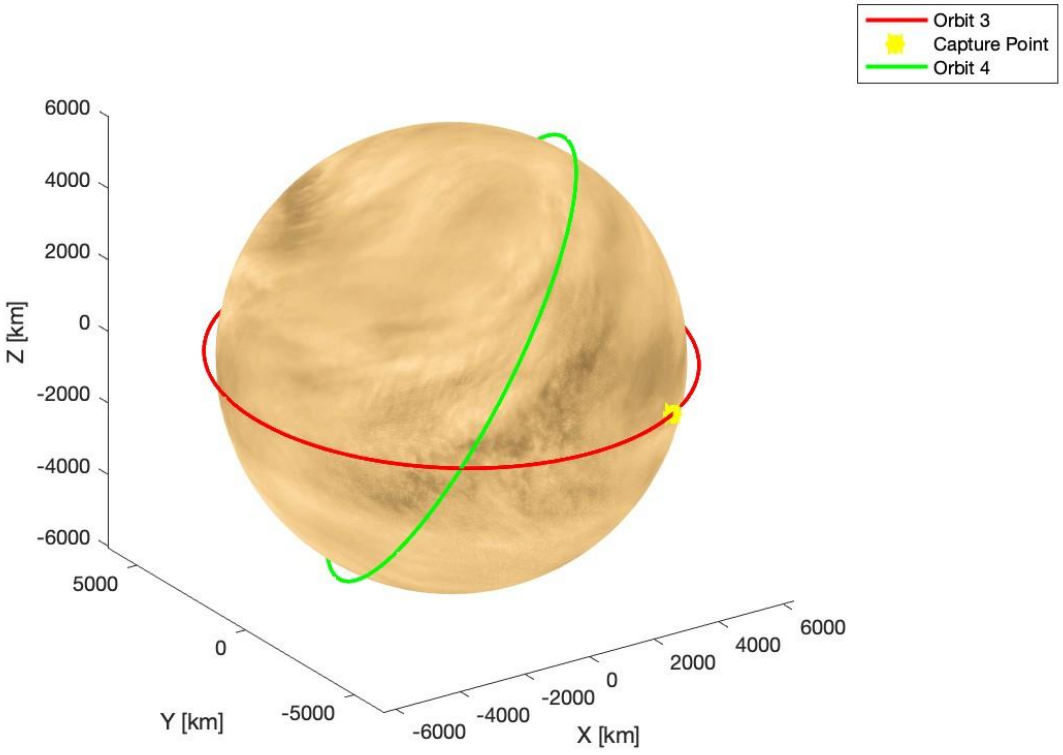


Figure 9. This figure shows orbit 4 around Venus.

4.7 All Orbits Combined

4.7a Earth's SOI

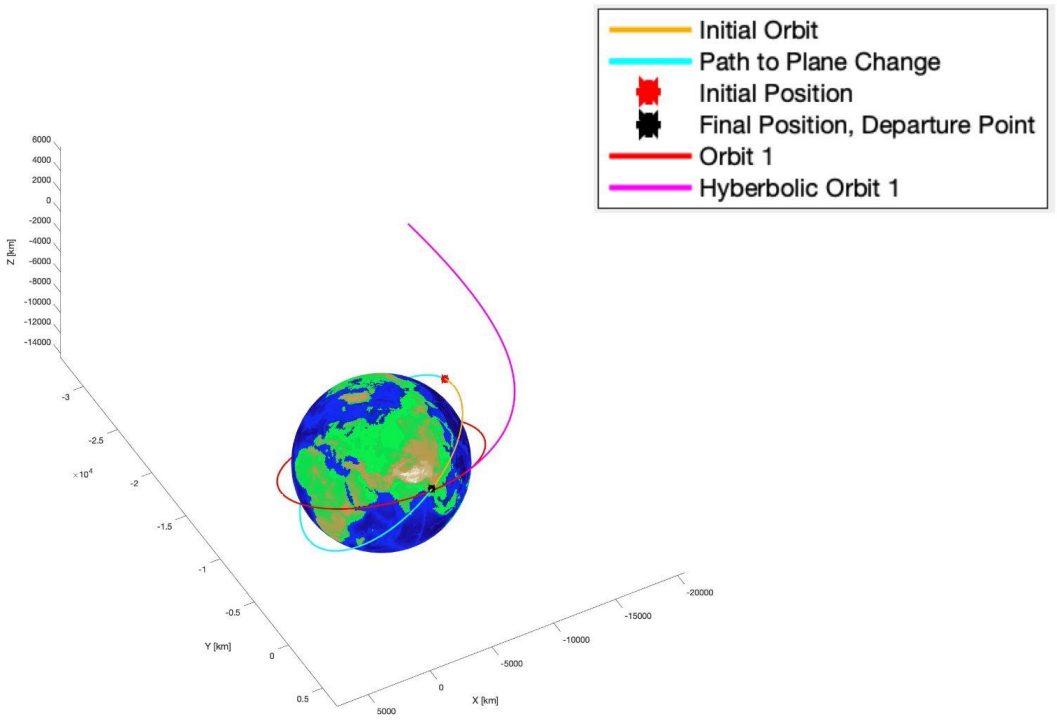


Figure 10. This figure illustrates the sphere of influence of Earth as well as the departure orbits around it.

4.7b Sun's SOI

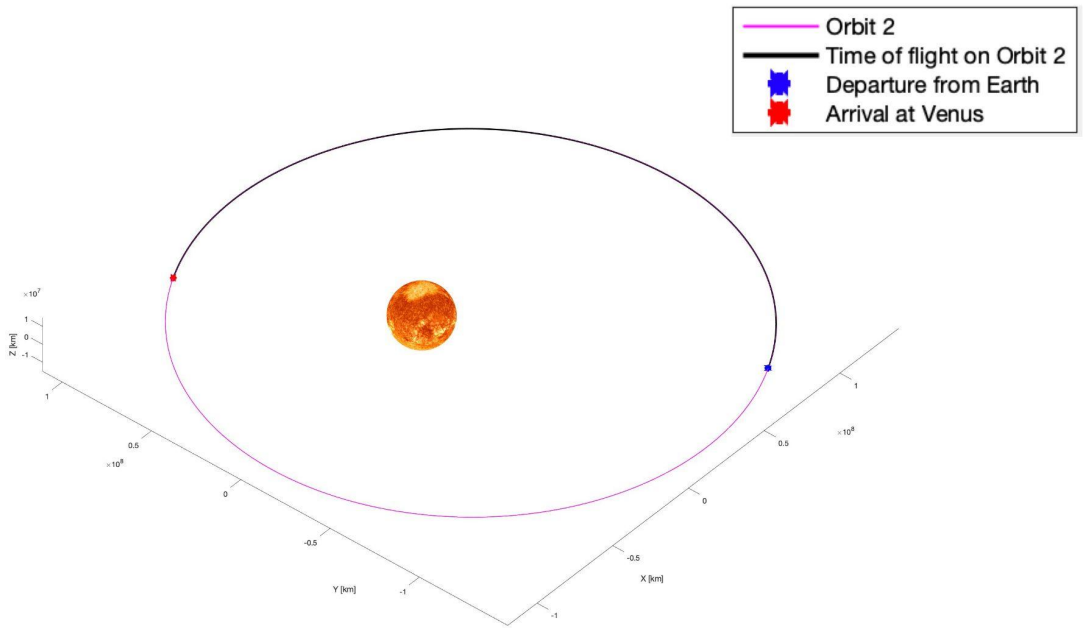


Figure 11. This figure shows the sphere of influence of the Sun where orbit 2 takes place.

4.7c Venus's SOI

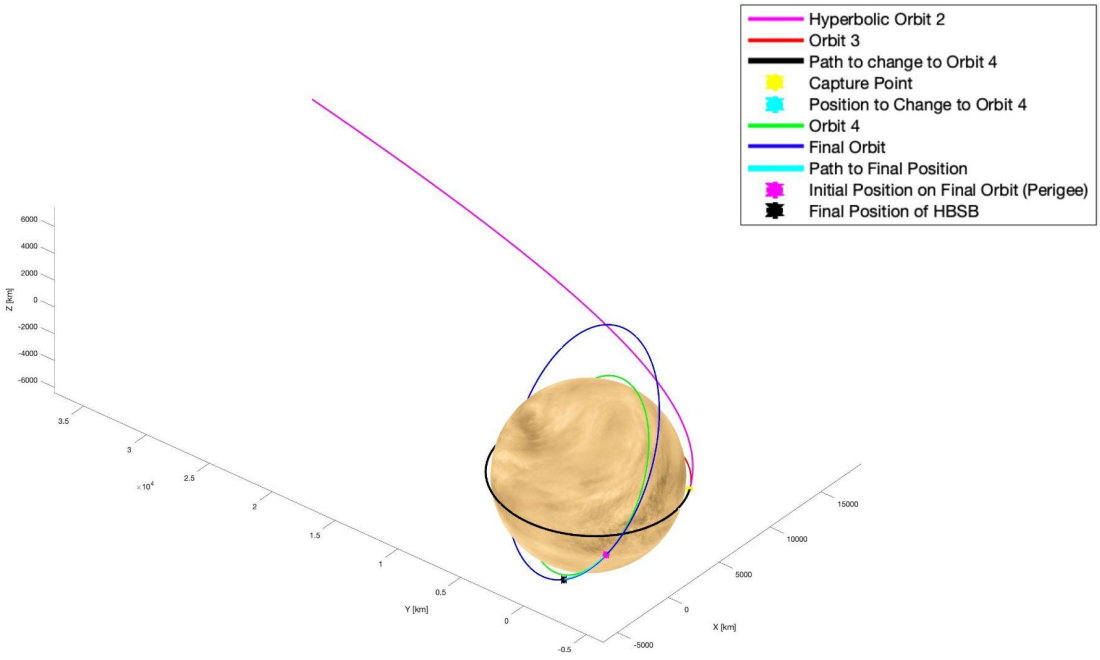


Figure 12. This figure illustrates the sphere of influence of Venus and the arrival orbits around it.

5. Conclusion

In conclusion, the success of the Hell Bound School Bus (HBSB) mission lies in the precisely calculated parameters and precise maneuvers done throughout the entire interplanetary journey. From the MATLAB code created for this mission, we see that the total velocity change (total impulse) of 23.18557 km/s reflects the propulsion strategy and fuel amount that need to be taken into consideration. This total impulse is the contributions from the first, third, first hyperbolic, second hyperbolic and the single-impulse stages, measuring 4.2423 km/s, 12.2768 km/s, 3.3308 km/s, 2.5701 and 0.7655 km/s, respectively. Additionally, we can conclude that the total time of flight (tof) for this mission is 148.935 days. This is from the varying durations for each orbital phase, with tof 1, tof 2, tof 3, and tof 4 measuring 1.3467 hours, 148.83 days, 1.039 hours, and 0.1603 hours. This journey aims for precision and thoughtful planning to ensure the Hell Bound School Bus not only reaches its destination but does so with maximum efficiency with the orbits prescribed by the mission. We could have opted for eccentric arrival and departure orbits, but it was decided that arriving to Venus quickly was of most importance. Thus, while some parts could have been done more efficiently, that would have opposed the mission's ultimate goal of ensuring the passengers on board were able to see Venus before running out of supplies and dying.

6. References

Graff, J. (2016, September). *A history of tectono-magmatism along the parga Chasma Rift System on Venus*. researchgate. https://www.researchgate.net/publication/308419050_A_History_of_Tectono-Magmatism_along_the_Parga_Chasma_Rift_System_on_Venus

Appendix: MATLAB Functions

Main Code:

```
clear, clc, close all
% Fall 2023 Semester
% Dr. Simone Servadio
% AerE 351 Final Project Code
% Group 11: Martin Hassan, Mayar Elamin, Karson Steffensmeier
mu_earth = 398600; %km^2/s^2
mu_venus = 324858.204; % km^3/s^2
r_venus = 1.0894*10^8; % Distance of venus from the sun in km
r_earth = 1.52096*10^8; % Distance of the earth from the sun in km
%Initial Orbit Characterization
Rp_i = 7500; %km
e_i = 0;
a_i = Rp_i; %km % circular orbit
incl_i = 40; %deg
argp_i = 100; %deg
raan_i = 60; %deg
theta_i = 30; % Initial Position of spacecraft
% We must define the final orbit we want at venus
%Final Orbit Characterization
Rp_f = 6352; %km
Ra_f = 10052; %km
a_f = 8202; %km
e_f = 0.2256;
incl_f = 120; %deg
argp_f = 200; %deg
raan_f = 60; %deg
theta_f = 40; %deg
odeoptions = odeset('RelTol',1e-13,'AbsTol',1e-13);
%% Plot Parking Orbit (Initial Orbit) around Earth
[ri,vi] = Kepl_2_Cart(a_i,e_i,argp_i,raan_i,incl_i,theta_i,mu_earth);
xi = [ri vi];
Ti = (2*pi*a_i^(3/2))/sqrt(mu_earth);
figure;
[~,sol] = ode45(@(t,x) two_body(x, mu_earth),[0 Ti], xi, odeoptions);
markerOne(1) = plot3(sol(:,1),sol(:,2),sol(:,3),'color','[0.9290 0.6940
0.1250]', 'lineWidth', 2);
hold on;
%% Plot the time of flight (tof) to get to ideal perigee
% First we must find the ideal "perigee" of this circular orbit to depart!
% To do this, we need Beta and the parameters of a hyperbolic orbit.
[Vinf1, del_Vph1, Beta1, a_h1, e_h1, Vph1] = hype_depart(r_earth, r_venus,
mu_earth, Rp_i, a_i, e_i);
% argp1 is the argument of periapsis that will allow us to depart with an
% impulse that is parallel to the infinite velocity and the velocity of the
% earth. We will set this as the true anomaly of the position of this
% "ideal perigee."
```

```
argp1 = Beta1+90; % explained above.
theta1 = 300; % Ignore this value, this is just used to place my marker
raan1 = 0; % Set RAAN to zero so that the Celestial "I axis" is  the line of
Nodes
incl1 = 23.4; % Inclination so that new orbit is on ecliptic plaane
Ei = True2Ecc(theta_i,e_i); %These are meaningless, since circular orbit
Mi = Ecc2Mean(Ei,e_i); %These are meaningless, since circular orbit
E1 = True2Ecc(theta1,e_i); %These are meaningless, since circular orbit
M1 = Ecc2Mean(E1,e_i); %These are meaningless, since circular orbit
tof1 = (M1-Mi)/sqrt(mu_earth/a_i^3);
[t,sol] = ode45(@(t,x) two_body(x, mu_earth),[0 tof1], xi, odeoptions);
markerOne(2) = plot3(sol(:,1),sol(:,2),sol(:,3),'c', 'lineWidth', 2); %hold on;
markerOne(3) = plot3(sol(1,1),sol(1,2),sol(1,3), '*r', 'lineWidth', 3,
'MarkerSize', 8);
markerOne(4) = plot3(sol(end,1),sol(end,2),sol(end,3), '*k', 'lineWidth', 3,
'MarkerSize', 8);
%% Plot Orbit 1,  the orbit we depart from
thetaOpt = 0; % True anomaly of the ideal perigee (at argp = beta + 90)
%Plane change to raan = 0 and inclination = 23.4
[~,theta2,delV1,~] =
PlaneChange(a_i,e_i,incl_i,incl1,raan_i,raan1,argp_i,mu_earth);
%Get position and velocity to plot
[r1,v1] = Kepl_2_Cart(a_i,e_i,argp1,raan1,incl1,thetaOpt,mu_earth);
%Plot the Orbit after Plane Change, Orbit 1
x1 = [r1 v1];
[~,sol] = ode45(@(t,x) two_body(x, mu_earth),[0 Ti], x1, odeoptions);
markerOne(5) = plot3(sol(:,1),sol(:,2),sol(:,3),'r','lineWidth',2);
%% Departure from Orbit 1, through hyperbolic Orbit 1
%Get position and velocity of departure hyperbolic orbit
[r_h1,v_h1] = Kepl_2_Cart(a_h1,e_h1,argp1,raan1,incl1,thetaOpt,mu_earth);
%Get unit vector of Vp when on the hyperbolic orbit
V_h1 = norm(v_h1);
v_h1_unit = v_h1/V_h1;
vph1 = Vph1*v_h1_unit;
%Get position of "ideal" perigee at omega = Beta + pi/2
[rp1,v_1] = Kepl_2_Cart(a_i,e_i,argp1,raan1,incl1,thetaOpt,mu_earth);
%Get unit vector at the ideal perigee
Rp1 = norm(rp1);
rp1_unit = rp1/Rp1;
r_esc1 = Rp_i*rp1_unit;
%x_h = [ri vph];
x_h1 = [r_esc1 vph1];
%x_h = [r_h1 vph];
% Plot Hyperbolic Orbit
[~,sol] = ode45(@(t,x) two_body(x, mu_earth),[0 7e3], x_h1, odeoptions);
markerOne(6) = plot3(sol(:,1),sol(:,2),sol(:,3), 'm', 'lineWidth', 2);
% Plot the Earth to make things nice
earth_sphere;
axis image % Set a nice axis for viewing
```

```
%Legend for Figure 1
legend1 = legend(markerOne,'Initial Orbit','Path to Plane Change','Initial
Position', ...
   'Final Position, Departure Point','Orbit 1','Hyberbolic Orbit 1');
hold off;
%% Interplanetary Hohmann Transfer to Venus (Orbit 2)
%Sphere of Influence of the Sun now, so hyperbolic orbit 1 becomes Orbit 2
mu_sun = 1.32712*10^11; %km^2/s^2
a_2 = (r_venus+r_earth)/2; %Semimajor axis of Orbit 2
e_2 = (r_earth - r_venus)/(r_earth + r_venus); % Eccentricity of orbit 2
%Set some orbit parameters of Orbit 2
incl_sun = 0; % Reset inclination to zero, we are orbiting around the sun's own
plane (ecliptic)
raan_sun = raan1; %Hyperbolic orbit 1 is the same as orbit 2, just a different
SOI.
argp_sun = argp1; %Hyperbolic orbit 1 is the same as orbit 2, just a different
SOI.
theta_T = 180; % Earth is the apogee of orbit 2
%Get orbit 2 characterization into cartesian
[r2,v2] = Kepl_2_Cart(a_2,e_2,argp_sun,raan_sun,incl_sun,theta_T,mu_sun);
x2 = [r2,v2];
%Figure 2
figure;
%background('Milky Way') % Choose a background to look cooler, comment if not
used
hold on;
sun_sphere; %Sphere of the Sun, just for visual effect, NOT TO SCALE!!!
axis image % Comment out if background is on!
view(3) %View 3D
%Period of Orbit 2
T2 = (2*pi*a_2^(3/2))/sqrt(mu_sun);
%Plot Orbit 2
[~,sol] = ode45(@(t,x) two_body(x, mu_sun),[0 T2], x2, odeoptions);
markerTwo(1) = plot3(sol(:,1),sol(:,2),sol(:,3), 'm', 'lineWidth', 1);
% Find the tof duration while on Orbit 2
tof2 = pi*sqrt(a_2^3/mu_sun);
% Plot tof2
[~,sol] = ode45(@(t,x) two_body(x, mu_sun),[0 tof2], x2, odeoptions);
hold on;
markerTwo(2) = plot3(sol(:,1),sol(:,2),sol(:,3), 'k', 'lineWidth', 2);
% ^ Comment if background is on! ^
%markerTwo(2) = plot3(sol(:,1),sol(:,2),sol(:,3), 'g', 'lineWidth', 2);
% ^ Uncomment if background is on! ^
% Markers to show arrival and departure points
markerTwo(3) = plot3(sol(1,1),sol(1,2),sol(1,3), '*b', 'lineWidth', 3,
'MarkerSize', 8);
markerTwo(4) = plot3(sol(end,1),sol(end,2),sol(end,3), '*r', 'lineWidth', 3,
'MarkerSize', 8);
%Legend for Figure 2
```

```
legend2 = legend(markerTwo,'Orbit 2','Time of flight on Orbit 2',...
   'Departure from Earth','Arrival at Venus');
hold off; % End figure 2
%% Venus Arrival Orbit, Hyperbolic Orbit 2
% Sphere of Influence of Venus now
% We must define the hyberbolic orbit at venus:
[Vinf2, del_Vph2, Beta2, DELTA, a_h2, e_h2, Vph2] = hype_arrival(r_earth,
r_venus, mu_venus, Rp_f, a_f, e_f);
% argp2 is the argument of periapsis that will allow us to arrive with an
% impulse that is parallel to the infinite velocity and the velocity of venus.
% We will set this as the true anomaly of the position of this
argp2 = Beta2+270; % explained above.
%theta2 = 300; % Ignore this value, this is just used to place my marker
raan2 = 0; % Set RAAN to zero so that the Celestial "I axis" is  the line of
Nodes
incl2 = 3.39; % Inclination so that new orbit is on ecliptic plane
%Get position and velocity of arrival hyperbolic orbit
[r_h2,v_h2] = Kepl_2_Cart(a_h2,e_h2,argp2,raan2,incl2,thetaOpt,mu_venus);
%Get unit vector of Vp when on the hyperbolic orbit
V_h2 = norm(v_h2);
v_h2_unit = v_h2/V_h2;
vph2 = Vph1*v_h2_unit;
%Get position of "ideal" perigee at omega = Beta + pi/2
[rp2,v_2] = Kepl_2_Cart(a_f,e_f,argp2,raan2,incl2,thetaOpt,mu_venus);
%Get unit vector at the ideal perigee
Rp2 = norm(rp2);
rp2_unit = rp2/Rp2;
r_esc2 = Rp_f*rp2_unit;
%x_h = [ri vph];
x_h2 = [r_esc2 vph2];
%x_h = [r_h1 vph];
% Figure 3
figure;
venus_sphere; % Plot Venus to make things nice
axis image; % Set a nice axis for viewing
view(3); %View 3D
hold on; %put everything on figure 3
%Plot Hyperbolic Orbit 2
[~,sol] = ode45(@(t,x) two_body(x, mu_venus),[0 7e3], x_h2, odeoptions);
markerThree(1) = plot3(sol(:,1),sol(:,2),sol(:,3),'m', 'lineWidth', 2);
%% Escape from Arrival Hyperboic Orbit (Hype Orbit 2) to Orbit 3
% Assume a circular arrival orbit
a_3 = Rp_f; %Circular orbit, means the perigee is the same as hyperbolic orbit
2
e_3 = 0; % Circular orbit, no eccentricity
%Get position and velocity to plot
[r3,v3] = Kepl_2_Cart(a_3,e_3,argp2,raan2,incl2,thetaOpt,mu_venus);
%We arrive retrograde, v3 must be set negative, so we propagate with ode45 in
the correct direction!
```

```
x3 = [r3,-v3];
%Calculate the Period of Orbit 3
T3 = (2*pi*a_3^(3/2))/sqrt(mu_venus);
% Plot the Arrival Orbit (Orbit 3), the escape from Hype Orbit 2
[~,sol] = ode45(@(t,x) two_body(x, mu_venus),[0 T3], x3, odeoptions);
markerThree(2) = plot3(sol(:,1),sol(:,2),sol(:,3), 'r', 'lineWidth', 2);
%% Plot tof on orbit 3 to get to the true anomaly to change to orbit 4
E3_1 = True2Ecc(0,e_3); %rad
M3_1 = Ecc2Mean(E3_1,e_3); %rad
%Find the true anomaly where orbits 3 & 4 meet, theta3.
[theta3,theta4,delV3,argp3] = PlaneChange(a_3, e_3,incl2, incl_f, raan2,
raan_f, argp2, mu_venus);
E3_2 = True2Ecc(360-theta3,e_3); %rad
M3_2 = Ecc2Mean(E3_2,e_3); %rad
%Find time of flight while on orbit 3
tof3 = (M3_2-M3_1)/sqrt(mu_venus/a_3^3);
% plot this time of flight onto orbit 3
[~,sol] = ode45(@(t,x) two_body(x, mu_venus),[0 tof3], x3, odeoptions);
markerThree(3) = plot3(sol(:,1),sol(:,2),sol(:,3),'k', 'lineWidth', 3);
%Markers for the start and end of tof3
markerThree(4) = plot3(sol(1,1),sol(1,2),sol(1,3), '*y', 'lineWidth', 3,
'MarkerSize', 8);
markerThree(5) = plot3(sol(end,1),sol(end,2),sol(end,3), '*c', 'lineWidth', 3,
'MarkerSize', 8);
%% We Will now change to orbit 4 at theta3, the true anomaly where we change
planes to  final orbit's plane
[r4,v4] = Kepl_2_Cart(a_3,e_3,argp3,raan_f,incl_f,thetaOpt+120,mu_venus);
x4 = [r4 v4];
%We use the same period as orbit 3, its the same orbit just inclined to our
%final orbit's inclination.
T4 = T3;
[~,sol] = ode45(@(t,x) two_body(x, mu_venus),[0 T3], x4, odeoptions);
markerThree(6) = plot3(sol(:,1),sol(:,2),sol(:,3), 'g', 'lineWidth', 2);
%% tof on Orbit 4
% the tof is half of the total period of orbit 4
tof4 = T4/2;
%% Orbit 4 becomes the Final Orbit
% ONE IMPULSE STATEGY SOLUTION
[delV_oneImp] = orbitChange(a_3,a_3,Rp_f,Ra_f,argp3,argp_f,mu_venus);
%Get the final orbit parameters into cartesian
[rf,vf] = Kepl_2_Cart(a_f,e_f,argp_f,raan_f,incl_f,thetaOpt,mu_venus);
xf = [rf,vf];
%Period of the final orbit
Tf = (2*pi*a_f^(3/2))/sqrt(mu_venus);
% Plot the Final Orbit
[~,sol] = ode45(@(t,x) two_body(x, mu_venus),[0 Tf], xf, odeoptions);
markerThree(7) = plot3(sol(:,1),sol(:,2),sol(:,3),'b', 'lineWidth', 2);
%% Plot tof to final position (true anomaly)
E0 = True2Ecc(0,e_f); %rad
```

```
M0 = Ecc2Mean(E0,e_f); %rad
Ef = True2Ecc(theta_f,e_f); %rad
Mf = Ecc2Mean(Ef,e_f); %rad
% Find time of flight to get to final position
tof4 = (Mf-M0)/sqrt(mu_venus/a_f^3);
%Plot tof4
[~,sol] = ode45(@(t,x) two_body(x, mu_venus),[0 tof4], xf, odeoptions);
markerThree(8) = plot3(sol(:,1),sol(:,2),sol(:,3),'c', 'lineWidth', 3);
%Markers for the start and end of tof5
markerThree(9) = plot3(sol(1,1),sol(1,2),sol(1,3), '*m', 'lineWidth', 3,
'MarkerSize', 8);
markerThree(10) = plot3(sol(end,1),sol(end,2),sol(end,3), '*k', 'lineWidth', 3,
'MarkerSize', 8);
%Legend for Figure 3
legend3 = legend(markerThree,'Hyperbolic Orbit 2','Orbit 3','Path to change to
Orbit 4', ...
   'Capture Point','Position to Change to Orbit 4','Orbit 4','Final Orbit', ...
   'Path to Final Position','Initial Position on Final Orbit (Perigee)','Final
Position of HBSB');
hold off;
% Total Needed Impulse for Mission!
delV_total = delV1 + del_Vph1 + del_Vph2+ delV3 + delV_oneImp;
% Total Transfer Time for Mission!
tof_Total = tof1 + tof2 + tof3 + tof4;
```

Main Functions:

```
function [r,v] = Kepl_2_Cart(a,ecc,argp,raan,inc,theta,mu)
p = a*(1-ecc^2);
r = p/(1+ecc*cosd(theta));
re = r*cosd(theta);
rp = r*sind(theta);
rk = 0;
ve = -sqrt(mu/p)*sind(theta);
vp = sqrt(mu/p)*(ecc+cosd(theta));
vk = 0;
r_orbit = [re;rp;rk];
v_orbit = [ve;vp;vk];
ROT313 = [cosd(argp)*cosd(raan)-sind(argp)*cosd(inc)*sind(raan),
-sind(argp)*cosd(raan)-cosd(argp)*cosd(inc)*sind(raan), sind(inc)*sind(raan);
   cosd(argp)*sind(raan)+sind(argp)*cosd(inc)*cosd(raan),
-sind(argp)*sind(raan)+cosd(argp)*cosd(inc)*cosd(raan), -sind(inc)*cosd(raan);
   sind(argp)*sind(inc), cosd(argp)*sind(inc), cosd(inc)];
r_matrix = ROT313*r_orbit;
v_matrix = ROT313*v_orbit;
x = r_matrix(1);
y = r_matrix(2);
z = r_matrix(3);
xdot = v_matrix(1);
```

```
ydot = v_matrix(2);
zdot = v_matrix(3);
r = [x;y;z]; %km
v = [xdot;ydot;zdot]; %km/s
end

function [theta1, theta2, delV, argp2] = PlaneChange(a, e, incl1, incl2, raan1,
raan2, argp1, mu)
% Gives impulse needed to change the plane of an orbit to another orbit of the
same shape/dimension.
% i.e. a1=a2 and e1=e2.
% Outputs [theta1,theta2,delV,argp2]
% Input units are km and degrees
% Output units are km/s and degrees
del_raan = raan2-raan1;
del_incl = incl2-incl1;
if ((del_raan > 0 && del_incl > 0) || (del_raan < 0 && del_incl < 0)) %
Concordant
alpha = acosd(cosd(incl1)*cosd(incl2) +
sind(incl1)*sind(incl2)*cosd(del_raan));
sinu1 = ((sind(del_raan)*sind(incl2))/sind(alpha));
sinu2 = ((sind(del_raan)*sind(incl1))/sind(alpha));
cosu1 = (cosd(incl2)-cosd(alpha)*cosd(incl1))/(-sind(alpha)*sind(incl1));
cosu2 = (cosd(incl1)-cosd(alpha)*cosd(incl2))/(sind(alpha)*sind(incl2));
u1 = atand(sinu1/cosu1);
u2 = atand(sinu2/cosu2);
theta1 = u1 - argp1;
theta2 = theta1;
argp2 = u2 - theta2;
% Make True Anomaly and Argument of Periapsis angles positive
if argp2 < 0
   argp2 = argp2 + 360;
end
if theta1 < 0
   theta1 = theta1 +360;
   theta2 = theta1;
end
% Calculating Impulse
p = a*(1-e^2);
vtheta = sqrt(mu/p)*(1+e*cosd(theta2));
delV = 2*vtheta*sind(alpha/2);
else % Discordant
alpha = acosd(cosd(incl1)*cosd(incl2) +
sind(incl1)*sind(incl2)*cosd(del_raan));
sinu1 = ((sind(del_raan)*sind(incl2))/sind(alpha));
sinu2 = ((sind(del_raan)*sind(incl1))/sind(alpha));
cosu1 = (cosd(incl2)-cosd(alpha)*cosd(incl1))/(sind(alpha)*sind(incl1));
cosu2 = (cosd(incl1)-cosd(alpha)*cosd(incl2))/(-sind(alpha)*sind(incl2));
u1 = atand(sinu1/cosu1);
```

```
u2 = atand(sinu2/cosu2);
theta1 = 2*pi - argp1 - u1;
theta2 = theta1;
argp2 = 2*pi - theta2 - u2;
% Make True Anomaly and Argument of Periapsis angles positive
if argp2 < 0
   argp2 = argp2 + 360;
end
if theta1 < 0
   theta1 = theta1 +360;
   theta2 = theta1;
end
%Calculating Impulse
p = a*(1-e^2);
vtheta = sqrt(mu/p)*(1+e*cosd(theta2));
delV = 2*vtheta*sind(alpha/2);
end
end

function [del_V] = reOrient(a, e, argp1, argp2, mu)
% Gives impulse needed to Re-Orient an orbit to another orbit of the same
shape/dimension.
% i.e. a1=a2 and e1=e2.
% Outputs [delV]
% Input units are km and degrees
% Output units are km/s
if argp1 < 0
   argp1 = argp1 +360;
else
   argp1 = argp1;
end
if argp2 < 0
   argp2 = argp2 +360;
else
   argp2 = argp2;
end
del_argp = argp2-argp1;
theta1  = del_argp/2;
theta2 = 360-theta1;
p = a*(1-e^2);
Vr = sqrt(mu/p)*e*sind(theta1);
del_V = 2*Vr;
end

function dx = two_body(x,mu)
r = sqrt(x(1)^2+x(2)^2+x(3)^2);
dx(1) = x(4);
dx(2) = x(5);
dx(3) = x(6);
```

```
dx(4) = -mu/r^3*x(1);
dx(5) = -mu/r^3*x(2);
dx(6) = -mu/r^3*x(3);
dx = [dx(1) dx(2) dx(3) dx(4) dx(5) dx(6)]';

end

function [delV] = orbitChange(rp1, ra1, rp2, ra2, argp1, argp2, mu)
a1 = (ra1+rp1)/2;
e1 = (ra1-rp1)/(ra1+rp1);
p1 = a1*(1-e1^2);
a2 = (ra2+rp2)/2;
e2 = (ra2-rp2)/(ra2+rp2);
p2 = a2*(1-e2^2);
del_argp = argp2-argp1;
%% One Impulse Strategy Method
%Change Shape/Dimension and do Re-Orientation at the same time
A = e1*p2 - e2*p1*cosd(del_argp);
B = -e2*p1*sind(del_argp);
C = p1-p2;
phi = atand(B/A);
theta1 = phi + acosd((C/A)*cosd(phi));
theta2 = theta1 - del_argp;
Vr1 = sqrt(mu/p1)*e1*sind(theta1);
Vtheta1 = sqrt(mu/p1)*(1+e1*cosd(theta1));
gamma1 = atand(Vr1/Vtheta1);
V1_vect = [Vr1 Vtheta1];
V1 = norm(V1_vect);
Vr2 = sqrt(mu/p2)*e2*sind(theta2);
Vtheta2 = sqrt(mu/p2)*(1+e2*cosd(theta2));
gamma2 = atand(Vr2/Vtheta2);
V2_vect = [Vr2 Vtheta2];
V2 = norm(V2_vect);
del_gamma = gamma2-gamma1;
delV = sqrt(V1^2 + V2^2 - 2*V1*V2*cosd(del_gamma));
%% Multiple Impulse Strategies to compare (If wanted by other group members)
% Use reOrient Function already coded!!!!
end
function E = True2Ecc(theta,e)
theta = theta*pi()/180; %Make true anomaly into radians
if e == 0
   E = theta;
end
if 0 < e && e < 1
   E = 2*atan(sqrt((1-e)/(1+e))*tan(theta/2));
end
if e == 1
   D = tan(theta/2);
   E = D;
```

```
end
if e > 1
   F = 2*atanh(sqrt((e-1)/(1+e))*tan(theta/2));
   E = F;
end
end

function M = Ecc2Mean(E,e)
if e == 0
   M = E;
end
if 0 < e && e < 1
   M = E - e.*sin(E);
end
if e == 1
   M = 0.5.*(E - E.^3/3);
end
if e > 1
   M = e.*sinh(E) - E;
end
end

function [Vinf, del_Vph, Beta, DELTA, a_h, e_h, Vph] = hype_arrival(R1, R2,
mu_planet, R_park, a_park, e_park)
%This function returns the infinite velocity of a hyperbola (Vinf), the impulse
%needed to arrive to a parking orbit specified in the sphere of influence
%of the planet we are entering (del_Vph), the angle in deg we need to orient
%ourselves with so the velocity of the planet will be parallel to Vinf (Beta),
%and the aiming radius (DELTA). It also returns the semimajor
%axis (a_h) and the eccentricity(e_h) of the departure hyperbolic orbit.
mu_sun = 1.32712*10^11; %km^3/s^2
a_H = (R1+R2)/2; % Hohmann Semimajor axis
V = sqrt(mu_sun/R2); % Velocity of the planet we are arriving to
Vp = sqrt((-mu_sun/a_H) + (2*mu_sun)/R2); % Perigee Velocity of Hohmann
Va = sqrt((-mu_sun/a_H) + (2*mu_sun)/R1); % Apogee Velocity of Hohmann
Ppark = a_park*(1-e_park^2);
Vpark = sqrt(mu_planet/Ppark)*(1+e_park);
delV = V - Vp;
Vinf = delV;
a_h = -mu_planet/Vinf^2;
Vph = sqrt((-mu_planet/a_h) +(2*mu_planet)/R_park);
del_Vph = Vph-Vpark;
r_ph = R_park;
e_h = 1-(r_ph/a_h);
Beta = acosd(1/e_h);
DELTA = R_park*sqrt(1+(2*mu_planet)/(R_park*Vinf^2));
end
```

```
function [Vinf, del_Vph, Beta, a_h, e_h, Vph] = hype_depart(R1, R2, mu_planet,
R_park, a_park, e_park)
%This function returns the infinite velocity of a hyperbola (Vinf), the impulse
%needed to escape the sphere of influence of the planet we are leaving
%(del_Vph),  and the the angle in deg we need to orient ourselves with so the
velocity of
%the planet will be parallel to Vinf (Beta). It also returns the semimajor
%axis (a_h) and the eccentricity(e_h) of the departure hyperbolic orbit.
%R1 is the distance of the planet we are leaving from the sun. R2 is the
%distance of the other planet or random position from the sun.
mu_sun = 1.32712*10^11; %km^3/s^2
a_H = (R1+R2)/2; % Hohmann Semimajor axis
V = sqrt(mu_sun/R1); % Velocity of the Planet we are departing
Vp = sqrt((-mu_sun/a_H) + (2*mu_sun)/R2); % Perigee Velocity of Hohmann
Va = sqrt((-mu_sun/a_H) + (2*mu_sun)/R1); % Apogee Velocity of Hohmann
Ppark = a_park*(1-e_park^2);
Vpark = sqrt(mu_planet/Ppark)*(1+e_park);
delV = V - Va;
Vinf = delV;
a_h = -mu_planet/Vinf^2; % Hyperbolic orbit semimajor axis for planet 1
Vph = sqrt((-mu_planet/a_h) +(2*mu_planet)/R_park);
del_Vph = Vph-Vpark;
r_ph = R_park;
e_h = 1-(r_ph/a_h); % eccentricity of the hyperbolic orbit
Beta = acosd(1/e_h);
end

function [xx,yy,zz] = venus_sphere(varargin)
%COPY OF EARTH_SPHERE THAT MAKES A VENUS SIZED SPHERE!!!!
%% Input Handling
[cax,args,nargs] = axescheck(varargin{:}); % Parse possible Axes input
error(nargchk(0,2,nargs)); % Ensure there are a valid number of inputs
% Handle remaining inputs.
% Should have 0 or 1 string input, 0 or 1 numeric input
j = 0;
k = 0;
n = 50; % default value
units = 'km'; % default value
for i = 1:nargs
if ischar(args{i})
units = args{i};
j = j+1;
elseif isnumeric(args{i})
n = args{i};
k = k+1;
end
end
if j > 1 || k > 1
error('Invalid input types')
```

```
end
%% Calculations
% Scale factors
Scale = {'km' 'm' 'mile' 'miles' 'nm' 'au' 'ft';
1 1000 0.621371192237334 0.621371192237334 0.539956803455724
6.6845871226706e-009 3280.839895};
% Identify which scale to use
try
myscale = 6052.1363*Scale{2,strcmpi(Scale(1,:),units)};
catch %#ok<*CTCH>
error('Invalid units requested. Please use m, km, ft, mile, miles, nm, or AU')
end
% -pi <= theta <= pi is a row vector.
% -pi/2 <= phi <= pi/2 is a column vector.
theta = (-n:2:n)/n*pi;
phi = (-n:2:n)'/n*pi/2;
cosphi = cos(phi); cosphi(1) = 0; cosphi(n+1) = 0;
sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;
x = myscale*cosphi*cos(theta);
y = myscale*cosphi*sintheta;
z = myscale*sin(phi)*ones(1,n+1);
%% Plotting
if nargout == 0
cax = newplot(cax);
% Load and define topographic data
load('topo.mat','topo','topomap1');
% Rotate data to be consistent with the Earth-Centered-Earth-Fixed
% coordinate conventions. X axis goes through the prime meridian.
%http://en.wikipedia.org/wiki/Geodetic_system#Earth_Centred_Earth_Fixed_.28ECEF
_or_E
% CF.29_coordinates
%
% Note that if you plot orbit trajectories in the Earth-Centered-
% Inertial, the orientation of the contintents will be misleading.
topo2 = [topo(:,181:360) topo(:,1:180)]; %#ok<NODEF>
% Define surface settings
props.FaceColor= 'texture';
props.EdgeColor = 'none';
props.FaceLighting = 'phong';
props.Cdata = topo2;
% Create the sphere with Earth topography and adjust colormap
surface(x,y,z,props,'parent',cax)
colormap(topomap1)
% Replace the calls to surface and colormap with these lines if you do
% not want the Earth's topography displayed.
% surf(x,y,z,'parent',cax)
% shading flat
% colormap gray
% Refine figure
```

```
imData = imread('venus.png');
ch = get(gca,'children');
set(ch,'cdata',imData,'edgecolor','none');
axis equal
xlabel(['X [' units ']'])
ylabel(['Y [' units ']'])
zlabel(['Z [' units ']'])
view(127.5,30)
else
xx = x; yy = y; zz = z;
end

function [xx,yy,zz] = sun_sphere(varargin)
%COPY OF EARTH_SPHERE THAT MAKES A SPHERE LOOK LIKE THE SUN,
%INCORRECT SIZE, ONLY USED TO SHOW UP ON INTERPLANETARY HOHMANN PLOT
%% Input Handling
[cax,args,nargs] = axescheck(varargin{:}); % Parse possible Axes input
error(nargchk(0,2,nargs)); % Ensure there are a valid number of inputs
% Handle remaining inputs.
% Should have 0 or 1 string input, 0 or 1 numeric input
j = 0;
k = 0;
n = 50; % default value
units = 'km'; % default value
for i = 1:nargs
if ischar(args{i})
units = args{i};
j = j+1;
elseif isnumeric(args{i})
n = args{i};
k = k+1;
end
end
if j > 1 || k > 1
error('Invalid input types')
end
%% Calculations
% Scale factors
Scale = {'km' 'm' 'mile' 'miles' 'nm' 'au' 'ft';
1 1000 0.621371192237334 0.621371192237334 0.539956803455724
6.6845871226706e-009 3280.839895};
% Identify which scale to use
try
myscale = 15000000.1363*Scale{2,strcmpi(Scale(1,:),units)};
catch %#ok<*CTCH>
error('Invalid units requested. Please use m, km, ft, mile, miles, nm, or AU')
end
% -pi <= theta <= pi is a row vector.
% -pi/2 <= phi <= pi/2 is a column vector.
```

```
theta = (-n:2:n)/n*pi;
phi = (-n:2:n)'/n*pi/2;
cosphi = cos(phi); cosphi(1) = 0; cosphi(n+1) = 0;
sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;
x = myscale*cosphi*cos(theta);
y = myscale*cosphi*sintheta;
z = myscale*sin(phi)*ones(1,n+1);
%% Plotting
if nargout == 0
cax = newplot(cax);
% Load and define topographic data
load('topo.mat','topo','topomap1');
% Rotate data to be consistent with the Earth-Centered-Earth-Fixed
% coordinate conventions. X axis goes through the prime meridian.
%http://en.wikipedia.org/wiki/Geodetic_system#Earth_Centred_Earth_Fixed_.28ECEF
_or_E
% CF.29_coordinates
%
% Note that if you plot orbit trajectories in the Earth-Centered-
% Inertial, the orientation of the contintents will be misleading.
topo2 = [topo(:,181:360) topo(:,1:180)]; %#ok<NODEF>
% Define surface settings
props.FaceColor= 'texture';
props.EdgeColor = 'none';
props.FaceLighting = 'phong';
props.Cdata = topo2;
% Create the sphere with Earth topography and adjust colormap
surface(x,y,z,props,'parent',cax)
colormap(topomap1)
% Replace the calls to surface and colormap with these lines if you do
% not want the Earth's topography displayed.
% surf(x,y,z,'parent',cax)
% shading flat
% colormap gray
% Refine figure
imData = imread('sun.png');
ch = get(gca,'children');
set(ch,'cdata',imData,'edgecolor','none');
axis equal
xlabel(['X [' units ']'])
ylabel(['Y [' units ']'])
zlabel(['Z [' units ']'])
view(127.5,30)
else
xx = x; yy = y; zz = z;
end

function [xx,yy,zz] = earth_sphere(varargin)
%EARTH_SPHERE Generate an earth-sized sphere.
```

```
% [X,Y,Z] = EARTH_SPHERE(N) generates three (N+1)-by-(N+1)
% matrices so that SURFACE(X,Y,Z) produces a sphere equal to
% the radius of the earth in kilometers. The continents will be
% displayed.
%
% [X,Y,Z] = EARTH_SPHERE uses N = 50.
%
% EARTH_SPHERE(N) and just EARTH_SPHERE graph the earth as a
% SURFACE and do not return anything.
%
% EARTH_SPHERE(N,'mile') graphs the earth with miles as the unit rather
% than kilometers. Other valid inputs are 'ft' 'm' 'nm' 'miles' and 'AU'
% for feet, meters, nautical miles, miles, and astronomical units
% respectively.
%
% EARTH_SPHERE(AX,...) plots into AX instead of GCA.
%
% Examples:
% earth_sphere('nm') produces an earth-sized sphere in nautical miles
%
% earth_sphere(10,'AU') produces 10 point mesh of the Earth in
% astronomical units
%
% h1 = gca;
% earth_sphere(h1,'mile')
% hold on
% plot3(x,y,z)
% produces the Earth in miles on axis h1 and plots a trajectory from
% variables x, y, and z
% Clay M. Thompson 4-24-1991, CBM 8-21-92.
% Will Campbell, 3-30-2010
% Copyright 1984-2010 The MathWorks, Inc.
%% Input Handling
[cax,args,nargs] = axescheck(varargin{:}); % Parse possible Axes input
error(nargchk(0,2,nargs)); % Ensure there are a valid number of inputs
% Handle remaining inputs.
% Should have 0 or 1 string input, 0 or 1 numeric input
j = 0;
k = 0;
n = 50; % default value
units = 'km'; % default value
for i = 1:nargs
if ischar(args{i})
units = args{i};
j = j+1;
elseif isnumeric(args{i})
n = args{i};
k = k+1;
end
```

```
end
if j > 1 || k > 1
error('Invalid input types')
end
%% Calculations
% Scale factors
Scale = {'km' 'm' 'mile' 'miles' 'nm' 'au' 'ft';
1 1000 0.621371192237334 0.621371192237334 0.539956803455724
6.6845871226706e-009 3280.839895};
% Identify which scale to use
try
myscale = 6378.1363*Scale{2,strcmpi(Scale(1,:),units)};
catch %#ok<*CTCH>
error('Invalid units requested. Please use m, km, ft, mile, miles, nm, or AU')
end
% -pi <= theta <= pi is a row vector.
% -pi/2 <= phi <= pi/2 is a column vector.
theta = (-n:2:n)/n*pi;
phi = (-n:2:n)'/n*pi/2;
cosphi = cos(phi); cosphi(1) = 0; cosphi(n+1) = 0;
sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;
x = myscale*cosphi*cos(theta);
y = myscale*cosphi*sintheta;
z = myscale*sin(phi)*ones(1,n+1);
%% Plotting
if nargout == 0
cax = newplot(cax);
% Load and define topographic data
load('topo.mat','topo','topomap1');
% Rotate data to be consistent with the Earth-Centered-Earth-Fixed
% coordinate conventions. X axis goes through the prime meridian.
%http://en.wikipedia.org/wiki/Geodetic_system#Earth_Centred_Earth_Fixed_.28ECEF
_or_E
% CF.29_coordinates
%
% Note that if you plot orbit trajectories in the Earth-Centered-
% Inertial, the orientation of the contintents will be misleading.
topo2 = [topo(:,181:360) topo(:,1:180)]; %#ok<NODEF>
% Define surface settings
props.FaceColor= 'texture';
props.EdgeColor = 'none';
props.FaceLighting = 'phong';
props.Cdata = topo2;
% Create the sphere with Earth topography and adjust colormap
surface(x,y,z,props,'parent',cax)
colormap(topomap1)
% Replace the calls to surface and colormap with these lines if you do
% not want the Earth's topography displayed.
% surf(x,y,z,'parent',cax)
```

```
% shading flat
% colormap gray
% Refine figure
axis equal
xlabel(['X [' units ']'])
ylabel(['Y [' units ']'])
zlabel(['Z [' units ']'])
view(127.5,30)
else
xx = x; yy = y; zz = z;
end
```