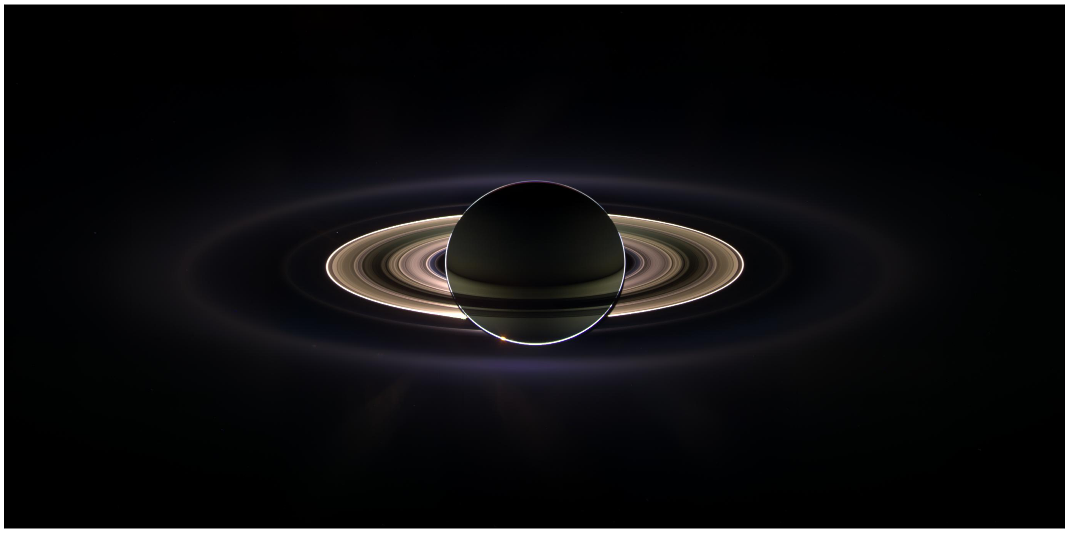
Cassini–Huygens Mission Analysis

David Arnold, Noah Curry, Rori Fortmann,
Will Hendrickson, Ricardo Herrera, and Evan Hochwalt
AA 310 Orbital Mechanics
William E. Boeing Department of Aeronautics
University of Washington
Seattle, WA 98195

2018-12-05

Abstract: The Cassini-Huygens mission, commonly called Cassini, was a joint effort between the European Space Agency (ESA), who designed and assembled the Huygens lander, and National Aeronautics and Space Administration (NASA). Cassini's mission goals were to study the moons and rings of Saturn along with sending the Huygens lander to the surface of Titan, Saturns largest moon [1]. Cassini was launched aboard a Titan IV-B rocket on $October\,15^{th}, 1997$, at Cape Canaveral Air Force Station, in Florida, USA [1]. Cassinis mission lasted nearly 20 years, with 13 of those years spent orbiting Saturn. The trajectory of Cassini consisted of planetary gravity assists from Venus ($April\,26^{th}, 1998$ and $June\,24^{th}, 1999$), Earth ($August\,18^{th}, 1999$) and Jupiter ($December\,30^{th}, 2000$) before entering Saturns orbit on $July\,1^{st}, 2004$ [1]. The Cassini-Huygens mission is widely perceived to be a success well beyond its original mission expectations.



Contents

List of Figures

Division of Labor

Noah Curry

Launch
Venus Flyby 1
Venus Flyby 2
Earth/Moon Flyby
Earth Impact Analysis

Will Hendrickson

Launch
Venus Flybys 1
Venus Flyby 2
Earth/Moon Flyby
Earth Impact Analysis

David Arnold

Orbital Insertion at Saturn
Titan Flybys and Ensuring Orbital Resonance
Huygens Planned Trajectory and Emergency Alteration
General Conclusion Writing
General Trajectory Analysis Support

Rori Fortmann

Citations
Table of Contents
Mission phases
Spacecraft
Subsystems
The Mission
Jupiter Flyby

Ricardo A. Herrera

Titan Lander
Spacecraft subsystems
Spacecraft
Launch
The Mission
Mission Phases

Evan Hochwalt

Reaction Wheels
Finding Mission Plans
End of mission
Deorbit into Saturn
Grand Finale

I. The Mission

I.A. Goals, Objectives, Time-line and Partnerships

The Cassini-Huygens mission was a cooperative project of NASA and the European Space Agency with the project managed by JPL. Cassini-Huygens launched on 15 Oct 1997 and used successive gravity assist maneuvers around Venus, Venus, Earth and finally Jupiter on its way to an arrival at Saturn on 1 July 2004. The missions scientific goals were split into the five following categories: Saturn, its ring system, the magnetosphere, the icy moons, and Titan. For both Saturn and Titan there was specific interest in the atmosphere, namely its composition and vertical distribution; and, in terms of the Saturn's rings, goals included study of composition, distribution and interaction with embedded moons in the rings [1]. Investigations into the global configuration of the magnetosphere were conducted in addition to its interactions with the icy moons. The Cassini-Huygens mission was extended twice in July 2008 and and again in July 2010, both times with the intent of furthering the investigations of the primary mission over the course of many of Saturn's seasons[1].

I.B. Spacecraft, Size, Weight and Design Requirements

The mission spacecraft consisted of two main components: the ASI/NASA Cassini orbiter, named for the Italian astronomer Giovanni Domenico Cassini, discoverer of Saturn's ring divisions and four of its satellites; and the ESA-developed Huygens probe, named for the Dutch astronomer, mathematician and physicist Christiaan Huygens, discoverer of Titan [1]. The Cassini orbiter had a mass of 2125[kg], the Huygens probe had a mass of 320[kg], in addition there was 3,267[kg] of liquid propellant for a total mass of 5,712 [kg]: the spacecraft was approximately 6.7 by 4 meter in size[4]. In addition the space craft was designed to withstand large temperature changes and employed the use of thermal blankets that not only insulated the craft but also served as protection from radiation and micro-impacts from dust and other small debris.

II. Spacecraft Subsystems

II.A. Spacecraft Diagram

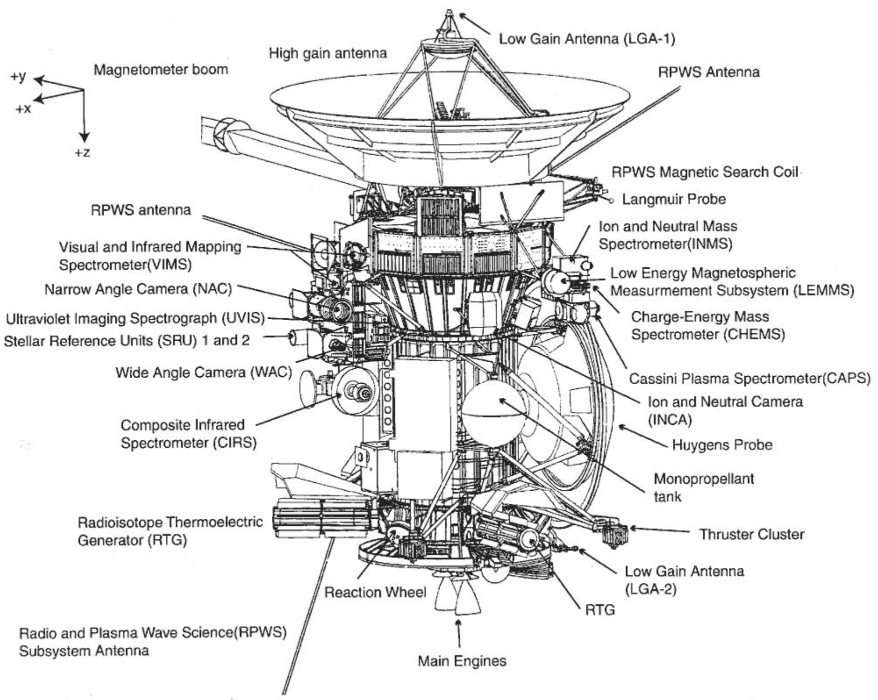

Low Gain Antenna (LGA-1)
High gain antenna
Magnetometer boom
+y
+x
+z
RPWS Antenna
RPWS Magnetic Search Coil
Langmuir Probe
RPWS antenna
Ion and Neutral Mass Spectrometer(INMS)
Visual and Infrared Mapping Spectrometer(VIMS)
Low Energy Magnetospheric Measurmement Subsystem (LEMMS)
Narrow Angle Camera (NAC)
Charge-Energy Mass Spectrometer (CHEMS)
Ultraviolet Imaging Spectrograph (UVIS)
Stellar Reference Units (SRU) 1 and 2
Wide Angle Camera (WAC)
Cassini Plasma Spectrometer(CAPS)
Ion and Neutral Camera (INCA)
Composite Infrared Spectrometer (CIRS)
Huygens Probe
Monopropellant tank
Radioisotope Thermoelectric Generator (RTG)
Thruster Cluster
Low Gain Antenna (LGA-2)
Reaction Wheel
RTG
Radio and Plasma Wave Science(RPWS) Subsystem Antenna
Main Engines

Fig. 1: Diagram of some of the subsystems discussed [2]

II.B. Power

Due to the extended length of the mission, Cassini was powered by three radioisotope thermoelectric generators (RTG). The heat generated by 32.7 kg of plutonium-238, from the material's radioactive decay was converted into electricity to power the instrumentation and on-board systems while using no moving parts. Huygens was powered by Cassini during cruise, but used internal batteries during its decent to the surface of Titan [5].

II.C. Propulsion

Cassini utilizes a main engine and an identical backup engine. The backup engine was never used during the course of the mission. The main engines were a bipropellant hypergolic engine. Hypergolic engine setups have a major advantage in that when the two propellants are mixed, they combust without a spark. The specific impulse was $I_{sp} = 312s$. This is the same legendary R-4D engine used in the Apollo attitude control thrusters [6]. Cassini also had 16 attitude control thrusters 8 of which were redundant. These thrusters were a monopropellant hydrazine system. [7]

II.D. Instrumentation

Cassini was equipped with twelve scientific instruments to measure and record the atmosphere and magnetosphere of Saturn and its moons and rings. Cassini's insturments were divided into three catagories; Optical Remote Sensing sensors, Fields-Particles-and-Waves sensors, and Microwave Remote Sensing sensors [4].

II.E. Composite Infrared Spectrometer (CIRS)

The CIRS sensor was used measure the temperatures of the celestial bodies in the Saturn system by measuring the emitted infrared radiation. CIRS measured the radiation in the atmosphere, rings and surfaces in the Saturn system. It also used the data to generate a 3D model of Saturns magnetosphere. CIRS was also able to determine the pressure and temperatures of Saturn as a function of altitude, along with the composition of gases and aerosols in the atmosphere [8].

II.F. Imaging Science Subsystem (ISS)

The ISS was a visible light camera that captured most of the images of Saturn and its systems celestial bodies. In addition to capturing images in the visible light spectrum, the ISS sensor was sensitive to ultraviolet and infrared light waves [8].

II.G. Ultraviolet Imaging Spectrograph (UVIS)

The UVIS was a sensor that captured images in the ultraviolet light reflected off an object, e.g. the clouds and rings of Saturn, to measure and record the composition and structure of the objects. This sensor was unique in that it could analyze the spectral composition and spatial measurements of particles gasses. UVIS was able to take multiple images and stitch them together to make a movies of the movement of the particles and gasses in the atmosphere [8].

II.H. Visible and Infrared Mapping Spectrometer (VIMS)

The VIMS instrument was used to capture images in the visible and infrared spectrum. VIMS was able to use this information to learn more about the composition of Saturn and the moons and rings in its system. VISM was also utilized to measure the sunlight passing through the rings to learn more about their structure. VIMS was able to map the movement of clouds to analyze weather patterns on Saturn [8].

II.I. Dual Technique Magnetometer (MAG)

The MAG was an instrument that measured the strength and direction of Saturn's magnetosphere [9]. Because the magnetic flux is partially caused by Saturn's molten core, MAG is able to analyze Saturn's core. Another goal of the MAG instrument was to generate a three-dimensional model of the magnetosphere of Saturn and her moons and rings.

II.J. Magnetospheric Imaging Instrument (MIMI)

The MIMI produced images of the ions, electrons, and energetic neutral atoms trapped in Saturn's magnetosphere. This information was used to study the solar winds interaction with Saturn's magnetosphere and the rest of the Saturnian System [8].

II.K. Radio and Plasma Wave Science Instrument (RPWS)

The RPWS measured the radio waves given off by the interaction of the solar wind with Saturn and and its moon and ring system. The RPWS also measured the electric and magnetic wave fields in Saturn's magnetospheres. The RPWS studied the configuration of Saturn's magnetic field and its relationship to Saturn Radiation, as well as monitoring and mapping Saturn's ionosphere, plasma, and lightning from Saturns atmosphere [8].

II.L. Cassini Plasma Spectrometer (CAPS)

CAPS was an instrument that measure the solar winds and its interaction with Saturns magnetosphere [8]. It was permanently disabled after a soft electrical short was detected first in June of 2011 and then again in May of 2012 [10].

II.M. Cosmic Dust Analyzer (CDA)

The CDA was an instrument that measured the size, speed, and direction of grains of dust and their respective chemical makeups near Saturn [10].

II.N. Ion and Neutral Mass Spectrometer (INMS)

The INMS used a quadruple mass spectrometer to measure the composition of particles near Titan and Saturn to learn more about their atmospheres Saturn's moons and rings[8].

II.O. Radar

Cassinis radar system produced maps of Titan's surface by sending radar wave that were powerful enough to penetrate Titans atmosphere, and measuring the return time of the signals to determine the height of large surface features. This gave the first glimpse of the surface features of Titan. Cassini also used its radar systems antenna to measure and record the radio waves in and around the Saturn system [8].

II.P. Radio Science Subsystem (RSS)

The RSS was a two-way radio system between a ground station on earth and an antenna on Cassini. The sensor system analyzed the changes in the radio waves as they transmitted through objects (e.g. the atmosphere and rings of Saturn) and used the information to compute the composition, pressures, and temperature of the atmosphere and ionosphere. It was also able to record sizes and distribution of particles within the sphere of influence (SOI) of Saturn. [8].

II.Q. Reaction Wheel

Cassini was equipped with three solid mounted reaction wheels and a fourth wheel that was attached via a gimbled system. The gimbled reaction wheel was capable as acting as a backup reaction wheel should one of the other wheels fail during Cassini's nearly 20 year mission. The reactions wheels acted as a system that could reorient the spacecraft by spinning a large flywheel to utilize the conservation of angular momentum and rotate the spacecraft. This system has an advantage over conventional thrusters because the reaction wheels work with electric motors, so the fuel can be reserved for for Δv maneuvers [11].

II.R. Communication

Cassini was equipped with three antennas: two low gain antennas and one high gain antenna. The high gain antenna was used to communicate with Earth. Data was also collected as the antenna sent out signals of various wavelengths that were changed as they passed through Saturn's atmosphere, moons and rings, providing information for study upon being received back on Earth [12].

II.S. Titan Lander

Named after the discoverer of Saturn's largest moon Titan, Christian Huygens; the Huygens lander separated from Cassini on December 25th, 2004, and landed on Titan on January 13th, 2005 [13]. The probe was left in a state of hibernation for 22 days while it descended towards Titan. Huygens came out of hibernation 15 minutes before initial contact with Titans atmosphere. The lander was powered by chemical batteries that were sized to last the full decent to Titans surface with an additional 30 min for surface measurements. The probe measured 2.7 meters in diameter and weighed roughly 318 kg [8].

The Huygens lander housed six main sensor arrays. They included a Huygens Atmospheric Structure Instrument (HASI) that hosted multiple sensors to measure the electrical properties of Titan's atmosphere, density and temperature of the atmosphere, and a sensor to measure wind gusts and wave motion should it have had a liquid landing. A Doppler Wind Experiment (DWE) measured wind speeds while descending onto Titan. The Descent Imager/Spectral Radiometer (DISR) was a camera system that imaged the surface of Titan in the visible, infrared, ultraviolet light spectra. A Gas Chromatograph Mass Spectrometer (GC/MS) to measure the gas composition of Titans atmosphere and soil. An Aerosol Collector and Pyrolyser (ACP) looked for organic material in the atmosphere. And the Surface Science Package (SSP); that was a sensor array to measure the speed of sound in Titans atmosphere and a probe that would penetrate the surface to measure the elasticity of the soil [13].

III. Mission Phases and Analysis

III.A. Launch

In 1997 the Cassini orbiter and the attached Huygens probe spacecraft was placed on board a Titan IV/ Centaur rocket in Cape Canaveral, Florida in preparation for an October

15th, 1997 launch. The Titan IVB/Centaur rocket was the most capable heavy lift space-launch vehicle available to NASA at the time and allowed for a maximum launch energy of 34 $\frac{km^2}{s^2}$ [14]. However, for an object of Cassinis mass, the estimated required launch energy was 108 $\frac{km^2}{s^2}$ [14]. The large difference between these two energy values required Cassini to fly a complex multi-gravity assist trajectory in order to successfully propel itself from Earth to the deep space Saturnian zone.

III.B. Venus Venus Earth Gravitational Assists

There were two multi-gravity assist trajectories proposed by the Jet Propulsion Laboratory (JPL) for phase I of Cassini's mission. The first required the spacecraft to drop into a Venus rendezvous orbit where a gravity assist from the planet would push it back outwards toward Earth. Once there, a second gravity assist would increase its semi-major axis, and a final Earth flyby would sling shot the spacecraft outwards toward Jupiter. This trajectory was called a Venus-Earth-Earth gravity assist (VEEGA). The second trajectory was a Venus-Venus-Earth gravity assist (VVEGA) where the spacecraft would drop into a Venus rendezvous orbit, perform a flyby and trajectory correction maneuver (TCM), then perform a second flyby of Venus where it would use another gravity assist to turn it outwards toward Earth on a quick rendezvous path for a final Earth-gravity-assist to push the spacecraft further outwards toward Jupiter.

Due to the spacecrafts four RTGs, there was a larger-than-average risk associated with trajectory determination. Government agencies were concerned about a trajectory anomaly resulting in an Earth impact on one of the three proposed home-planet flybys. The concern was great enough that JPL calculated the probability of an unexpected impact during one of the Earth flybys associated with a number of different mechanical and astronomical anomalies. These included solar pressure acceleration of the spacecraft and trajectory alteration due to micrometeorite impact. Using the Monte Carlo method, which generates a simulation based on possible outcomes of n variables [15], JPL created the equation shown in Fig. 2 to analyze the potential for an impact.

$$P_I = \sum_i P_F(i) \cdot P_{\frac{I}{F}}(i) \cdot P_{NR}(i)$$

Fig. 2: JPL Impact Analysis Equation

Here, P_I is the probability of earth impact, $P_F(i)$ is the probability of failure for the ith failure mode, $P_{\frac{I}{F}}$ is the probability of an earth impact trajectory in an failure event of the ith failure mode, and $P_{NR}(i)$ is the probability of no recovery while Cassini was on an impact trajectory with Earth [16]. The project requirement was set at an earth impact probability of less than one in a million, or less than 10^{-6}. The impact probability was calculated to be 7.6x10^{-7} for VEEGA and 8.3x10^{-7} for VVEGA. Since the primary trajectory met the project requirements, and allowed for an earlier launch opportunity, the VVEGA trajectory was chosen.

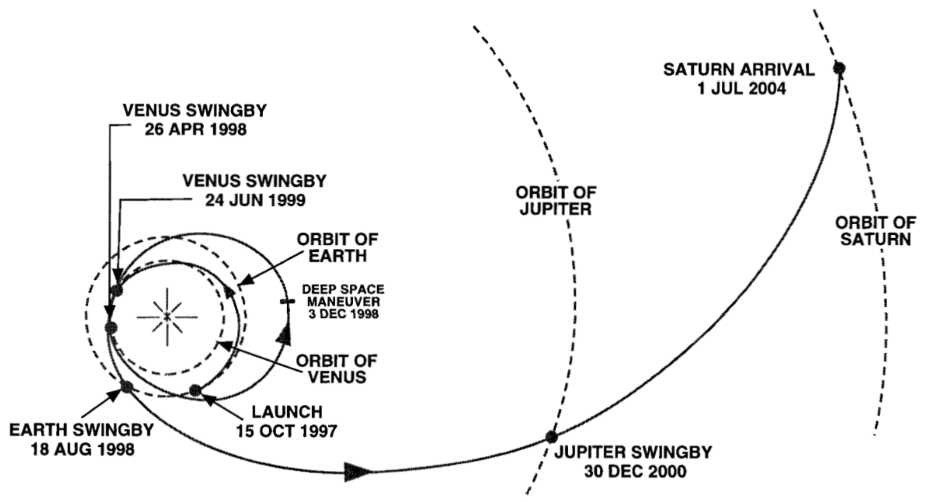


Fig. 3: Cassini's Interplanetary Trajectory [3]

Liftoff occurred at 8:27 UT on October 15. Cassini successfully entered a hyperbolic Earth escape trajectory approximately 40 minutes later [17] and the spacecraft reported an excess velocity of $V_\infty = 4.07\frac{km}{s}$ [2]. JPL planned for three TCM's between earth's escape and arrival at Venus. However, only two were executed. The first was to correct for irregularities in the escape orbit insertion and the second was to account for solar radiation pressure acceleration. After analyzing the trajectory data prior to the third TCM, the spacecraft was deemed "on course" and the third TCM was canceled. Since fuel is a precious resource on any space mission, every opportunity to extend the spacecraft's resources was seized. After traveling through the inner solar system for 6.5 months, the spacecraft approached Venus for its first gravity assist. An approach velocity was reported as $V_\infty = 6.03\frac{km}{s}$ with a turn angle of $\delta = 71.53°$. Using this V_∞ value, the hyperbolic eccentricity, turn angle, angular momentum, and periapsis velocity were calculated from the following equations:

Table 1: Equations for Analyzing Venus Flybys

$$R_{venus} = 6052km$$
$$r_p = R_{venus} + alt$$
$$e = 1 + \frac{r_p V_\infty^2}{\mu_v}$$
$$\delta = 2sin^{-1}(\frac{1}{e})$$
$$h = r_p\sqrt{V_\infty^2 + \frac{2\mu_v}{r_p}}$$
$$v_p = \sqrt{V_\infty^2 + \frac{2\mu_v}{r_p}}$$

Table 2: Orbital Elements for Venus Flyby 1

	Reported	Calculated	% Difference
e	1.711	1.709	0.122%
δ	71.53°	71.62 °	0.126%
v_p	11.8 $\frac{km}{s}$	11.78 $\frac{km}{s}$	0.116%

On its 14 month Venus-Venus orbit, the spacecraft performed a crucial deep space maneuver (DSM) near aphelion in order to decrease its perihelion on the second planetary flyby. The DSM reportedly increased the spacecraft's velocity by $452\frac{m}{s}$. After 617 days of space flight, Cassini approached Venus on its second flyby with an excess velocity of $V_\infty = 9.41\frac{km}{s}$ [18]. It utilized the planets gravity to turn through an angle $\delta = 41.65°$ onto a rapid Earth flyby trajectory where it rendezvoused with the home planet only 55 days later. The phasing of Earth and Venus was carefully chosen so that the energy requirement for Cassini to make it to Jupiter could be met by performing both gravity assists plus the DSM. This triple gravity-assist trajectory has been called "the most complex orbital maneuver to date [19]." As it flew by Earth on its final inner planetary maneuver, Cassini had an excess earth-relative velocity of $V_\infty = 16.01\frac{km}{s}$. The Earth-moon gravity assist increased Cassini's speed by $5.5\frac{km}{s}$ [20], sending the spacecraft hurtling on a non-Hohmann trajectory for Jupiter. The true eccentricity and turn angle data could not be found. As an alternative, the reported eccentricity values were obtained via the Jet Propulsion Laboratory HORIZON web-interface [21] and the turn angles were computed using these eccentricities.

Table 3: Earth Flyby

	Reported	Calculated	% Difference
e	5.857	5.857	0%
δ	19.66	19.66°	0%
v_p	19.0 $\frac{km}{s}$	19.02 $\frac{km}{s}$	0.105%

Table 4: Orbital Elements for Second Venus Flyby

	Reported	Calculated	% Difference
e	2.813	2.726	3.07%
δ	41.65°	43.03 °	3.207%
v_p	13.6 $\frac{km}{s}$	13.824 $\frac{km}{s}$	1.647%

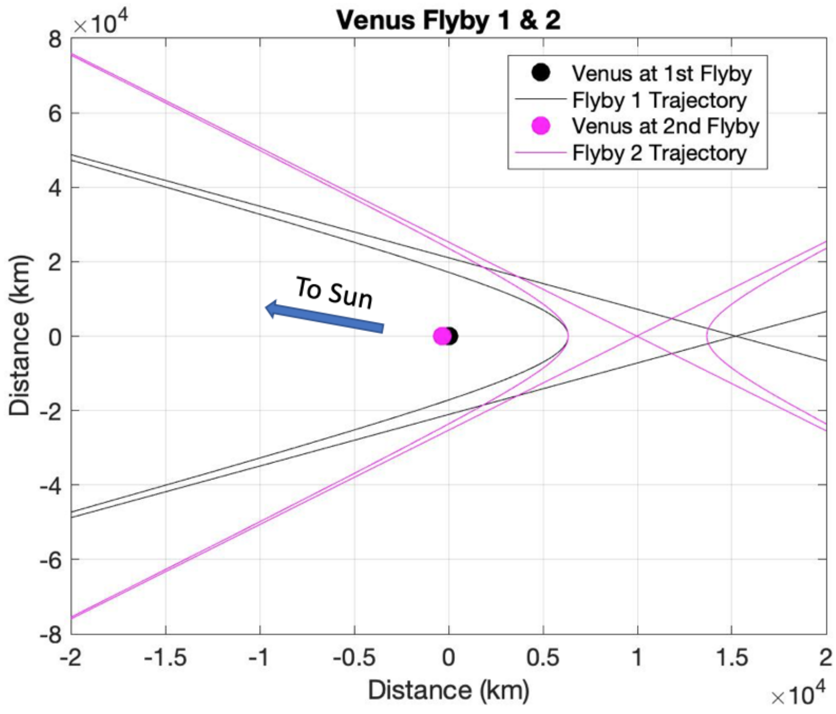


Fig. 4: Reconstruction of hyperbolic trajectory for first and second Venus flybys. Venus modeled as point mass.

III.C. Earth-Venus Analysis Complications

In an attempt to reconstruct the non-Hohmann transfer orbit from Earth to Venus, matching a calculated V_∞ with that of the C_3 value from the mission report was met with major complication. The multiple reconstruction approaches are shown in Fig. 5 along with the reported characteristic energy and excess velocity from Earth launch. The errors in each approach are still unknown and further investigation into the launch energy data and launch trajectory is necessary to isolate the cause of the $\approx 30\%$ difference between calculations and the reported value. The consistency between each calculation was noteworthy since the largest percent difference was only 10.43%. This signifies that there may be an analysis component missing from each of the separate approaches.

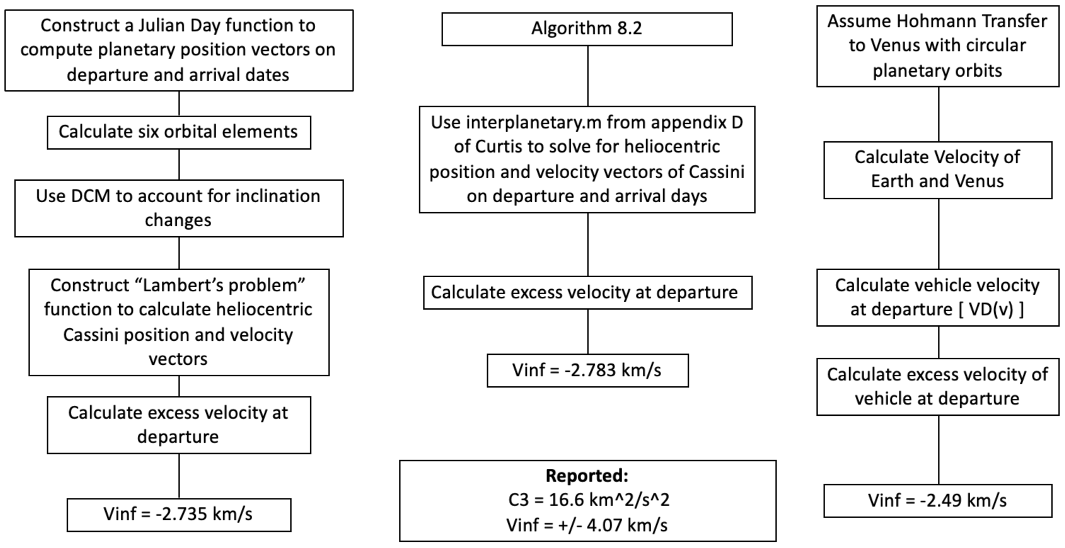

Construct a Julian Day function to compute planetary position vectors on departure and arrival dates
Calculate six orbital elements
Use DCM to account for inclination changes
Construct "Lambert's problem" function to calculate heliocentric Cassini position and velocity vectors
Calculate excess velocity at departure
Vinf = -2.735 km/s
Algorithm 8.2
Use interplanetary.m from appendix D of Curtis to solve for heliocentric position and velocity vectors of Cassini on departure and arrival days
Calculate excess velocity at departure
Vinf = -2.783 km/s
Reported:
C3 = 16.6 km^2/s^2
Vinf = +/- 4.07 km/s
Assume Hohmann Transfer to Venus with circular planetary orbits
Calculate Velocity of Earth and Venus
Calculate vehicle velocity at departure [VD(v)]
Calculate excess velocity of vehicle at departure
Vinf = -2.49 km/s

Fig. 5: Earth-Venus transfer analysis road map

III.D. Jupiter Gravitational Assists

Cassini completed an Earth flyby on its way to Jupiter on August 18, 1997 at 03:28 UT, and completed the Jupiter flyby, to be analyzed in this section, on December 30, 2000 at 10:05 UT [22, 23]. Along the Earth-Jupiter orbit, four TCM were scheduled although only two would end up being necessary to ensure Cassini's proper approach to Jupiter, these TCM are neglected in the analysis below due to lack of ability to evaluate them [24]. The times above are the times of Cassini's closeted approach to the respective planets. To calculate the state vectors at the above times, Algorithm 8.2 and the m-files provided by Curtis were employed[25]. As the times that Cassini departed Earth's SOI and arrived at Jupiter's SOI could not be found the times used to solve Algorithm 8.2 are the times at periapsis.

$$\begin{aligned}
\vec{R}_{Earth} &= [1.2372, -0.8734, -0.0000] * 10^8[\text{km}] \quad & R_{Earth} &= 1.5144 * 10^8[\text{km}] \\
\vec{V}_{Earth} &= [16.6938, 24.2249, 0.0000][\text{km/s}] \quad & V_{Earth} &= 29.4198[\text{km/s}] \\
\vec{R}_{Jupiter} &= [2.7002, 7.0522, -0.0897] * 10^8[\text{km}] \quad & R_{Jupiter} &= 7.5520 * 10^8[\text{km}] \\
\vec{V}_{Jupiter} &= [-12.3662, 5.2883, 0.2550][\text{km/s}] \quad & V_{Jupiter} &= 13.4520[\text{km/s}]
\end{aligned}$$

On departing Earth the time of flight to Jupiter was calculated to be 500.28 days on a Non-Hohmann trajectory. From the perspective of the sun the spacecrafts initial and final position are the same as that of the Earth and Jupiter respectively and as part of implementing Algorithm 8.2, Lambert's problem is solved using Algorithm 5.2 so that the velocity of Cassini relative to the sun at arrival to Jupiter was calculated to be [25]:

$$\vec{V}_{Arrival} = [-3.5863, 11.2802, -0.0876][km/s] \quad V_{Arrival} = 11.7887[\text{km/s}]$$

The hyperbolic excess velocity relative to Jupiter was then calculated by subtracting the velocity of the planet around the sun from the velocity above:

$$\vec{V}_{\infty)Arrival} = [8.7799, 5.9919, -.03425][\text{km/s}] \quad V_{\infty)Arrival} = 10.6352[\text{km/s}]$$

Cassini reached its closest approach to Jupiter on 2000 Dec 12 at 10:05 UT with an altitude of 9,723,890[km] [2]. From this altitude the radius at periapsis was calculated to be 9,795,380[km] and using the calculated hyperbolic excess velocity the following orbital elements were calculated where $\mu_j = 126686000[\frac{km^3}{s^2}]$ [25]:

Table 5: Orbital elements for Jupiter Flyby

$$e = 1 + \frac{r_p v_\infty^2}{\mu_J} = 9.745$$
$$\delta = 2\arcsin\frac{1}{e} = 11.779°$$
$$h = r_p\sqrt{v_\infty^2 + \frac{2\mu_J}{r_p}} = 115475008.6[km^2/s]$$
$$v_p = \frac{h}{r_p} = 11.789[km/s]$$

The values for eccentricity e, V_p and turn angle δ are compared with the reported values in Table 6 [2, 21, 24]. The calculated values are in line with those provided by the Mission Plan and the approximation of periapsis time as time of departure from SOI holds as an appropriate approximation.

Table 6: Comparison Jupiter Flyby

	Reported	Calculated	% Difference
e	9.405	9.745	3.62%
δ	12.2°	11.779°	3.45%
v_p	11.6[km/s]	11.789[km/s]	1.62%

The Δv was calculated by assuming a leading side fly by, this is due to the fact that Jupiter was used to push Cassini further out from the sun. The flight path angle upon encountering Jupiter's SOI was calculated to be 34.312°. The magnitude of the hyperbolic excess speed when departing Jupiter's SOI is the same as that of arrival but bent through the turn angle so the hyperbolic excess speed at departure was calculated to be:

$$\vec{V}_{\infty)Departure} = [7.3192, 7.7160, -0.3425][\text{km/s}] \quad V_{\infty)Departure} = 10.6407[\text{km/s}]$$

The magnitude of the hyperbolic excess speed for arrival and departure differ slightly due to rounding error however it is only in the hundredths place and for calculation purposes small enough to be neglected. The calculated delta-V is compared to the reported value in Table 7 [2]:

Table 7: Delta-V's for Jupiter Flyby

Reported ΔV [km/s]	Calculated ΔV [km/s]	% Difference
2.2	2.1820	0.81%

The percent difference is small for the calculation above, however, for the sake of interest the delta-V was calculated again by solving Lambert's problem twice, firstly from Earth to Jupiter and the then from Jupiter to Saturn. The delta-V calculated from this method was 1.9194[km/s], yielding a percent difference of 12.75%. The larger error is possibly due to the size of Jupiter, Saturn and the respective SOI's for the second half of the fly by.

III.E. Phoebe Flyby

The Cassini team had a science-acquisition opportunity to fly by one of Saturn's farthest moons, Phoebe. Phoebe is small, roughly 8.292×10^{18} kg, just 0.0113% the mass of Earth's moon, and orbits retrograde, in the opposite direction of Saturn's rings and most other satellites [26], [27]. Direct observations could yield key data to support the prevailing theory that Phoebe's irregular shape and orbit distinguish it as a captured asteroid from elsewhere in the solar system. After the orbiter was captured deep inside Saturn's gravity well, its highest planned apoapses failed to achieve the distance of the orbit of Phoebe at 12952000 km [2]. Not only did any close observations of Phoebe need to take place on the inbound flight, but the team also needed to verify Phoebe lacks the mass to significantly affect Cassini's trajectory on this inbound portion of the flight. This low-mass hypothesis is supported here by first confirming the spacecraft enters Phoebe's sphere of influence using data for the distance from Phoebe to Saturn and their respective masses [2], [28]:

$$SOI_{ph} = r_{phoebe\ to\ saturn}(\frac{M_{phoebe}}{M_{saturn}})^{(\frac{2}{5})} \tag{1}$$

$$SOI_{ph} = 12952000(\frac{8.292 \times 10^{18}}{5.6832 \times 10^{26}})^{\frac{2}{5}} = 9505.25\,\text{km} \tag{2}$$

Since the spacecraft's known orbit has a Phoebe flyby periapsis of 2070.9 km, Cassini does enter the small moon's SOI with a published flyby velocity of $v_\infty = 6.35\ \frac{km}{s}$ [2]. However, as shown in Eqs. 3 and 4 for the hyperbolic flyby orbit, the turning angle of $\delta = 1.1496 \times 10^{-5}$ rad proves negligible.

$$e = 1 + \frac{r_p v_\infty^2}{\mu_{phoebe}} = 1 + \frac{2070.9 \times 6.35^2}{0.48} = 173967.386 \tag{3}$$

$$\delta = 2sin^{-1}(\frac{1}{e}) = 2sin^{-1}(\frac{1}{173967.386}) = 1.1496 \times 10^{-5}\ \text{rad} \tag{4}$$

The tiny change in angle creates a commensurately tiny change in velocity given by:

$$\Delta v = \sqrt{v_\infty^2 + v_\infty^2 - 2v_\infty^2 cos(\delta)} = \sqrt{2 \times 6.35^2 - 2 \times 6.35^2 cos(1.1496 \times 10^{-5})} = 0.00418\text{km/s} \tag{5}$$

The calculation of the 4.18 $\frac{m}{s}$ Δv from the Phoebe flyby is neglected (left untabulated) in the mission reports as it is generally considered insignificant for the purposes of this report [2].

III.F. Entry Into Saturn Sphere of Influence (SOI)

The Cassini probe entered Saturn's sphere of influence in mid-June of 2004, from the leading side moving retrograde relative to the planet, approaching the morning side of the planet at a phase angle of 75° (angle between the planet and the Sun and the planet and the craft) [29]. At the initial periapsis on the trailing side of the planet, the craft had a phase angle of 94°. The true anomaly at infinity, θ_∞, is considered to be the sum of these two phase angles, equal to -169° or -2.9496 rad. In addition to this θ_∞, defining a periapsis at $r_p = 1.3R_{sat} = 78348.4$ km (where $R_{sat} = 60268$ km, the radius of Saturn) is sufficient to work backwards and determine the craft's relative approach velocity into Saturn V_∞ as follows:

$$v_\infty = \frac{\mu}{h} \times e \times sin(\theta_\infty) = \frac{\mu}{r_p\sqrt{v_\infty^2 + \frac{2\mu}{r_p}}} \times (1 + \frac{r_p v_\infty^2}{\mu}) \times sin(\theta_\infty) \tag{6}$$

The equations for eccentricity $e = (1+\frac{r_p v_\infty^2}{\mu})$ and angular momentum $h = r_p\sqrt{v_\infty^2 + \frac{2\mu}{r_p}}$ for a hyperbola are substituted into Eq. 6. μ is that of Saturn and will be so for the rest of the analysis unless specified. To ensure a positive magnitude v_∞, θ_∞ must be the negative of the previously stated $\theta_\infty = -169°$. Solving for v_∞ yields:

$$v_\infty = 3.01021\,\text{km/s} \tag{7}$$

One must obtain v_∞ to obtain the angular momentum of the incoming hyperbola, which, combined with the periapsis distance yields the periapsis velocity for orbital segment 0 (the incoming hyperbola in Fig. 6):

$$h = r_p\sqrt{v_\infty^2 + \frac{2\mu}{r_p}} = 78348.4\sqrt{3.01021^2 + \frac{2 \times 37391000}{78348.4}} = 2449353\,km^2/s \tag{8}$$

$$v_{p0} = \frac{h}{r_p} = \frac{2449353}{78348.4} = 31.26233\,\text{km/s} \tag{9}$$

For an ellipse, angular momentum is related to semimajor axis $a = \frac{r_p + r_a}{2}$ and eccentricity $e = \frac{r_a - r_p}{r_a + r_p}$ by: $h = \sqrt{\mu a(1-e^2)}$. To find the final velocity at periapsis of orbital segment 1 (the ellipse created after the injection burn), one prescribes a post-burn orbit (orbital segment 1 in Fig. 6) with the required radii of apoapsis and periapsis stated in the mission plans of $r_a = 9.0884144 \times 10^6$ km and $r_p = 78348.4$ km, respectively. If one assumes an instantaneous impulsive maneuver:

$$v_{p1} = \frac{h}{r_p} = \frac{\sqrt{\mu a(1-e^2)}}{r_p} = 30.9838\,\text{km/s} \tag{10}$$

$$\Delta v = |v_{p1} - v_{p0}| = 278.5\text{m/s} \tag{11}$$

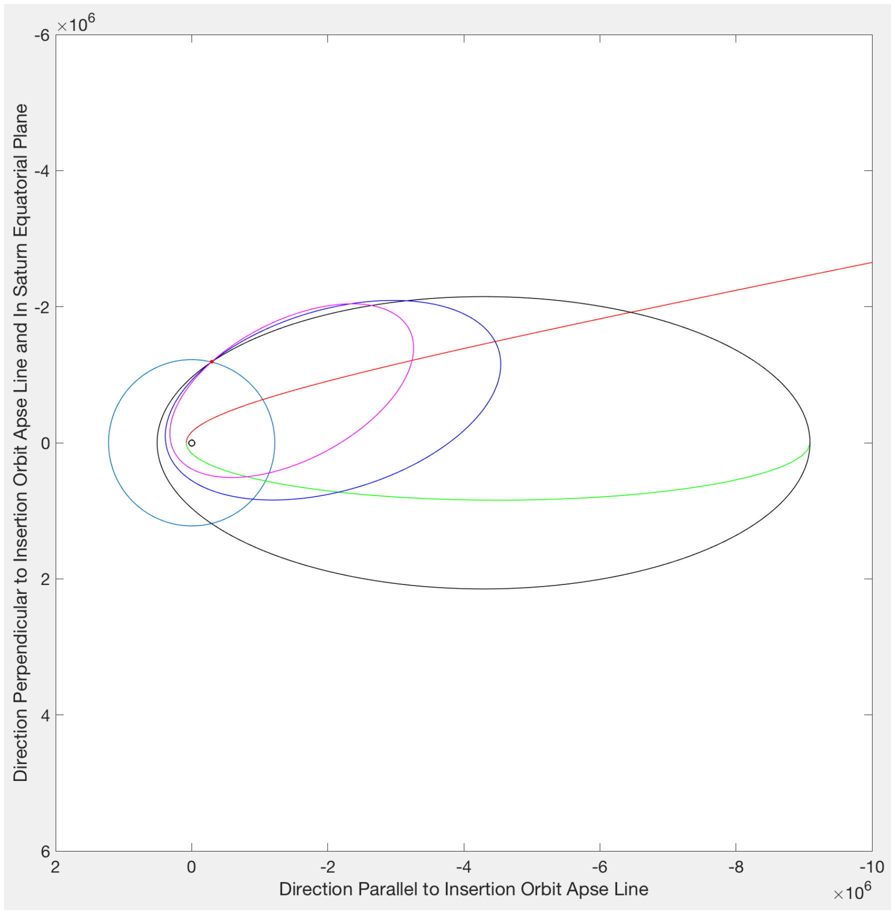


Fig. 6: Titan's orbit: light blue circle, Orbit Segment 0: red, Orbit Segment 1: green, Orbit Segment 2: black, Orbit Segment 3: dark blue, Orbit Segment 4: magenta

The actual Δv of the first burn was reported as 626 m/s. The egregious 55.5% error is most likely due to the non-instantaneous nature of real spacecraft maneuver burns [2]. The Cassini injection burn took 1 hour, 36 minutes and 20 seconds, spreading out the Δv to less efficient locations in the orbit in a manner beyond the scope of this report [2]. An additional major source of error is how the spacecraft initiated its injection burn such that the burn finished at periapsis, sacrificing the full benefits of the Oberth effect (which governs the high efficiency of burns at periapsis) in exchange for the opportunity to conduct science operations at the closest approach (1.3 Saturn radii) of the entire mission until crashing into the planet during the grand finale [2]. Additional error could be due to the gravitational perturbations on Cassini from Saturn's rings and other larger moons such as Enceladus.

Applying a similar technique detailed in the Matlab code in the Appendix, fixing the period of orbital segment 2 in Fig. 6 (the ellipse created after the OTM-002 burn at apoapsis to raise the periapsis) at 124.1 days as called for in the mission plan, one obtains [2]:

$$\Delta v = |v_{a2} - v_{a1}| = |0.6646 - 0.2671|\frac{km}{s} = 397.5\,\mathrm{m/s} \tag{12}$$

The actual burn resulted in a Δv of 393 m/s, yielding a 1.15% error between the above calculation and observed reality [2]. Burning at apoapsis, the spacecraft has a smaller velocity, moving through a smaller distance than at periapsis in the same amount of time by Kepler's Second Law. Thus, the actual 51 minute burn at apoapsis covered a comparatively small distance and more closely approximated the instantaneous impulsive maneuver analyzed in this report [2].

III.G. Achieving Resonance With Titan in the First Saturnian Orbits

The following section reconstructs Cassini's orbital trajectories incorporating encounters of Titan, Saturn's largest moon, to provide Δv relative to Saturn instead of using chemical thrusters with valuable fuel. The following calculations are made assuming Cassini's orbits to be co-planar with the Saturn-Titan system. This approximation must be made due to limitations in the knowledge used to generate this report. One could assume an impulsive maneuver executed at the descending node between maneuver burns 1 and 2 and using the law of pure plane change maneuvers $\Delta v = 2vsin(\frac{i}{2})$ (i is the angle of inclination change, 17.2° according to the mission plan) [2]. The required Δv to make this simplification a reality would be on the order of 6.5 km/s, over 10 times the published injection burn (the largest burn of the mission). Although not wholly reflective of reality, assuming in-plane orbits allows for the calculation of Titan flyby Δv values using the knowledge of AA310 to verify the feasibility of the rest of the mission. In reality, Cassini's inclined trajectories allowed it to remain clear of the inner and outer portions of Saturn's rings, complicating the mathematics (beyond those of this report) to avoid the potential for spacecraft damage.

If one assumes Titan to be in a constant, circular orbit of Saturn at the semimajor axis distance of 1221900 km (a reasonable simplification given Titan's low orbital eccentricity of 0.0292), one finds Titan's orbital velocity [2]:

$$\vec{v}_{titan} = \sqrt{\frac{\mu_{titan}}{r_{titan}}}\hat{u}_V = \sqrt{\frac{37931000}{1221900}}\hat{u}_V = 5.5716\hat{u}_V \text{ km/s} \tag{13}$$

By setting Titan's radius equal to that of Cassini on orbit segment 2, one can find the true anomaly at the intersection where Cassini encounters Titan on the inbound portion of the spacecraft's trajectory:

$$\theta = 2\pi - cos^{-1}(\frac{1}{e_2}(\frac{a_2(1-e_2^2)}{r_{titan}} - 1)) = 4.47205 \, \text{rad} \tag{14}$$

The corresponding perpendicular and radial components of velocity yield the inbound velocity components, which can be used to enter the Titan frame of reference by simply subtracting Titan's velocity from that of the spacecraft:

$$v_{perp} = \frac{h_2}{r_{titan}} = 4.9432\hat{u}_V \text{ km/s} \tag{15}$$

$$v_r = \frac{\mu}{h_2}e_2 sin(\theta) = 5.4538\hat{u}_S \text{ km/s} \tag{16}$$

$$\vec{v}_{\infty_{in}} = 5.4538\hat{u}_S + 4.9432\hat{u}_V - 5.5716\hat{u}_V = -0.6283\hat{u}_V + 5.4538\hat{u}_S \text{ km/s} \tag{17}$$

The details for the above calculations are included in the commented Matlab code in the Appendix. The key requirement of these calculations is that the flyby of Titan be prepared (with negligible correction maneuvers) such that the hyperbolic flyby periapsis altitude yields the proper Δv to put Cassini on an orbit with a period that is a simple multiple of that of Titan (which is roughly 16 days) [2]. The mission report calls for the period of Cassini's orbit after leaving the first Titan flyby to be 47.9 days, thus Titan will complete three orbits before it and Cassini return to the same location relative to Saturn in Fig. 6 [2]. Enforcing this period yields the desired semimajor axis of orbit segment 3 (the ellipse after the first Titan encounter) via Kepler's Third Law [25]. One must also enforce a second parameter: the periapsis of orbit segment 3 lies at 6.2 Saturn radii as detailed in the mission plan [2]. The Matlab code finds the true anomaly at the Titan encounter of this new ellipse, along with the required perpendicular and radial velocities that allow for the calculation of the required turning angle and thus the required periapsis radius over Titan and Δv of this first encounter.

$$\theta = 2\pi - cos^{-1}(\frac{1}{e_3}(\frac{a_3(1-e_3^2)}{r_{titan}} - 1)) = 4.1796\,\text{rad} \tag{18}$$

$$v_{perp} = \frac{\mu}{h_3}(1 + e_3 cos(\theta)) = 4.6654\hat{u}_V\ \text{km/s} \tag{19}$$

$$v_r = \frac{\mu}{h_3} e_3 sin(\theta) = 5.4389\hat{u}_S\ \text{km/s} \tag{20}$$

$$\vec{v}_{\infty_{out}} = -0.9063\hat{u}_V + 5.4389\hat{u}_S\ \text{km/s} \tag{21}$$

$$\delta = 2sin^{-1}(\frac{1}{e_3}) = 0.0504\,\text{rad} = 2sin^{-1}(\frac{1}{1+\frac{r_p v_\infty^2}{\mu_{titan}}}) \tag{22}$$

$$r_p = \frac{\mu_{titan}}{v_\infty^2}(\frac{1}{sin(\frac{\delta}{2})} - 1) = 11474\,\text{km} \tag{23}$$

$$\Delta v = 278\,\text{m/s} \tag{24}$$

The required periapsis for the first Titan flyby sits 11474 km above the center of the moon, 3 times the radius of the published flyby, which was 3749 km [2]. The much closer actual flyby was required due to the larger Δv required to make inclination changes (which would cost large amounts of fuel if made by engine burns) necessary for further science acquisition and future observation of Titan. The craft passes on the leading side of the Titan as evidenced by how the component of velocity aligned with Titan's velocity becomes more negative, indicating that the probe is turned counterclockwise as it passes around the moon (looking down at the north pole).

The difference in true anomalies relative to the same point in space (the Titan encounter) becomes important in the Matlab simulation that generated Fig. 6. This change in anomaly represents the angle between apse lines of orbit segments 2 and 3, applying a rotation matrix to the post-Titan trajectory in the Matlab code ensures that the two segments intersect at the Titan encounter in Fig. 6. Thus, Fig. 6 properly matches the shape of the actual trajectory from the mission plan [2].

Since Cassini makes one full orbit before coming back to encounter Titan again, it enters Titan's SOI with the same velocity it left with in the previous analysis:

$$\vec{v}_{\infty_{in}} = -0.9063\hat{u}_V + 5.4389\hat{u}_S\ \text{km/s} \tag{25}$$

Given the period of orbit segment 4 (the ellipse after the second Titan encounter) must be 32.1 days (two Titan orbits), and segment 4 must yield a periapsis at 4.8 Saturn radii, one can apply the same analysis as above to discover:

$$\theta = 3.9921\,\text{rad at Titan encounter 2} \tag{26}$$

$$\vec{v}_{\infty_{out}} = -1.1808\hat{u}_V + 5.3856\hat{u}_S\,\text{km/s} \tag{27}$$

$$\delta = 0.0507\,\text{rad} \tag{28}$$

$$r_p = 11348\,\text{km} \tag{29}$$

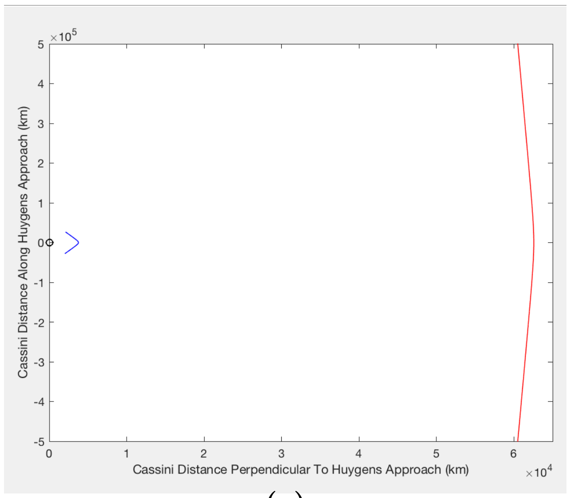
$$\Delta v = 280\,\text{m/s} \tag{30}$$

This second encounter periapsis is also 3 times higher than the published 3767 km flyby, the same explanations apply since an 8.6° inclination change was required [2]. Although the calculated periapsis altitudes were far from those of the mission plan, the feasibility of each required orbit was successfully verified as each stage from the first insertion burn to the first two Titan gravity assists yielded reasonable results that were within Titan's 43327 km SOI, but not within its 2575 km radius [30], [2]. Additionally the calculated encounters saved 278 and 280 m/s of Δv that would otherwise have required fuel, successfully demonstrating the practical importance of these flybys to extending the mission.

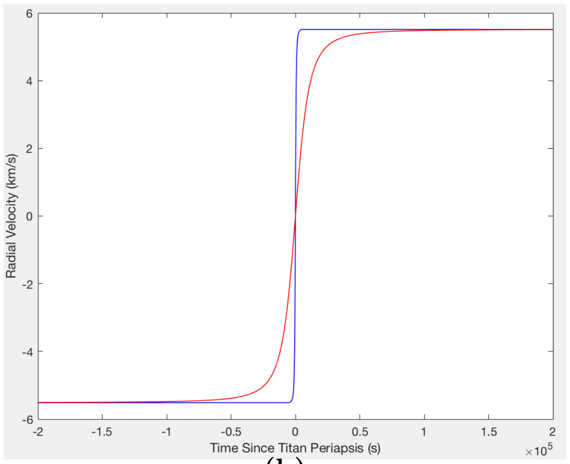
III.H. Huygens Separation and Emergency Maneuver Change

With Cassini on orbit segment 4, on a collision course with Titan, the Huygens probe (lacking its own navigational equipment) separated from the rest of the spacecraft, ensuring the probe would hit its intended target by tracing out the pink ellipse. The change in mass and impulse imparted to the spacecraft during the Huygens separation process is negligible, especially compared to the 24 m/s getaway burn executed two days later to put Cassini on a slightly different trajectory than the lander, arriving at Titan roughly 3775 km or 62575 km above the center of the moon [2].

The difference between these two periapses proved critical to the mission. During the Earth flyby, an insistent engineer and the rest of the Cassini team performed a test, sending packets of simulated data to be relayed between Cassini and Huygens [31]. The data sent back to Earth was garbled beyond reconstruction. The data simulated a Doppler-shifted frequency similar to that expected as the orbiter passed close to Huygens' landing zone on Titan (the blue trajectory in Fig. 6). The Doppler shift pushed the frequency past the tolerance of Huygens' pre-programmed components, which could not be changed in flight. Thus, the Doppler shift needed to be reduced by lowering the relative velocity between the probe and orbiter during the landing [32]. In this analysis, the relevant component of velocity is approximated as the radial component of Cassini's velocity with respect to Titan during the 2.5 hours prior to periapsis. The Matlab code in the Appendix applies the equations for hyperbolic orbits using the original periapsis of 3775 km and the modified periapsis of 62575 km as well as an inbound velocity relative to Titan equal to the outbound velocity after the second Titan encounter. Applying the reverse of Algorithm 3.2 to find the times relative to periapsis of each hyperbolic orbit, one can find the radial velocities at each of these times and obtain the plot in Fig. 7b [25].



(a)



(b)

Fig. 7: Left: Cassini Trajectories During Huygens Landing (blue: original, red: corrected, Titan at the origin); Right: Cassini Radial Velocity over Time During Huygens Landing (blue: original, red: corrected)

As evidenced in Fig. 7b, the modified, higher periapsis trajectory has a lower magnitude radial component during the flyby (expressed by the distance between the modified trajectory's red line and the blue line of the original, planned orbital path). The average difference in radial velocity during the 2.5 hours before periapsis (approximately when Huygens was stationary relative to Titan, descending through the atmosphere) was 3.23 km/s [2]. This significant lowering of the radial velocity means the Doppler shift will be reduced during the critical time period of the landing. When this method was put to use during the mission, Huygens' data was successfully saved.

III.I. Grand Finale

As Cassini ran out of fuel, mission planners decided to crash Cassini into Saturn in order to prevent potential contamination of the outer moons if the probe crashed landed on any of them after NASA lost contact with the probe. On its final orbits of Saturn, Cassini would fly between the innermost rings (the D Rings) and on the last 5 orbits, Cassini would sample the uppermost regions of Saturn's atmosphere, before plunging into Saturn's atmosphere, burning up in the process.

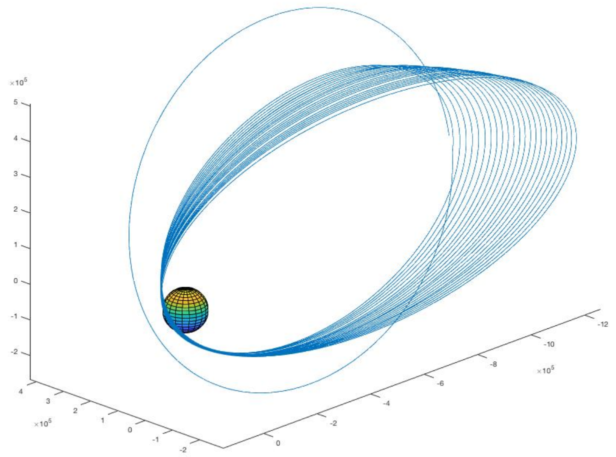


Fig. 8: Cassini's last 23 orbits. Orbit with the larger r_p is the last of the standard orbits. All others are the Grand Finale Orbits. Data from NASA's SPICE Geometry Calculator.

Hard data on this section of the mission is difficult to obtain. All of the data used here is from WebGeoCalc [33]. The data collected was the **r** and **v** for Cassini and Titan with Respect to Saturn from April 15th to September 22nd of 2017. Due to size limitations from the WebGeoCalc service, data was only given in 10 minute intervals. To increase accuracy, MATLAB's interp1 function was used in spline mode to increase the interval rate to 1 minute between data points.

Table 8: Orbital Parameters of Original Orbits

Parameter	Value	Units
r_a	1286900	km
v_a	2.55	$\frac{km}{s}$
h	3165400	$\frac{km^2}{s}$
i	58.05	$^\circ$
e	0.795	

Note: No orbital parameter information was available, so no comparison to the actual values could be done.

In order to place Cassini in the Grand Finale orbits, a maneuver had to be done to lower the r_p from $r_p = 147320km$ to $r_p = 63177km$. For comparison sake, a Hohmann transfer into the new orbit has a $\Delta v = 0.996\frac{km}{s}$. Assuming an ISP of 310$s$ for Cassini's R-4D Monomethylhydrazine [25] main engine, and assuming that Cassini had $\frac{1}{4}$ of its original propellant load (783kg) remaining (a gross overestimation), this maneuver would require an extra 221kg of propellant to complete the maneuver.

Because using the main engines would be impossible, Cassini used a gravity assist with Titan to accomplish the same Δv maneuver. By comparing the $r_{Titan} - r_{Cassini}$ with $r_{SOITitan} = 43321km$, Cassini entered Titan's sphere of influence at 3:57 UTC on April 22, 2017.

Table 9: Orbital Parameters of Titan Flyby

Parameter	Value	Units
r_a	43500	km
v_a	5.4336	$\frac{km}{s}$
h	17475	$\frac{km^2}{s}$
i	78.1	$^\circ$
e	10.55	
δ	9.557	$^\circ$

For this orbital maneuver, $V_{\infty Arrival} = \langle 3.92, 2.41, -2.89\rangle \frac{km}{s}$ and $V_{\infty Departure} = \langle 4.41, 2.05, -2.12\rangle \frac{km}{s}$. Therefore $\Delta V = 0.98\frac{km}{s}$. This agrees quite nicely with the Hohmann transfer $\Delta V = 0.996\frac{km}{s}$. Cassini reached an altitude of $979km$ and a top speed of $5.8\frac{km}{s}$ [34] during this flyby.

Table 10: Calculated vs. Actual Values of Titan Flyby

Parameter	Actual	Calculated	% Difference
v_p	$5.8\frac{km}{s}$	$5.86\frac{km}{s}$	1.1%
Altitude	$979km$	$819.2km$	17.8%

There is a significant error in the altitude. The biggest possible sources of error are the interpolation function applied to the data, and a having a sample rate of only once a minute. It should also be noted that "Actual" data cited here came from a NASA press release, which will round off values for brevity.

Table 11: Orbital Parameters of Grand Finale Orbit 1

Parameter	Value	Units
r_a	108480	km
v_a	3.62	$\frac{km}{s}$
h	213250	$\frac{km^2}{s}$
i	56.86	$^\circ$
e	0.905	

Even though Cassini dipped into Saturn's atmosphere during the final 5 orbits of the Grand Finale, there wasn't much of a ΔV imparted on the spacecraft. In order to crash it into Saturn, Cassini had one more encounter with Titan. However the closest approach was 119049 km which is 2.75 times greater than Titans sphere of influence [35], therefore the method of patched conics described in Curtis will not be useful with this encounter. The little data about the last orbit indicated that assuming Saturn was just a point mass, the r_p would have been about $3240km$ lower than the previous orbit.

IV. Conclusions

Cassini-Huygens was a scientific marvel that gave a better understanding of the magnetosphere and atmosphere of Saturn and its moons. This report provided valuable insight into the orbital mathematics and procedures used to place Cassini in orbit around Saturn with a high degree of precision.

The VVEJGA executed by Cassini is considered to be the most complex orbital trajectory to date. Analyzing this trajectory with knowledge gained in AA310 proved challenging and enriching. The most difficult part of the analysis came when attempting to match the characteristic energy of Cassini when it left Earth's sphere of influence, which was reported by JPL to be 16.6 $\frac{km^2}{s^2}$. Using many different algorithms, including ones provided by Curtis' "Orbital Mechanics" textbook [25], the analysis determined that Cassini should have had a characteristic energy of 7.747 $\frac{km^2}{s^2}$. Other than this anomaly, the calculations for all three of the inner solar system flybys came within 3.207% or better of their respective reported values. An attempt was made to analyze the deep space maneuver between Venus flybys one and two using a bi-elliptic Hohmann transfer. This provided inconclusive results and it was determined that the trajectory could not be evaluated with purely 310 knowledge and the reported data. NASA provided extensive documentation examining the potential Earth impact Cassini faced on its Earth flyby. Using Monte Carlo simulations, it was determined that the spacecraft would have less than a one in one million chance to collide with earth on both of its proposed trajectories. This was deemed an acceptable probability and Cassini flew past Earth and onto Jupiter with no irregularities. The Jupiter flyby imparted the final delta-V needed to reach its destination in Saturn's sphere of influence. The flyby was analyzed using the trajectory and flybys analysis based on the patch conic approximation covered in the 310 class. The calculated for v_p was 11.789[km/s] and had a 1.62% error in addition, the delta v calculated was within 0.81% of the reported value when the turn angle was used in the calculations. These results were within acceptable error. The Jupiter flyby also served as a test for some of the systems that would be implemented in the exploration of Saturn and Titan.

While the Phoebe flyby upon arrival at Saturn proved insignificant to the orbital trajectories, considering this unique opportunity to study one of Saturn's distant, most curious moons demonstrated the criticality of every portion of the meticulously planned mission to the scientific goals beyond simply putting the craft in the proper spot at the proper time. The calculated orbital capture burns at Saturn between 278.5 and 397.5 m/s yielded invaluable experience with the high changes in orbital speed required to stay at one's destination, performing years of science instead of being a guest for the few hours of a flyby. Despite being limited to planar flybys, the methods of AA310 proved highly effective at matching the apoapses and periapses of Cassini's first orbits around Saturn. However the extremely long capture burn was beyond the scope of AA310. The resonant orbits with Titan were also successfully modelled, returning reasonable Titan flyby periapses limited by not considering the real inclination changes during the mission. Perhaps most compelling was the extension of relatively simple ideas concerning hyperbolic flybys such as those at Titan or the larger planets to empirically verify the real solution to the mission's greatest problem: correcting the Doppler shift between Cassini and Huygens. The model of the corrected trajectory saved an average of 3.23 km/s of radial velocity during the 2.5 hours prior to Cassini-Titan periapsis (roughly the time during which Huygens descended through the moon's atmosphere). Proving able to save real mission data was an excellent validation of the knowledge gained over the course of AA 310.

V. References

[1] Agency, E. E. S., “Cassini-Huygens mission home,” http://sci.esa.int/cassini-huygens/, 2016.

[2] “Cassini Mission Plan Rev O, chg 1,” PDF, Aug. 2005.

[3] Goodson, T., Grey, D., Hahn, Y., and Peralta, F., *Cassini Maneuver Experience: Finishing Inner Cruise*, Jet Propulsion Laboratory, 1997.

[4] Satiales, C. C. N. D., “Cassini-Huygens: Two spacecraft to probe the secrets of Saturn,” https://cassini-huygens.cnes.fr/en/CASSINI/GP_satellite.htm, March 2012.

[5] Zaitsev, Y., “Russia to Develop Nuclear Powered Spacecraft for Mars,” `https://sputniknews.com/analysis/2009111156797969/`, Nov. 2009.

[6] Wire, G. N., “Aerojet Rocketdyne Propulsion Guides Cassini on its Grand Finale at Saturn,” https://globenewswire.com/news-release/2017/09/15/1123350/0/en/Aerojet-Rocketdyne-Propulsion-Guides-Cassini-on-its-Grand-Finale-at-Saturn.html, 2017.

[7] “Engine,” https://solarsystem.nasa.gov/missions/cassini/engine/, 2018.

[8] Thompson, J., “Solar System Explanation,” `https://solarsystem.nasa.gov/missions/cassini/mission/spacecraft/cassini-orbiter/`, Sept. 2018.

[9] Dougherty, M., “The Cassini Magnetic Field Investigation,” `http://lasp.colorado.edu/~horanyi/graduate_seminar/Magnetometer.pdf`, Dec. 2002.

[10] Altobelli, N., “Flux and composition of interstellar dust at Saturn from Cassinis Cosmic Dust Analyzer,” `http://science.sciencemag.org/content/352/6283/312`, April 2016.

[11] “Reaction/Momentum Wheel,” https://spinoff.nasa.gov/spinoff1997/t3.html.

[12] NASA, “Solar System Exploration Cassini,” https://solarsystem.nasa.gov/missions/cassini/the-journey/the-spacecraft/, March 2012.

[13] ESA, “The Cassini-Huygens Voyage Through the Solar System,” .

[14] Steven Flanagan, F. P., *Cassini 1997 VVEJGA Trajectory Launch/Arrival Space Analysis*, Jet Propulsion Laboratory, 2005.

[15] Dizikes, P., “Explained: Monte Carlo Simulations,” `http://news.mit.edu/2010/exp-monte-carlo-0517`, May 2010.

[16] NASA, “Cassini Program Environmentl Impact Statement Supporting Study,” `https://solarsystem.nasa.gov/resources/17807/cassini-environmental-impact-statement-supporting-studies/`, Nov. 1993.

[17] Roth, D., Guman, M., Ionasescu, R., and Taylor, A., *Cassini Earth Swingby Plan Supplement*, Jet Propulsion Laboratory, 1997.

[18] Goodson, T., Grey, D., Hahn, Y., and Peralta, F., *Cassini Maneuver Experience: Finishing Inner Cruise*, Jet Propulsion Laboratory, 1997.

[19] Bellarose, J., Roth, D., and Criddle, K., "The Cassini Mission: Reconstructing Thirteen Years of the Most Complex Gravity-Assist Trajectory Flown to Date," .

[20] NASA, "Cassini: Mission Timeline," `https://solarsystem.nasa.gov/missions/cassini/the-journey/timeline/#earth-moon-flyby`, 2018.

[21] NASA, "HORIZONS Web-Interface," https://ssd.jpl.nasa.gov/horizons.cgi, Dec. 2018.

[22] JPL/NASA, "Cassini celebrates 10 years since Jupiter encounter," `https://phys.org/news/2010-12-cassini-celebrates-years-jupiter-encounter.html`, Dec. 2010.

[23] NASA, "Cassini Completes Earth Flyby," `https://solarsystem.nasa.gov/news/12192/cassini-completes-earth-flyby/`, Aug. 1999.

[24] D.C. Roth, M.D. Guman, R. I., "Cassini Orbit Reconstruction From Earth To Jupiter," PDF, 2005.

[25] Curtis, H., *Orbital Mechanics: For Engineering Students*, Butterworth-Heinemann, 2015.

[26] "Earth's Moon - By the Numbers," https://solarsystem.nasa.gov/moons/earths-moon/by-the-numbers/, 2018.

[27] Antreasian, P., Bordi, J., Criddle, K., Ionasescu, R., Jacobson, R., Jones, J., MacKenzie, R., Meek, M., Pelletier, F., Roth, D., Roundhill, I., and Stauch, J., "Cassini Orbit Determination Performance During the First Eight Orbits of the Saturn Satellite Tour," Print, 2018.

[28] NASA, "Saturn - By the Numbers," https://solarsystem.nasa.gov/planets/saturn/by-the-numbers/, 2018.

[29] NMSU, "PDS Stream," `https://pds-atmospheres.nmsu.edu/data_and_services/atmospheres_data/Cassini/mission-cat_3-26-13`.

[30] NASA, "Titan - By the Numbers," https://solarsystem.nasa.gov/moons/saturn-moons/titan/by-the-numbers/, 2018.

[31] Atkinson, N., "The Incredible Story of How the Huygens Mission to Titan Succeeded When It Could Have Failed," https://www.universetoday.com/132839/incredible-story-huygens-mission-titan-succeeded-failed/, 2017.

[32] ESA, "ESA and NASA Agree a New Mission Scenario for Cassini-Huygens," http://sci.esa.int/cassini-huygens/27650-esa-and-nasa-agree-a-new-mission-scenario-for-cassini-huygens/, 2018.

[33] NASA, "WebGeocalc - A GUI Interface to SPICE Version 1.10," `https://wgc.jpl.nasa.gov:8443/webgeocalc/#OrbitalElements`, Oct. 2018.

[34] NASA, "Titan Flyby T-126: Final Close Encounter, Gateway to the Grand Finale," https://solarsystem.nasa.gov/news/13019/titan-flyby-t-126-final-close-encounter-gateway-to-the-grand-finale/, April 2017.

[35] NASA, "Cassini Makes its 'Goodbye Kiss' Flyby of Titan," https://solarsystem.nasa.gov/news/13116/cassini-makes-its-goodbye-kiss-flyby-of-titan/, April 2017.

VI. Appendices

Venus Venus Earth Gravity Assist and Flyby Analysis

```
%% FIRST HYPERBOLIC FLYBY OF VENUS
clear all

% Define the Orbit: Values obtained from Backup_Cassini_Phase_1.m
mu = 324900;
e = 1.70909;
delta = 71.6212; % Degrees
h = 74678.2;
Vp = 11.7863;

% Plot the orbit
figure(2)
theta = linspace(0,2*pi,100);
r = (h^2/mu)*(1./(1+e.*cos(theta)));
[theta,r] = pol2cart(theta,r);
plot(0,0,'ko','MarkerSize',10,'MarkerFaceColor','k'), hold on, grid on
plot(theta,r,'k')
set(gca, 'fontsize', 14);xlim([-2e+04 0.2e+05])
% title('Venus Flyby 1');legend('Venus','Flyby Trajectory','Location','Best')
xlabel('Distance (km)');ylabel('Distance (km)')
% print -djpeg 'Venus_flyby_1_trajectory.jpg'

%% VENUS FLYBY 2
clear all

% Define the Orbit: Values obtained from Backup_Cassini_Phase_1.m
mu = 324900;
e = 2.72681;
delta = 43.028; % Degrees
h = 87589.3;
Vp = 13.8241;

% Plot the orbit
theta = linspace(0,2*pi,100);
r = (h^2/mu)*(1./(1+e.*cos(theta)));
[theta,r] = pol2cart(theta,r);
plot(-319,0,'mo','MarkerSize',10,'MarkerFaceColor','m'), hold on, grid on
plot(theta,r,'m')
set(gca, 'fontsize', 14);xlim([-2e+04 0.2e+05])
title('Venus Flyby 1 & 2');
legend('Venus at 1st Flyby','Flyby 1 Trajectory','Venus at 2nd Flyby','Flyby
2 Trajectory','Location','Best')
xlabel('Distance (km)');ylabel('Distance (km)')
print -djpeg 'Both_Venus_flyby_trajectorys.jpg'

clear all; close all; clc

%% Compare to algorithm 8.2 - Example 8.8 .m file
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Example_8_08
% ~~~~~~~~~~~~
%{
  This program uses Algorithm 8.2 to solve Example 8.8.
```

```
mu           - gravitational parameter of the sun (km^3/s^2)
deg          - conversion factor between degrees and radians
pi           - 3.1415926...

planet_id    - planet identifier:
                1 = Mercury
                2 = Venus
                3 = Earth
                4 = Mars
                5 = Jupiter
                6 = Saturn
                7 = Uranus
                8 = Neptune
                9 = Pluto

year         - range: 1901 - 2099
month        - range: 1 - 12
day          - range: 1 - 31
hour         - range: 0 - 23
minute       - range: 0 - 60
second       - range: 0 - 60

depart       - [planet_id, year, month, day, hour, minute, second]
               at departure
arrive       - [planet_id, year, month, day, hour, minute, second]
               at arrival

planet1      - [Rp1, Vp1, jd1]
planet2      - [Rp2, Vp2, jd2]
trajectory   - [V1, V2]

coe          - orbital elements [h e RA incl w TA]
               where
                 h    = angular momentum (km^2/s)
                 e    = eccentricity
                 RA   = right ascension of the ascending
                        node (rad)
                 incl = inclination of the orbit (rad)
                 w    = argument of perigee (rad)
                 TA   = true anomaly (rad)
                 a    = semimajor axis (km)

jd1, jd2     - Julian day numbers at departure and arrival
tof          - time of flight from planet 1 to planet 2 (days)

Rp1, Vp1     - state vector of planet 1 at departure (km, km/s)
Rp2, Vp2     - state vector of planet 2 at arrival (km, km/s)
R1, V1       - heliocentric state vector of spacecraft at
               departure (km, km/s)
R2, V2       - heliocentric state vector of spacecraft at
               arrival (km, km/s)

vinf1, vinf2 - hyperbolic excess velocities at departure
               and arrival (km/s)

User M-functions required: interplanetary, coe_from_sv,
                                month_planet_names
%}
% ---------------------------------------------

%clear all;
global mu
mu  = 1.327124e11;
deg = pi/180;

%...Data declaration for Example 8.8:

%...Departure
planet_id = 3;
year      = 1997;
month     = 10;
day       = 15;
hour      = 8;
minute    = 27;
second    = 0;
depart = [planet_id  year  month  day  hour  minute  second];

%...Arrival
planet_id = 2;
year      = 1998;
month     = 4;
day       = 26;
hour      = 13;
minute    = 45;
second    = 0;
arrive = [planet_id  year  month  day  hour  minute  second];

%...

%...Algorithm 8.2:
[planet1, planet2, trajectory] = interplanetary(depart, arrive);

R1  = planet1(1,1:3);
Vp1 = planet1(1,4:6);
jd1 = planet1(1,7);

R2  = planet2(1,1:3);
Vp2 = planet2(1,4:6);
jd2 = planet2(1,7);

V1  = trajectory(1,1:3);
V2  = trajectory(1,4:6);

tof = jd2 - jd1;

%...Use Algorithm 4.2 to find the orbital elements of the
%   spacecraft trajectory based on [Rp1, V1]...
coe  = coe_from_sv(R1, V1, mu);
%   ... and [R2, V2]
coe2 = coe_from_sv(R2, V2, mu);

%...Equations 8.94 and 8.95:
```

```
vinf1 = V1 - Vp1;
vinf2 = V2 - Vp2;

%...Echo the input data and output the solution to
%   the command window:
fprintf('-----------------------------------------------------')
fprintf('\n Example 8.8')
fprintf('\n\n Departure:\n');
%fprintf('\n   Planet: %s', planet_name(depart(1)))
fprintf('\n   Year  : %g', depart(2))
%fprintf('\n   Month : %s', month_name(depart(3)))
fprintf('\n   Day   : %g', depart(4))
fprintf('\n   Hour  : %g', depart(5))
fprintf('\n   Minute: %g', depart(6))
fprintf('\n   Second: %g', depart(7))
fprintf('\n\n   Julian day: %11.3f\n', jd1)
fprintf('\n   Planet position vector (km)    = [%g  %g  %g]', ...
                                               R1(1),R1(2), R1(3))

fprintf('\n   Magnitude                      = %g\n', norm(R1))

fprintf('\n   Planet velocity (km/s)         = [%g  %g  %g]', ...
                                 Vp1(1), Vp1(2), Vp1(3))

fprintf('\n   Magnitude                      = %g\n', norm(Vp1))

fprintf('\n   Spacecraft velocity (km/s)     = [%g  %g  %g]', ...
                                               V1(1), V1(2), V1(3))

fprintf('\n   Magnitude                      = %g\n', norm(V1))

fprintf('\n   v-infinity at departure (km/s) = [%g  %g  %g]', ...
                                       vinf1(1), vinf1(2), vinf1(3))

fprintf('\n   Magnitude                      = %g\n', norm(vinf1))

fprintf('\n\n Time of flight = %g days\n', tof)

fprintf('\n\n Arrival:\n');
%fprintf('\n   Planet: %s', planet_name(arrive(1)))
fprintf('\n   Year  : %g', arrive(2))
%fprintf('\n   Month : %s', month_name(arrive(3)))
fprintf('\n   Day   : %g', arrive(4))
fprintf('\n   Hour  : %g', arrive(5))
fprintf('\n   Minute: %g', arrive(6))
fprintf('\n   Second: %g', arrive(7))
fprintf('\n\n   Julian day: %11.3f\n', jd2)
fprintf('\n   Planet position vector (km)   = [%g  %g  %g]', ...
                                              R2(1), R2(2), R2(3))

fprintf('\n   Magnitude                     = %g\n', norm(R1))

fprintf('\n   Planet velocity (km/s)        = [%g  %g  %g]', ...
                                  Vp2(1), Vp2(2), Vp2(3))

fprintf('\n   Magnitude                     = %g\n', norm(Vp2))
fprintf('\n   Spacecraft Velocity (km/s)      = [%g  %g  %g]', ...
                                                    V2(1), V2(2), V2(3))

fprintf('\n   Magnitude                       = %g\n', norm(V2))

fprintf('\n   v-infinity at arrival (km/s)  = [%g  %g  %g]', ...
                                           vinf2(1), vinf2(2), vinf2(3))

fprintf('\n   Magnitude                       = %g', norm(vinf2))

fprintf('\n\n\n Orbital elements of flight trajectory:\n')

fprintf('\n  Angular momentum (km^2/s)                     = %g',...
                                                                coe(1))
fprintf('\n  Eccentricity                                  = %g',...
                                                                coe(2))
fprintf('\n  Right ascension of the ascending node (deg) = %g',...
                                                            coe(3)/deg)
fprintf('\n  Inclination to the ecliptic (deg)            = %g',...
                                                            coe(4)/deg)
fprintf('\n  Argument of perihelion (deg)                 = %g',...
                                                            coe(5)/deg)
fprintf('\n  True anomaly at departure (deg)              = %g',...
                                                            coe(6)/deg)
fprintf('\n  True anomaly at arrival (deg)                = %g\n', ...
                                                           coe2(6)/deg)
fprintf('\n  Semimajor axis (km)                          = %g',...
                                                                coe(7))
% If the orbit is an ellipse, output the period:
if coe(2) < 1
    fprintf('\n  Period (days)                                 = %g', ...
                                        2*pi/sqrt(mu)*coe(7)^1.5/24/3600)
end
fprintf('\n-----------------------------------------------------\n')
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%
%
fprintf('\n Earth relative excess velocity (Vinf) at departure: %g (km/s)
\n\n',norm(V1)-norm(Vp1))
%
%
% Generate positions throughout the traveled transfer orbit
thetatheta = linspace(coe(6),coe2(6),100);
rr = (coe(1)^2/mu)*(1./(1+coe(2)*cos(thetatheta)));

% Plot the transfer
figure(1)
polarplot(thetatheta,rr,'k'),grid on; hold on;
title('Cassini Transfer Orbit 1: Non-Hohmann Earth to Venus Trajectory')

%% Plot Earth and Venus orbits
%
%
% Here, I use interplanetary.m function to calculate the departure Velocity
of
% Cassini. I then build the transfer orbit and attempt to plot it...
```

```
%
%
% Define departure and arrival components to pass to the .m function
depart = [3,1997,10,15,8,27,0];
arrive = [2,1998,4,26,13,45,0];

% Calculate the position and velocity vectors and magnitudes for Cassini on
% departure day and first arrival at Venus.
[planet1, planet2, trajectory] = interplanetary(depart, arrive);
R1 = [planet1(1) planet1(2) planet1(3)];
r1 = norm(R1);
R2 = [planet2(1) planet2(2) planet2(3)];
r2 = norm(R2);
VdVvec = [trajectory(1) trajectory(2) trajectory(3)];
VdV = norm(VdVvec);
VaVvec = [trajectory(4) trajectory(5) trajectory(6)];
VaV = norm(VaVvec);

% Define the transfer orbit
[Theta,r,h,Vr,e] = Six_Orb_Elem_No_DCM(R1,VdVvec,mu);
VrE = Vr;
hTran1 = h;
eTran1 = e;
thetaEarth = Theta;
rrEarth = r;
    % Calculate semi-major axis
    rp1 = (hTran1^2/mu)*(1/(1+eTran1));
    ra1 = (hTran1^2/mu)*(1/(1-eTran1));
    aTran1 = (rp1+ra1)/2;
    % Calculate period
    period1 = ((2*pi)/sqrt(mu))*aTran1^(3/2);
        % Find true anomaly for Venus at arrival
        [Theta,r,h,Vr,e] = Six_Orb_Elem_No_DCM(R2,VaVvec,mu);
        VrV = Vr;
        thetaVenus = Theta;
        rrVenus = r;

polarplot(thetaEarth,rrEarth,'bo','MarkerSize',8,'MarkerFaceColor','b')
polarplot(thetaVenus,rrVenus,'ro','MarkerSize',8,'MarkerFaceColor','r')

% Earth
hEarth = (30.4)*(1.46e+08);
eEarth = 0.0167;
ThetaEarth = linspace(0,2*pi,100);
rrEarth = (hEarth^2/mu)*(1./(1+eEarth*cos(ThetaEarth)));
polarplot(ThetaEarth,rrEarth,'b--')


% Venus
hVenus = norm(cross([1.36218e+07  -1.07934e+08  -2.26085e+06],[34.51  4.26073
-1.93382]));
eVenus = 0.0067;
ThetaVenus = linspace(0,2*pi,100);
rrVenus = (hVenus^2/mu)*(1./(1+eVenus*cos(ThetaVenus)));
polarplot(ThetaVenus,rrVenus,'r--')

% Sol
```

```
polarplot(0,0,'ro','MarkerSize',12,'MarkerFaceColor',[0.9100    0.4100
0.1700])
legend('Flown Transfer Orbit','Earth at Launch','Venus at Arrival','Earth
Orbit','Venus Orbit','Sol','Location','SouthEast')
%% Venus Flyby 1
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%
% V infinity is found by norm(V1)-norm(Vp1). This is for leaving Earth and
% and going to Venus. I need to take this value and use it as my approach
% velocity for the flyby.
%
% The values that I need to find are listed in Rori's section of the
% report. Calculate those same ones and put the in the report. Also,
% compare those values with data that you can find in Venus trajectory
% papers.
%
% ORBITAL ELEMENTS EQUATIONS:
%           efly1 = 1 + ((rperiapsis*Vinf^2)/(muVenus))
%           delta (turn angle) = 2*asin(1/efly1)
%           h = rperiapsis*sqrt(vinf^2 + ((2*muVenus)/rperiapsis)
%           Vperiapsis = sqrt(norm(Vinf)^2+(2*muVenus/rperiapsis))
%
%
muVenus = 324900;
Vinf1 = 6.03; % Taken from Cassini Manuever Expreience: Launch and Early C.
rVenus = 6052;
Alt1 = 284;
rperiapsis = rVenus+Alt1;
efly1 = 1 + ((rperiapsis*Vinf1^2)/(muVenus));
delta1 = 2*asin(1/efly1)*(180/pi);
h1 = rperiapsis*sqrt(Vinf1^2 + ((2*muVenus)/rperiapsis));
Vperiapsis = sqrt(norm(Vinf1)^2+(2*muVenus/rperiapsis));

fprintf('Venus Flyby 1 Orbital Parameters: \n')
fprintf('Eccentricity =          %g \n',efly1)
fprintf('Turn Angle =            %g degrees \n',delta1)
fprintf('Angular Momentum =      %g (km^2/s^2) \n',h1)
fprintf('Periapsis Velocity =    %g (km/s) \n',Vperiapsis)

%
%
% NOTES FROM MEHRAN:
% Calculate the hyperbolic exit energy using energy eq with mu/r going to
% zero and then from that you can get a (semi major). Use mu earth.
%
%
%% VENUS-VENUS ORBIT
%
% The known period is 425 days, which = 36720000 seconds.
TVV = 36720000;

% Calculate semi-major axis
aVV = ((sqrt(mu)*TVV)/(2*pi))^(2/3);

% Use julian day function to get h and e and then calculate the distance of
% Venus from the sun on the day of flyby1
d = 26;
m = 4;
```

```
y = 1998;
UT = 12;
planet = 3;

% Call Julian Day Function to obtain six orbital elements (VERIFIED)
[J0,T0,JD,h,a,e,I,Omega,omegaBar,L,omega,M] =
Julian_Day_Function(d,m,y,UT,planet);
hVVcheck = h;
aVVcheck = a;
eVVcheck = e;
MVV = M;

% Calcualte distance from sun
% Find Eccentric Anomaly for Earth on day of launch
[Eanomaly]=Newtons_Alg_Function(MVV*(pi/180),eVVcheck*(pi/180));
Eanomaly = Eanomaly*(180/pi);

% Find the true anomaly
ThetaVV = 2*atand(sqrt((1+eVVcheck)/(1-eVVcheck))*tand(Eanomaly/2));
if ThetaVV < 0
    ThetaVV = ThetaVV + 360;
end

% Calculate the distance of earth from sun on launch day
rVV = (hVVcheck^2/mu)*(1/(1+eVVcheck*cosd(ThetaVV)));

% Calculate radius of perigee and apogee
rpVV = (rVenus + 603) + rVV;
raVV = (2*aVV) - rpVV;

% Calculate eccentricity and angular momentum
eVV = (raVV - rpVV)/(raVV + rpVV);
hVV = sqrt(mu*r*(1+eVV));

% Generate postion values for both calculated values (check and regular)
% rVVcheck = (hVVcheck^2/mu)*(1./(1+eVVcheck.*cos(ThetaVenus)));
% polarplot(ThetaVenus,rVVcheck,'m')
rVV = (hVV^2/mu)*(1./(1+eVV.*cos(ThetaVenus)));
%polarplot(ThetaVenus,rVV,'k')



%% VENUS FLYBY 2
%
%
muVenus = 324900;
Vinf2 = 9.41; % Taken from Cassini Manuever Expreience: Launch and Early C.
rVenus = 6052;
Alt2 = 284;
rperiapsis2 = rVenus+Alt2;
efly2 = 1 + ((rperiapsis2*Vinf2^2)/(muVenus));
delta2 = 2*asin(1/efly2)*(180/pi);
h2 = rperiapsis2*sqrt(Vinf2^2 + ((2*muVenus)/rperiapsis2));
Vperiapsis2 = sqrt(norm(Vinf2)^2+(2*muVenus/rperiapsis2));

fprintf('\n Venus Flyby 2 Orbital Parameters: \n')
fprintf('Eccentricity =         %g \n',efly2)
fprintf('Turn Angle =           %g degrees \n',delta2)
fprintf('Angular Momentum =     %g (km^2/s^2) \n',h2)
fprintf('Periapsis Velocity =   %g (km/s) \n',Vperiapsis2)

%% EARTH FLYBY
%
%
muEarth = 398600;
Vinf3 = 16.01; % Taken from Cassini Manuever Expreience: Launch and Early C.
rEarth = 6378;
Alt3 = 1175;
rperiapsis3 = rEarth+Alt3;
efly3 = 1 + ((rperiapsis3*Vinf3^2)/(muEarth));
delta3 = 2*asin(1/efly3)*(180/pi);
h3 = rperiapsis3*sqrt(Vinf3^2 + ((2*muEarth)/rperiapsis3));
Vperiapsis3 = sqrt(norm(Vinf3)^2+(2*muEarth/rperiapsis3));

fprintf('\n Earth Flyby Orbital Parameters: \n')
fprintf('Eccentricity =         %g \n',efly3)
fprintf('Turn Angle =           %g degrees \n',delta3)
fprintf('Angular Momentum =     %g (km^2/s^2) \n',h3)
fprintf('Periapsis Velocity =   %g (km/s) \n',Vperiapsis3)
```

```
%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~
  function ...
  [planet1, planet2, trajectory] = interplanetary(depart,
arrive)
%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~
%{
  This function determines the spacecraft trajectory from
the sphere
  of influence of planet 1 to that of planet 2 using
Algorithm 8.2

  mu          - gravitational parameter of the sun
(km^3/s^2)
  dum         - a dummy vector not required in this
procedure

  planet_id   - planet identifier:
                 1 = Mercury
                 2 = Venus
                 3 = Earth
                 4 = Mars
                 5 = Jupiter
                 6 = Saturn
                 7 = Uranus
                 8 = Neptune
                 9 = Pluto

  year        - range: 1901 - 2099
  month       - range: 1 - 12
  day         - range: 1 - 31
  hour        - range: 0 - 23
  minute      - range: 0 - 60
  second      - range: 0 - 60

  jd1, jd2    - Julian day numbers at departure and arrival
  tof         - time of flight from planet 1 to planet 2 (s)

  Rp1, Vp1    - state vector of planet 1 at departure (km,
km/s)
```

```
  Rp2, Vp2   - state vector of planet 2 at arrival (km,
km/s)
  R1, V1     - heliocentric state vector of spacecraft at
               departure (km, km/s)
  R2, V2     - heliocentric state vector of spacecraft at
               arrival (km, km/s)

  depart     - [planet_id, year, month, day, hour, minute,
second]
               at departure
  arrive     - [planet_id, year, month, day, hour, minute,
second]
               at arrival

  planet1    - [Rp1, Vp1, jd1]
  planet2    - [Rp2, Vp2, jd2]
  trajectory - [V1, V2]

  User M-functions required: planet_elements_and_sv,
lambert
%}
% ----------------------------------------------------------
-----------

global mu


planet_id = depart(1);
year      = depart(2);
month     = depart(3);
day       = depart(4);
hour      = depart(5);
minute    = depart(6);
second    = depart(7);

%...Use Algorithm 8.1 to obtain planet 1's state vector
(don't
%...need its orbital elements ["dum"]):
[dum, Rp1, Vp1, jd1] = planet_elements_and_sv ...
              (planet_id, year, month, day, hour, minute,
second);

planet_id = arrive(1);
year      = arrive(2);
```

```
month      = arrive(3);
day        = arrive(4);
hour       = arrive(5);
minute     = arrive(6);
second     = arrive(7);

%...Likewise use Algorithm 8.1 to obtain planet 2's state
vector:
[dum, Rp2, Vp2, jd2] = planet_elements_and_sv ...
              (planet_id, year, month, day, hour, minute,
second);

tof = (jd2 - jd1)*24*3600;

%...Patched conic assumption:
R1 = Rp1;
R2 = Rp2;

%...Use Algorithm 5.2 to find the spacecraft's velocity at
%   departure and arrival, assuming a prograde trajectory:
[V1, V2] = lambert(R1, R2, tof, 'retro');

planet1    = [Rp1, Vp1, jd1];
planet2    = [Rp2, Vp2, jd2];
trajectory = [V1, V2];

end %interplanetary
%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  function [V1, V2, A] = lambert(R1, R2, t, string)
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%{
  This function solves Lambert's problem.

  mu          - gravitational parameter (km^3/s^2)
  R1, R2      - initial and final position vectors (km)
  r1, r2      - magnitudes of R1 and R2
  t           - the time of flight from R1 to R2 (a
constant) (s)
  V1, V2      - initial and final velocity vectors (km/s)
  c12         - cross product of R1 into R2
```

```
  theta      - angle between R1 and R2
  string     - 'pro'   if the orbit is prograde
               'retro' if the orbit is retrograde
  A          - a constant given by Equation 5.35
  z          - alpha*x^2, where alpha is the reciprocal of
the
               semimajor axis and x is the universal
anomaly
  y(z)       - a function of z given by Equation 5.38
  F(z,t)     - a function of the variable z and constant t,
             - given by Equation 5.40
  dFdz(z)    - the derivative of F(z,t), given by Equation
5.43
  ratio      - F/dFdz
  tol        - tolerance on precision of convergence
  nmax       - maximum number of iterations of Newton's
procedure
  f, g       - Lagrange coefficients
  gdot       - time derivative of g
  C(z), S(z) - Stumpff functions
  dum        - a dummy variable

  User M-functions required: stumpC and stumpS
%}
% ----------------------------------------------

global mu
mu = 1.32712e11;

%...Magnitudes of R1 and R2:
r1 = norm(R1);
r2 = norm(R2);

c12   = cross(R1, R2);
theta = acos(dot(R1,R2)/r1/r2);

%...Determine whether the orbit is prograde or retrograde:
if nargin < 4 || (~strcmp(string,'retro') &
(~strcmp(string,'pro')))
    string = 'pro';
    fprintf('\n ** Prograde trajectory assumed.\n')
end

if strcmp(string,'pro')
```

```
    if c12(3) <= 0
        theta = 2*pi - theta;
    end
elseif strcmp(string,'retro')
    if c12(3) >= 0
        theta = 2*pi - theta;
    end
end

%...Equation 5.35:
A = sin(theta)*sqrt(r1*r2/(1 - cos(theta)));

%...Determine approximately where F(z,t) changes sign, and
%...use that value of z as the starting value for Equation
5.45:
z = -100;
while F(z,t) < 0
    z = z + 0.1;
end

%...Set an error tolerance and a limit on the number of
iterations:
tol   = 1.e-8;
nmax  = 5000;

%...Iterate on Equation 5.45 until z is determined to
within the
%...error tolerance:
ratio = 1;
n     = 0;
while (abs(ratio) > tol) & (n <= nmax)
    n     = n + 1;
    ratio = F(z,t)/dFdz(z);
    z     = z - ratio;
end

%...Report if the maximum number of iterations is exceeded:
if n >= nmax
    fprintf('\n\n **Number of iterations exceeds %g in
''lambert'' \n\n ',nmax)
end

%...Equation 5.46a:
f    = 1 - y(z)/r1;
```

```
%...Equation 5.46b:
g    = A*sqrt(y(z)/mu);

%...Equation 5.46d:
gdot = 1 - y(z)/r2;

%...Equation 5.28:
V1   = 1./g*(R2 - f*R1)

%...Equation 5.29:
V2   = 1/g*(gdot*R2 - R1)


%return

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Subfunctions used in the main body:
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%...Equation 5.38:
function dum = y(z)
    dum = r1 + r2 + A*(z*S(z) - 1)/sqrt(C(z));
end

%...Equation 5.40:
function dum = F(z,t)
    dum = (y(z)/C(z))^1.5*S(z) + A*sqrt(y(z)) - sqrt(mu)*t;
end

%...Equation 5.43:
function dum = dFdz(z)
    if z == 0
        dum = sqrt(2)/40*y(0)^1.5 + A/8*(sqrt(y(0)) +
A*sqrt(1/2/y(0)));
    else
        dum = (y(z)/C(z))^1.5*(1/2/z*(C(z) - 3*S(z)/2/C(z))
...
               + 3*S(z)^2/4/C(z)) +
A/8*(3*S(z)/C(z)*sqrt(y(z)) ...
               + A*sqrt(C(z)/y(z)));
    end
end
```

```
%...Stumpff functions:
function dum = C(z)
    dum = stumpC(z);
end

function dum = S(z)
    dum = stumpS(z);
end

end %lambert


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


% This function evaluates the Stumpff function C(z)
according to
% Equation 3.50

function c = stumpC(z)


% z - input arguement
% s - Value of C(z)

if z > 0;
    c = (1-cos(sqrt(z)))/z;
elseif z < 0
    c = (cosh(sqrt(z))-1)/z;
else
    c = 1/2;
end




% This function evaluates the Stumpff function C(z)
according to
% Equation 3.50

function c = stumpC(z)


% z - input arguement
% s - Value of C(z)
```

```
if z > 0;
    c = (1-cos(sqrt(z)))/z;
elseif z < 0
    c = (cosh(sqrt(z))-1)/z;
else
    c = 1/2;
end



% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Example_8_08
% ~~~~~~~~~~~~
%{
  This program uses Algorithm 8.2 to solve Example 8.8.

  mu            - gravitational parameter of the sun
(km^3/s^2)
  deg           - conversion factor between degrees and
radians
  pi            - 3.1415926...

  planet_id     - planet identifier:
                   1 = Mercury
                   2 = Venus
                   3 = Earth
                   4 = Mars
                   5 = Jupiter
                   6 = Saturn
                   7 = Uranus
                   8 = Neptune
                   9 = Pluto

  year          - range: 1901 - 2099
  month         - range: 1 - 12
  day           - range: 1 - 31
  hour          - range: 0 - 23
  minute        - range: 0 - 60
  second        - range: 0 - 60

  depart        - [planet_id, year, month, day, hour,
minute, second]
                  at departure
```

```
  arrive        - [planet_id, year, month, day, hour,
minute, second]
                  at arrival

  planet1       - [Rp1, Vp1, jd1]
  planet2       - [Rp2, Vp2, jd2]
  trajectory    - [V1, V2]

  coe           - orbital elements [h e RA incl w TA]
                  where
                    h    = angular momentum (km^2/s)
                    e    = eccentricity
                    RA   = right ascension of the ascending
                           node (rad)
                    incl = inclination of the orbit (rad)
                    w    = argument of perigee (rad)
                    TA   = true anomaly (rad)
                    a    = semimajor axis (km)

  jd1, jd2      - Julian day numbers at departure and
arrival
  tof           - time of flight from planet 1 to planet 2
(days)

  Rp1, Vp1      - state vector of planet 1 at departure (km,
km/s)
  Rp2, Vp2      - state vector of planet 2 at arrival (km,
km/s)
  R1, V1        - heliocentric state vector of spacecraft at
                  departure (km, km/s)
  R2, V2        - heliocentric state vector of spacecraft at
                  arrival (km, km/s)

  vinf1, vinf2 - hyperbolic excess velocities at departure
                  and arrival (km/s)

  User M-functions required: interplanetary, coe_from_sv,
                             month_planet_names
%}
% ---------------------------------------------

clear all; clc
global mu
mu  = 1.327124e11;
```

```
deg = pi/180;

%...Data declaration for Example 8.8:

%...Departure
planet_id = 3;
year      = 1996;
month     = 11;
day       = 7;
hour      = 0;
minute    = 0;
second    = 0;
depart = [planet_id  year  month  day  hour  minute
second];

%...Arrival
planet_id = 4;
year      = 1997;
month     = 9;
day       = 12;
hour      = 0;
minute    = 0;
second    = 0;
arrive = [planet_id  year  month  day  hour  minute
second];

%...

%...Algorithm 8.2:
[planet1, planet2, trajectory] = interplanetary(depart,
arrive);

R1  = planet1(1,1:3);
Vp1 = planet1(1,4:6);
jd1 = planet1(1,7);

R2  = planet2(1,1:3);
Vp2 = planet2(1,4:6);
jd2 = planet2(1,7);

V1  = trajectory(1,1:3);
V2  = trajectory(1,4:6);

tof = jd2 - jd1;
```

```
%...Use Algorithm 4.2 to find the orbital elements of the
%   spacecraft trajectory based on [Rp1, V1]...
coe  = coe_from_sv(R1, V1, mu);
%   ... and [R2, V2]
coe2 = coe_from_sv(R2, V2, mu);

%...Equations 8.94 and 8.95:
vinf1 = V1 - Vp1;
vinf2 = V2 - Vp2;

%...Echo the input data and output the solution to
%   the command window:
fprintf('--------------------------------------------------
---')
fprintf('\n Example 8.8')
fprintf('\n\n Departure:\n');
fprintf('\n   Planet: %s', planet_name(depart(1)))
fprintf('\n   Year  : %g', depart(2))
fprintf('\n   Month : %s', month_name(depart(3)))
fprintf('\n   Day   : %g', depart(4))
fprintf('\n   Hour  : %g', depart(5))
fprintf('\n   Minute: %g', depart(6))
fprintf('\n   Second: %g', depart(7))
fprintf('\n\n   Julian day: %11.3f\n', jd1)
fprintf('\n   Planet position vector (km)    = [%g  %g
%g]', ...
                                               R1(1),R1(2),
R1(3))

fprintf('\n   Magnitude                      = %g\n',
norm(R1))

fprintf('\n   Planet velocity (km/s)         = [%g  %g
%g]', ...
                                 Vp1(1), Vp1(2), Vp1(3))

fprintf('\n   Magnitude                      = %g\n',
norm(Vp1))

fprintf('\n   Spacecraft velocity (km/s)     = [%g  %g
%g]', ...
                                               V1(1),
V1(2), V1(3))
```

```
fprintf('\n   Magnitude                          = %g\n',
norm(V1))

fprintf('\n   v-infinity at departure (km/s) = [%g  %g
%g]', ...
                                      vinf1(1), vinf1(2),
vinf1(3))

fprintf('\n   Magnitude                          = %g\n',
norm(vinf1))

fprintf('\n\n Time of flight = %g days\n', tof)

fprintf('\n\n Arrival:\n');
fprintf('\n   Planet: %s', planet_name(arrive(1)))
fprintf('\n   Year  : %g', arrive(2))
fprintf('\n   Month : %s', month_name(arrive(3)))
fprintf('\n   Day   : %g', arrive(4))
fprintf('\n   Hour  : %g', arrive(5))
fprintf('\n   Minute: %g', arrive(6))
fprintf('\n   Second: %g', arrive(7))
fprintf('\n\n   Julian day: %11.3f\n', jd2)
fprintf('\n   Planet position vector (km)    = [%g  %g
%g]', ...
                                            R2(1), R2(2),
R2(3))

fprintf('\n   Magnitude                         = %g\n',
norm(R1))

fprintf('\n   Planet velocity (km/s)         = [%g  %g
%g]', ...
                                  Vp2(1), Vp2(2), Vp2(3))

fprintf('\n   Magnitude                         = %g\n',
norm(Vp2))

fprintf('\n   Spacecraft Velocity (km/s)     = [%g  %g
%g]', ...
                                            V2(1), V2(2),
V2(3))
```

```
fprintf('\n   Magnitude                         = %g\n',
norm(V2))

fprintf('\n   v-infinity at arrival (km/s)   = [%g  %g
%g]', ...
                                    vinf2(1), vinf2(2),
vinf2(3))

fprintf('\n   Magnitude                         = %g',
norm(vinf2))

fprintf('\n\n\n Orbital elements of flight trajectory:\n')

fprintf('\n  Angular momentum (km^2/s)                   =
%g',...

coe(1))
fprintf('\n  Eccentricity                                =
%g',...

coe(2))
fprintf('\n  Right ascension of the ascending node (deg) =
%g',...

coe(3)/deg)
fprintf('\n  Inclination to the ecliptic (deg)           =
%g',...

coe(4)/deg)
fprintf('\n  Argument of perihelion (deg)                =
%g',...

coe(5)/deg)
fprintf('\n  True anomaly at departure (deg)             =
%g',...

coe(6)/deg)
fprintf('\n  True anomaly at arrival (deg)               =
%g\n', ...

coe2(6)/deg)
fprintf('\n  Semimajor axis (km)                         =
%g',...
coe(7))
% If the orbit is an ellipse, output the period:
if coe(2) < 1
    fprintf('\n  Period (days)
= %g', ...

2*pi/sqrt(mu)*coe(7)^1.5/24/3600)
end
fprintf('\n-----------------------------------------------
-----\n')
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
% script to calculate earth e from document %
alt = 1175; % km
radius = 6378; % km
vinf = 16.01; % km/s
rp = alt + radius; %km
mu_earth = 398600; % km
convert = 180/pi;


e = 1 + (rp*vinf^2)/mu_earth
delta = 2*asin(1/e) * convert
h = rp*sqrt(vinf^2+(2*mu_earth)/rp)
a = h^2/mu_earth*(1/(e^2-1))
vp = sqrt(mu_earth/a+2*mu_earth/rp)
vinf_calc = sqrt(mu_earth/a)




function [ qqq ] = position_julian(p1, UT, y, m, d)
% orbital_elements = orbital_elements(p1, UT, y, m, d) :
finds the six orbital elements for a given planet as well
as
% the position vector and velocity vector in the
heliocentric plane.
% p1 is the desired planet*.
% UT is the universal time on a 24 hour clock.
% y is the year.
% m is the month.
% d is the day.
```

```
% * for the planet input (p1) please enter the number
associated with the
% planet (1-9)
% Mercury (1), Venus (2), Earth (3), Mars (4), Jupiter (5)
Saturn (6), Uranus (7), Neptune (8), and Pluto (9).

% for orbital elements, unmute lines 236-242 and 293


format long

% assign known variables %
c = 1721013.5;
counter = 10000;
tol = 10e-10;
mu = 132712000000;  %km^3/s^2

%% determination of the planets p1 and p2 %

% Mercury (1) %
if p1 == 1
   a0 = 0.38709927 * 1.49597871e8; %km
   da0 = 0.00000037 * 1.49597871e8; %km/Cy
   e0 = 0.00000037 ;
   de0 = 0.00001906 ; %1/Cy
   i0 = 7.00497902 /180*pi; %rad
   di0 = -0.00594749 /180*pi; %rad/Cy
   Omega0 = 48.33076593  /180*pi; %rad
   dOmega0 = -0.1253408 /180*pi; %rad/Cy
   varpi0 = 77.45779628 /180*pi; %rad
   dvarpi0 = 0.16047689 /180*pi; %rad/Cy
   L0 = 252.25032350 /180*pi; %rad
   dL0 = 149472.67411175 /180*pi; %rad/Cy
end

% Venus (2) %
if p1 == 2
   a0 = 0.72333566 * 1.49597871e8; %km
   da0 = 0.00000390 * 1.49597871e8; %km/Cy
   e0 = 0.00677672;
   de0 =  -0.00004107; %1/Cy
   i0 = 3.39467605 /180*pi; %rad
   di0 = -0.00078890 /180*pi; %rad/Cy
   Omega0 = 76.67984255 /180*pi; %rad
   dOmega0 = -0.27769418 /180*pi; %rad/Cy
   varpi0 = 131.60246718 /180*pi; %rad
   dvarpi0 = 0.00268329 /180*pi; %rad/Cy
   L0 = 181.97909950  /180*pi; %rad
   dL0 = 58517.81538729  /180*pi; %rad/Cy
end

% Earth (3) %
if p1 == 3
    a0 = 1.00000261 * 1.49597871e8; %km
    da0 = 0.00000562* 1.49597871e8; %km/Cy
    e0 = 0.01671123;
    de0 = -0.00004392; %1/Cy
    i0 = -0.00078890 /180*pi; %rad
    di0 = -0.00001531 /180*pi; %rad/Cy
    Omega0 = 0; %rad
    dOmega0 = 0; %rad/Cy
    varpi0 = 102.93768193 /180*pi; %rad
    dvarpi0 =  0.32327364 /180*pi;  %rad/Cy
    L0 = 100.46457166 /180*pi; %rad
    dL0 = 35999.37244981 /180*pi; %rad/Cy
end

% Mars (4) %
if p1 == 4
    a0 = 1.52371034* 1.49597871e8; %km
    da0 = 0.0001847* 1.49597871e8; %km/Cy
    e0 = 0.09339410;
    de0 = 0.00007882; %1/Cy
    i0 = 1.84969142 /180*pi; %rad
    di0 = -0.00813131 /180*pi; %rad/Cy
    Omega0 = 49.55953891 /180*pi; %rad
    dOmega0 = 0.29257343 /180*pi; %rad/Cy
    varpi0 = 23.94362959 /180*pi; %rad
    dvarpi0 = 0.44441088 /180*pi; %rad/Cy
    L0 = -4.55343205 /180*pi; %rad
    dL0 = 19140.30268499 /180*pi; %rad/Cy
end

% Jupiter (5) %
if p1 == 5
    a0 = 5.20288700 * 1.49597871e8; %km
    da0 = -0.00011607 * 1.49597871e8; %km/Cy
    e0 = 0.04838624 ;
    de0 = 0.00013253 ; %1/Cy
    i0 = 1.30439695 /180*pi; %rad
    di0 = -0.00183714  /180*pi; %rad/Cy
    Omega0 = 100.47390909 /180*pi; %rad
    dOmega0 = 0.20469106  /180*pi; %rad/Cy
    varpi0 = 14.72847983 /180*pi; %rad
    dvarpi0 = 0.21252668  /180*pi; %rad/Cy
    L0 = 34.39644501  /180*pi; %rad
    dL0 = 3034.74612775  /180*pi; %rad/Cy
end

% Saturn (6) %
if p1 == 6
    a0 = 9.53667594 * 1.49597871e8; %km
    da0 = -0.00125060 * 1.49597871e8; %km/Cy
    e0 = 0.05386179 ;
    de0 = -0.00050991 ; %1/Cy
    i0 = 2.48599187 /180*pi; %rad
    di0 = 0.00193609  /180*pi; %rad/Cy
    Omega0 = 113.66242448 /180*pi; %rad
    dOmega0 = -0.28867794 /180*pi; %rad/Cy
    varpi0 = 92.59887831 /180*pi; %rad
    dvarpi0 = -0.41897216 /180*pi; %rad/Cy
    L0 = 49.95424423 /180*pi; %rad
    dL0 = 1222.49362201  /180*pi; %rad/Cy
end

% Uranus (7) %
if p1 == 7
    a0 = 19.18916464 * 1.49597871e8; %km
    da0 = -0.00196176  * 1.49597871e8; %km/Cy
    e0 = 0.04725744  ;
    de0 = -0.00004397  ; %1/Cy
    i0 = 0.77263783  /180*pi; %rad
    di0 = -0.00242939   /180*pi; %rad/Cy
    Omega0 = 74.01692503  /180*pi; %rad
    dOmega0 = 0.04240589  /180*pi; %rad/Cy
    varpi0 = 170.95427630 /180*pi; %rad
    dvarpi0 = 0.40805281 /180*pi; %rad/Cy
    L0 = 313.23810451 /180*pi; %rad
    dL0 = 428.48202785 /180*pi; %rad/Cy
end

% Neptune (8) %
```

```
if p1 == 8
    a0 = 30.06992276 * 1.49597871e8; %km
    da0 = 0.00026291 * 1.49597871e8; %km/Cy
    e0 = 0.00859048;
    de0 = 0.00005105; %1/Cy
    i0 = 1.77004347  /180*pi; %rad
    di0 = 0.00035372    /180*pi; %rad/Cy
    Omega0 = 131.78422574 /180*pi; %rad
    dOmega0 = -0.00508664 /180*pi; %rad/Cy
    varpi0 = 44.96476227 /180*pi; %rad
    dvarpi0 = -0.32241464 /180*pi; %rad/Cy
    L0 = -55.12002969 /180*pi; %rad
    dL0 = 218.45945325 /180*pi; %rad/Cy
end

% Pluto (9) %
if p1 == 9
    a0 = 39.48211675 * 1.49597871e8; %km
    da0 = -0.00031596 * 1.49597871e8; %km/Cy
    e0 = 0.24882730;
    de0 = 0.00005170; %1/Cy
    i0 = 17.14001206 /180*pi; %rad
    di0 = 0.00004818 /180*pi; %rad/Cy
    Omega0 = 110.30393684 /180*pi; %rad
    dOmega0 = -0.01183482 /180*pi; %rad/Cy
    varpi0 = 224.06891629 /180*pi; %rad
    dvarpi0 = -0.04062942 /180*pi; %rad/Cy
    L0 = 238.92903833 /180*pi; %rad
    dL0 = 145.20780515 /180*pi; %rad/Cy
end

%% calculate the Julian day %
j0 = 367*y - fix(7*(y+fix((m+9)/12))/4) + fix(275*m/9) + d
+ c;
hr = UT/24;
jd = j0+hr;

T0 = (jd-2451545)/36525;

%% find six orbital elements at the given Julian date %

a = a0 + da0*T0;
e = e0 + de0*T0;
i = i0 + di0*T0;
```

```
Omega = Omega0 + dOmega0*T0;
varpi = varpi0 + dvarpi0*T0;
L = (L0 + dL0*T0);

%% adjust the variables to be withing the range of 0 to
2*pi radians

if i > (2*pi)
    while i > (2*pi)
        i = i-2*pi;
    end
end

if i < 0
    while i < 0
        i = i+2*pi;
    end
end

if Omega > (2*pi)
    while Omega > (2*pi)
        Omega = Omega-2*pi;
    end
end

if Omega < 0
    while Omega < 0
        Omega = Omega+2*pi;
    end
end

if varpi > (2*pi)
    while varpi > (2*pi)
        varpi = varpi-2*pi;
    end
end

if varpi < 0
    while varpi < 0
        varpi = varpi+2*pi;
    end
end

if L > (2*pi)
```

```
    while L > (2*pi)
        L = L-2*pi;
    end
end

if L < 0
    while L < 0
        L =L+2*pi;
    end
end

%% calculate angular momentum (h), argument of perihelion
(\omega), and mean anomaly (M) at JD for each planet p1 and
p2%
h = sqrt(mu*a*(1-e^2));
omega = varpi-Omega;

omega_in_degrees = omega *180/pi; % output for six orbital
elements

M = L-varpi;

if M > (2*pi)
    while M > (2*pi)
        M = M-2*pi;
    end
end

if M < 0
    while M < 0
        M =M + 2*pi;
    end
end

%% calculate the eccentric anomaly (E) for p1 %

% determine the initial E value
if M < pi
    E0 = M+e/2;
end

if M > pi
    E0 = M-e/2;
end
```

```
for ii = 1:counter
     E = E0-(E0-e*sin(E0)-M)/(1-e*cos(E0));
     if (E0-e*sin(E0)-M)/(1-e*cos(E0)) < tol
         break
     end
     E0 = E;
end

checkE = E*180/pi; % unmut to check if E matches value in
book

%% calculate theta for p1 %

theta = 2*atan(tan(E/2)/sqrt((1-e0)/(1+e0))) ;

%% check if theta is between 0 and 2*pi

if theta > (2*pi)
    while theta > (2*pi)
        theta = theta-2*pi;
    end
end

if theta < 0
    while theta < 0
        theta = theta+2*pi;
    end
end
theta_in_degrees = theta *180/pi; % output (unmute for
theta as an output)

%% find dcm and initial r
dcm = [-sin(Omega)*cos(i)*sin(omega)+cos(Omega)*cos(omega)
-sin(Omega)*cos(i)*cos(omega)-cos(Omega)*sin(omega)
sin(Omega)*sin(i);
    cos(Omega)*cos(i)*sin(omega)+sin(Omega)*cos(omega)
cos(Omega)*cos(i)*cos(omega)-sin(Omega)*sin(omega) -
cos(Omega)*sin(i);
    sin(i)*sin(omega) sin(i)*cos(omega) cos(i)];

r_before_dcm = h^2/mu*(1/(1+e*cos(theta)));

%% find R and V in perifocal frame
```

```
R_peri = r_before_dcm*[cos(theta);sin(theta);0];
V_peri = mu/h.*[-sin(theta);e+cos(theta);0]; % output
v_peri = norm(V_peri);
%% find the new R and V and r and v

R = dcm*R_peri %ourput
V = dcm*V_peri %output
r = norm(R);
v = norm(V)

%% checking variables
%check the six orbital elements
a_in_km = a; % output for six orbital elements
e_planet = e; % output for six orbital elements
i_in_degrees = i *180/pi; % output for six orbital elements
Omega_in_degrees = Omega *180/pi; % output for six orbital
elements
cvarpi = varpi *180/pi;
L_in_degrees = L *180/pi; % output for six orbital elements
theta_in_degrees = theta *180/pi; % output (unmute for
theta as an outp
checkM = M*180/pi;
checkE = E*180/pi; % unmut to check if E matches value in
book
h = sqrt(mu*a*(1-e^2));

end



function [lagrange_rvtime] = lagrange_rvtime(R, V, time)
% lagrange_rv(R, V, theta) is a funtion that computes
poistion and velocity
% vectors after a delta time around earth
% R is the initial position vector in km
% V is the initial velocity vector in km/s
% time is the time (in seconds) from when the object was
first observed to
% the desired observation point. delta time [s]
% all vectors must be in the format of [xi yj zk]

% contsants %
mu = 398600; % mu for earth [km^3/s^2]
```

```
%% calculating magnitudes of initial vectors
r0 = norm(R); % magnitude of initial position vector [km]
v0 = norm(V); % magnitude of initial velocity vector [km/s]

%% radial components of velocity [km] and magnitude of
angular momentum [km^2/s]
v_radial = dot(R,V)/r0;
h = r0*sqrt(v0^2-v_radial^2);

%% calculating f, g, fdot, gdot

% breaking the equatino up to identify problems

fa = 1-mu/(2*r0^3)*time^2;
fb = mu/2*dot(R,V)/r0^5*time^3;
fc = mu/24*(-2*mu/r0^6+3*v0^2/r0^5-
15*(dot(R,V))^2/r0^7)*time^4;

f = fa +fb +fc; % equ 2.172
g = time-1/6*mu/r0^3*time^3+mu/4*dot(R,V)/r0^5*time^4;

%% calculating the new position and velocity vectors
R_final = f*R + g*V
r_final = norm(R_final);

%% finding the true anomaly
H = cross(R,V);
C = cross(V,H)-mu*(R/r0); % equation 2.39
e_vector = C/mu;
e = norm(e_vector);
theta_initial = acos((h^2/(mu*r0)-1)/e);
theta_final = acos((h^2/(mu*r_final)-1)/e);
dtheta = theta_final-theta_initial

%check = theta_initial + theta;
%if (pi <= check)&&(check < 2*pi)
%    theta_final = theta_final+pi;
%end
%theta_final = theta_final *180/pi


end
```

```
close all; clear all; clc
% Define planet
%   Earth = 1
%   Mars = 2
%   Venus = 3
    planet = 1;

% Define initial day/time
    d = 1;
    m = 1;
    y = 2005;
    UT = 12;

% Create mean anomaly and eccentricity vectors
    listM = [];
    liste = [];
for month = 1:12
    for day = 1:30
        [J0,T0,JD,h,a,e,I,Omega,omegaBar,L,omega,M] =
Julian_Day_Function(day,month,y,UT,planet);
        listM(day,month) = M;
        liste(day,month) = e;
    end
end


%% Find the eccentric anomaly (E)
Eanomaly = [];
for monthCount = 1:12
    for dayCount = 1:30
        Eanomaly(dayCount,monthCount)=
Newtons_Alg_Function(listM(dayCount,monthCount)*(pi/180),li
ste(dayCount,monthCount));
    end
end
Eanomaly = Eanomaly.*(180/pi);


%% Find the true anomaly (Theta)
Theta = [];
for months = 1:12
    for days = 1:30
```

```
        Theta(days,months) =
2*atand(sqrt((1+liste(days,months))/(1-
liste(days,months)))*tand(Eanomaly(days,months)/2));
        if Theta(days,months) > 0
            while Theta(days,months) > 0
                Theta(days,months) = Theta(days,months) -
360;
            end
        end
        if Theta(days,months) < 0
            while Theta(days,months) < 0
                Theta(days,months) = Theta(days,months) +
360;
            end
        end
    end
end


%% Calculate the position (r) and itemize the r matrix into
months
mu = 1.32712e+11;
r = [];
for monthz = 1:12
    for dayz = 1:30
        r(dayz,monthz) =
(h^2/mu)*(1/(1+(e*cosd(Theta(dayz,monthz)))));
    end
end
Jan = r(1:end,1);
Feb = r(1:end,2);
March = r(1:end,3);
April = r(1:end,4);
May = r(1:end,5);
June = r(1:end,6);
July = r(1:end,7);
Aug = r(1:end,8);
Sep = r(1:end,9);
Oct = r(1:end,10);
Nov = r(1:end,11);
Dec = r(1:end,12);
```

```
%% Sort through months to find maximum values and create
maxMonths vector
maxJan = max(Jan);
maxFeb = max(Feb);
maxMarch = max(March);
maxApril = max(April);
maxMay = max(May);
maxJune = max(June);
maxJuly = max(July);
maxAug = max(Aug);
maxSep = max(Sep);
maxOct = max(Oct);
maxNov = max(Nov);
maxDec = max(Dec);

maxMonths = [maxJan maxFeb maxMarch maxApril maxMay maxJune
maxJuly maxAug ...
    maxSep maxOct maxNov maxDec];


%% Determine which month has max r value
for maxCount = 1:length(maxMonths)
    Month = maxMonths(maxCount);
    if Month == max(maxMonths)
        maxMonth = maxCount;
    end
end


%% Determine which day in maxMonth has the max value
for maxDayCount = 1:30
    Day = July(maxDayCount);
    if Day == max(July)
        maxDay = maxDayCount;
    end
end
```

Table of Contents

```
%David Arnold

clear all; close all; %#ok

global mu;
mu = 37931267.73; %km^3/s^2
global Rsat;
Rsat = 60268; %km (radius of saturn Rs)
%define Titan's uV and uS
%find Titan orbital distance and speed
a_T=1221900; %km
r_T=a_T;
v_T=sqrt(mu/a_T);%orbital speed of Titan's circular orbit relative to
 Saturn
[x_T,y_T]=pol2cart(0:pi/100:2*pi,r_T);
```

Phoebe flyby

```
%calculate SOI radius of Phoebe
r_phToSat=12952000; %km
M_ph = 8.292*10^18; %kg
M_sat = 5.6834*10^26; %kg

r_SOIph = SOIradius(r_phToSat, M_ph, M_sat); %km

v_infPh = 6.35; %km/s
r_p = 2070.9; %km
mu_ph = 0.48; %km^3/s^2

e = 1+(r_p*v_infPh^2)/0.48;
%h = ;
delta_turn0 = 2*asin(1/e); %rad
%negligible turning angle
```

Orbital insertion to first burn (OTM-001)

```
phase_inf = 75*pi/180; %phase at infinity, rad
phase_peri = 94*pi/180; %phase at periapsis, rad
theta_inf = -(phase_inf+phase_peri); %true anomaly at infinity, rad
```

```
rp=1.3*Rsat;
f = @(v_inf) (mu/(rp*sqrt(v_inf^2+(2*mu)/rp)))*(1+(rp*v_inf^2)/
mu)*sin(theta_inf)-v_inf;
v_inf=abs(fzero(f, 3));
e = calcecc(rp, v_inf, mu);
h = rp*sqrt(v_inf^2+2*mu/rp);

theta0 = theta_inf:0.01:0;
%r = zeros(1, length(theta));
r0 = (h^2/mu)./(1+e*cos(theta0));
[x0,y0] = pol2cart(theta0,r0);
% for i=1:length(r0)
%     plot(0,0,'ko');
%     hold on;
%     plot(x0(1:i),y0(1:i));
%     hold off;
%     axis([-10^7 10^5 -5*10^5 10^5])
%     set(gca, 'Xdir', 'Reverse')
%     set(gca, 'Ydir', 'Reverse')
%     drawnow;
%     %hold on;
%     %pause(0.05)
% end

%first burn (OTM-001)
ra = 150.8*Rsat;
a1 = (rp+ra)/2; %desired semimajor axis after burn
e1 = (ra-rp)/(ra+rp); %desired eccentricity after burn
vpi = h/rp; %initial velocity at periapsis before burn (km/s)

h1=sqrt(mu*a1*(1-e1^2)); %angular momentum of new orbit
vpf = h1/rp; %final velocity at periapsis after burn (km/s)
deltaV1 = abs(vpf-vpi)*1000; %delta v required from first burn (m/s)

theta1 = 0:0.01:pi;
r1 = (a1*(1-e1^2))./(1+e1*cos(theta1));
[x1,y1] = pol2cart(theta1, r1);

x=[x0 x1];
y=[y0 y1];
```

Second burn (OTM-002)

```
T2 = 124.1*24*3600; %required period after burn
a2 = ((T2*sqrt(mu))/(2*pi))^(2/3);
ra = 150.8*Rsat;

f = @(e2) (a2*(1-e2^2))/(1-e2)-ra;
e2 = fzero(f, 0);
h2 = sqrt(mu*a2*(1-e2^2));

vai = h1/ra; %initial velocity at apoapse before burn
vaf = h2/ra; %final velocity at apoapse after burn
```

```
deltaV2 = abs(vaf-vai)*1000; %delta v required from second burn m/s

theta2 = pi:0.01:(4.47205+2*pi);
r2 = (a2*(1-e2^2))./(1+e2*cos(theta2));
[x2,y2] = pol2cart(theta2, r2);

x=[x x2];
y=[y y2];
```

Titan encounter 1 (TA)

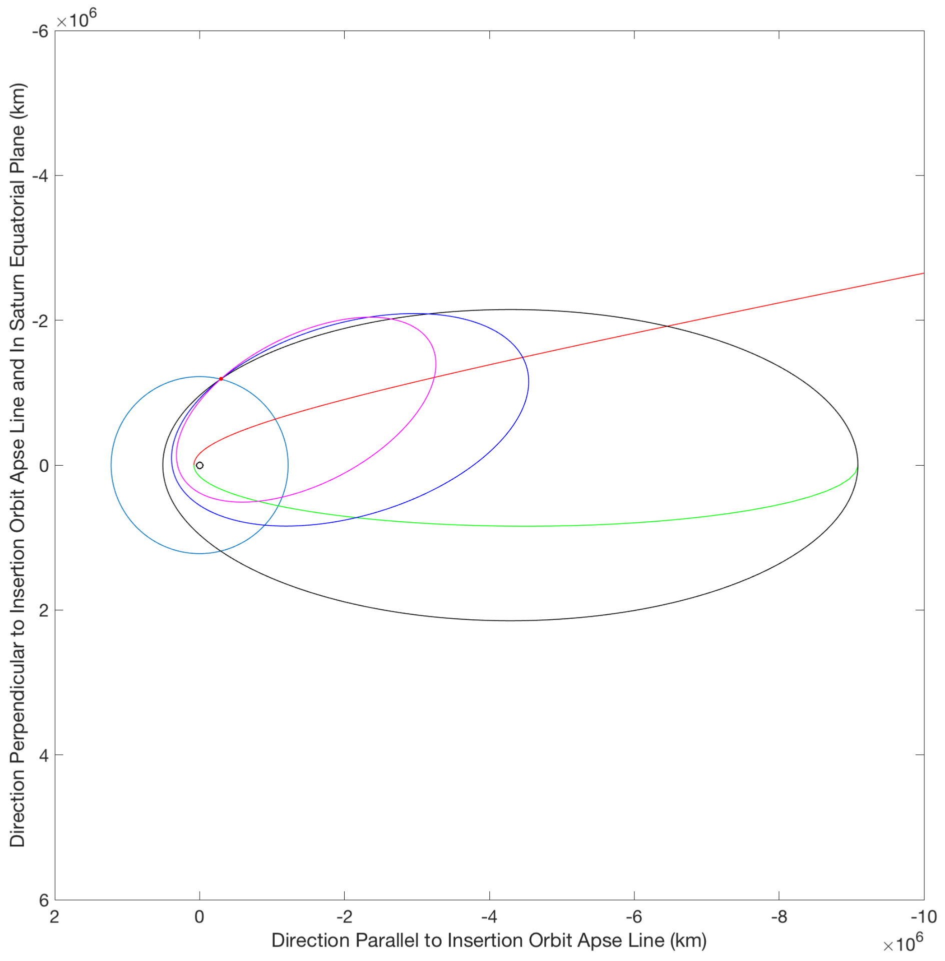
```
T3 = 47.9*24*3600; %desired resonant period after encounter
a3 = ((T3*sqrt(mu))/(2*pi))^(2/3); %desired semimajor axis after
 encounter
rp3 = 6.2*Rsat; %desired periapsis radius after encounter
e3 = 1-rp3/a3;
h3 = sqrt(mu*a3*(1-e3^2));

theta_Titan2 = 2*pi-acos((1/e2)*(((a2*(1-e2^2))/r_T)-1));
theta_Titan3 = 2*pi-acos((1/e3)*(((a3*(1-e3^2))/r_T)-1));
delta_apse = theta_Titan3-theta_Titan2;

v_perp2 = h2/r_T;
v_rad2 = (mu/h2)*e2*sin(theta_Titan2);
v_infin=[v_perp2-v_T; -v_rad2];

v_perp3 = (mu/h3)*(1+e3^2*cos(theta_Titan3));
v_rad3 = (mu/h3)*(e3*sin(theta_Titan3));
v_infout=[v_perp3-v_T; -v_rad3];

deltaVTA = norm(v_infout-v_infin)*1000; %magnitude of deltaV at Titan
 in m/s
%percent difference between inbound and outbound velocities relative
 to
%Titan
perdiff=((norm(v_infin)-norm(v_infout))/
(0.5*(norm(v_infin)+norm(v_infout))))*100;

v_infavg=(norm(v_infin)+norm(v_infout))/2;
v_infavg1 = v_infavg;
delta_turn1=acos((dot(v_infin,v_infout))/
(norm(v_infin)*norm(v_infout)));
mu_T = 8978.2; %Cassini-measured mu of Titan (km^3/s^2)
%calculate required rp above Titan to ensure the desired resulting
 orbit
%around Saturn with the specified period and periapsis mimicking the
 real
%mission
rp_encounter1=(mu_T/v_infavg^2)*(1/sin(delta_turn1/2)-1);

theta3 = theta_Titan3:0.01:theta_Titan3+2*pi;
r3 = (a3*(1-e3^2))./(1+e3*cos(theta3));
```

```
[x3, y3] = pol2cart(theta3, r3);
apse_rotation = [cos(-delta_apse) -sin(-delta_apse); sin(-delta_apse)
 cos(-delta_apse)];
true1=apse_rotation*[x3;y3];
x3true = true1(1,:);
y3true = true1(2,:);
x = [x x3true];
y = [y y3true];
%x2 = [x x3];
%y2 = [y y3];
% for i=1:length(x)
%     plot(0,0,'ko');
%     hold on;
%     plot(x(1:i),y(1:i));
%     %plot(x2(1:i),y2(1:i));
%     plot(x(i),y(i), 'r.', 'MarkerSize', 10);
%     hold off;
%     axis([-10^7 2*10^6 -6*10^6 6*10^6])
%     daspect([1 1 1]);
%     set(gcf, 'Position', [100 100 1000 1000]);
%     %set(gca, 'Xdir', 'Reverse')
%     %set(gca, 'Ydir', 'Reverse')
%     drawnow;
%     %hold on;
%     %pause(0.05)
% end
```

Titan encounter 2 (TB)

```
%define new inbound v as previous outbound v
v_infin = [v_perp3-v_T; -v_rad3];

T4 = 32.1*24*3600; %desired period after Titan encounter 2
a4 = ((T4*sqrt(mu))/(2*pi))^(2/3); %desired semimajor axis
rp4 = 4.8*Rsat;
e4 = 1-rp4/a4;
h4 = sqrt(mu*a4*(1-e4^2));

theta_Titan4 = 2*pi-acos((1/e4)*(((a4*(1-e4^2))/r_T)-1));
delta_apse2 = theta_Titan4-theta_Titan3;

%define outgoing
v_perp4 = (mu/h4)*(1+e4^2*cos(theta_Titan4));
v_rad4 = (mu/h4)*(e4*sin(theta_Titan4));
v_infout=[v_perp4-v_T; -v_rad4];

deltaVTB = norm(v_infout-v_infin)*1000;

delta_turn2=acos((dot(v_infin,v_infout))/
(norm(v_infin)*norm(v_infout)));
mu_T = 8978.2;

v_infavg=(norm(v_infin)+norm(v_infout))/2;
```

```
v_infavg2 = v_infavg;
rp_encounter2 = (mu_T/v_infavg^2)*(1/sin(delta_turn2/2)-1);
theta4 = theta_Titan4:0.01:theta_Titan4+2*pi;
r4 = (a4*(1-e4^2))./(1+e4*cos(theta4));
[x4, y4] = pol2cart(theta4, r4);
delta_apse = delta_apse+delta_apse2;
apse_rotation = [cos(-delta_apse) -sin(-delta_apse); sin(-delta_apse)
 cos(-delta_apse)];
true1=apse_rotation*[x4;y4];
x4true = true1(1,:);
y4true = true1(2,:);
x = [x x4true];
y = [y y4true];
figure(1);
for i=1:length(x)
    plot(0,0,'ko');
    hold on;
    set(0, 'DefaultLineLineWidth', 0.75);
    plot(x_T,y_T);
    if i>0 && i<=length(x0)
        plot(x(1:i),y(1:i),'r');
    elseif i>length(x0) && i<=length(x1)+length(x0)
        plot(x(1:length(x0)),y(1:length(x0)),'r');
        plot(x(length(x0):i),y(length(x0):i),'g');
    elseif i>length(x1)+length(x0) &&
 i<=length(x2)+length(x1)+length(x0)
        plot(x(1:length(x0)),y(1:length(x0)),'r');
        plot(x(length(x0):(length(x0)+length(x1))),y(length(x0):
(length(x0)+length(x1))),'g');

 plot(x((length(x0)+length(x1)):i),y((length(x0)+length(x1)):i),'k');
    elseif i>length(x2)+length(x1)+length(x0) &&
 i<=length(x3)+length(x2)+length(x1)+length(x0)
        plot(x(1:length(x0)),y(1:length(x0)),'r');
        plot(x(length(x0):(length(x0)+length(x1))),y(length(x0):
(length(x0)+length(x1))),'g');
        plot(x((length(x0)+length(x1)):
(length(x0)+length(x1)+length(x2))),y((length(x0)+length(x1)):
(length(x0)+length(x1)+length(x2))),'k');

 plot(x((length(x0)+length(x1)+length(x2)):i),y((length(x0)+length(x1)+length(x2))
    elseif i>length(x3)+length(x2)+length(x1)+length(x0)
        plot(x(1:length(x0)),y(1:length(x0)),'r');
        plot(x(length(x0):(length(x0)+length(x1))),y(length(x0):
(length(x0)+length(x1))),'g');
        plot(x((length(x0)+length(x1)):
(length(x0)+length(x1)+length(x2))),y((length(x0)+length(x1)):
(length(x0)+length(x1)+length(x2))),'k');
        plot(x((length(x0)+length(x1)+length(x2)):
(length(x0)+length(x1)+length(x2)+length(x3))),y((length(x0)+length(x1)+length(x2)
(length(x0)+length(x1)+length(x2)+length(x3))),'b');

 plot(x((length(x0)+length(x1)+length(x2)+length(x3)):i),y((length(x0)+length(x1)+
    end
    %plot(x(1:i),y(1:i));
    %plot(x2(1:i),y2(1:i));
    plot(x(i),y(i), 'r.', 'MarkerSize', 10);
    hold off;
    axis([-10^7 2*10^6 -6*10^6 6*10^6])
    daspect([1 1 1]);
    set(gcf, 'Position', [100 100 1000 1000]);
    set(gca, 'Xdir', 'Reverse')
    set(gca, 'Ydir', 'Reverse')
    xlabel('Direction Parallel to Insertion Orbit Apse Line (km)');
    set(gca,'FontSize',16);
    ylabel('Direction Perpendicular to Insertion Orbit Apse Line and
 In Saturn Equatorial Plane (km)');
    set(gca,'FontSize',16);
    drawnow;
    %hold on;
    %pause(0.05)
end
```

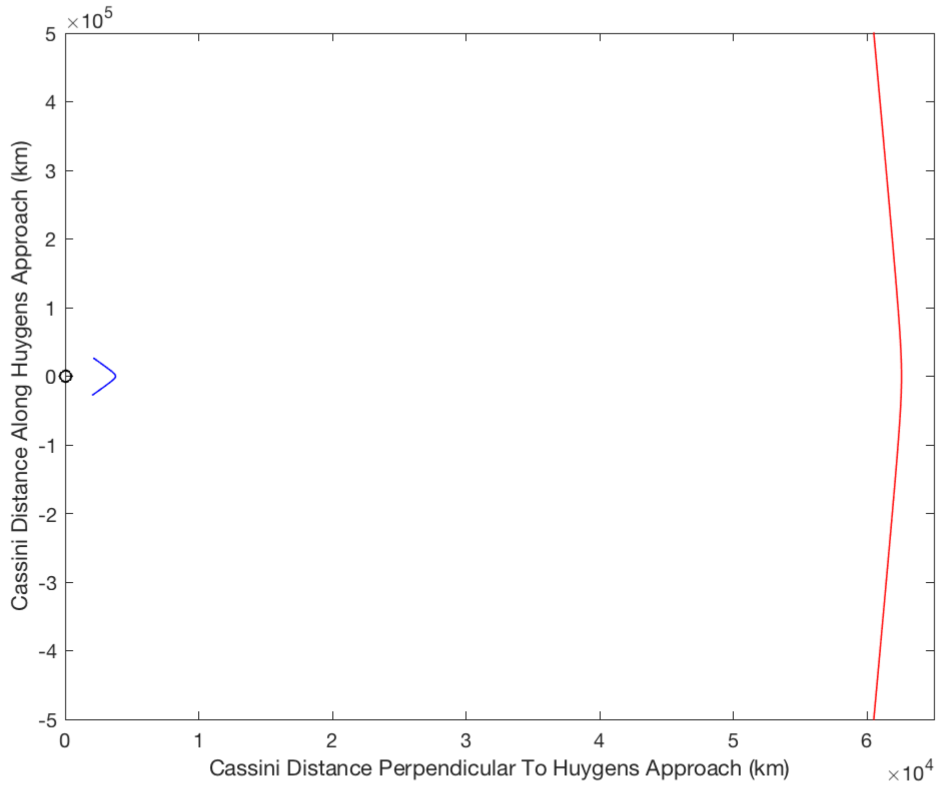


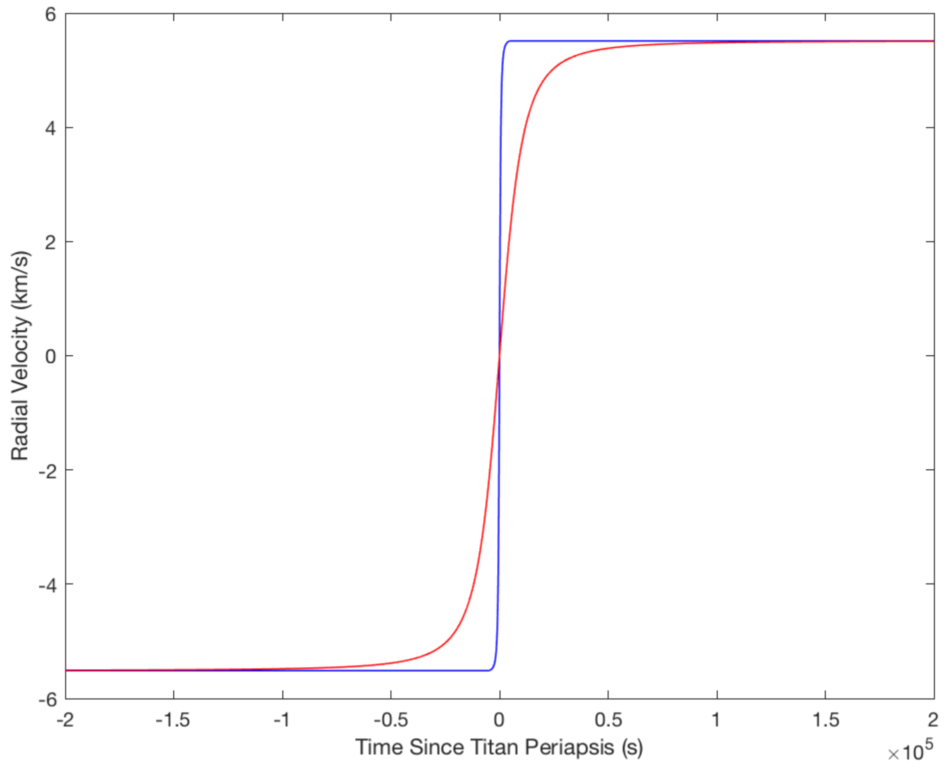
Differing Huygens Approaches

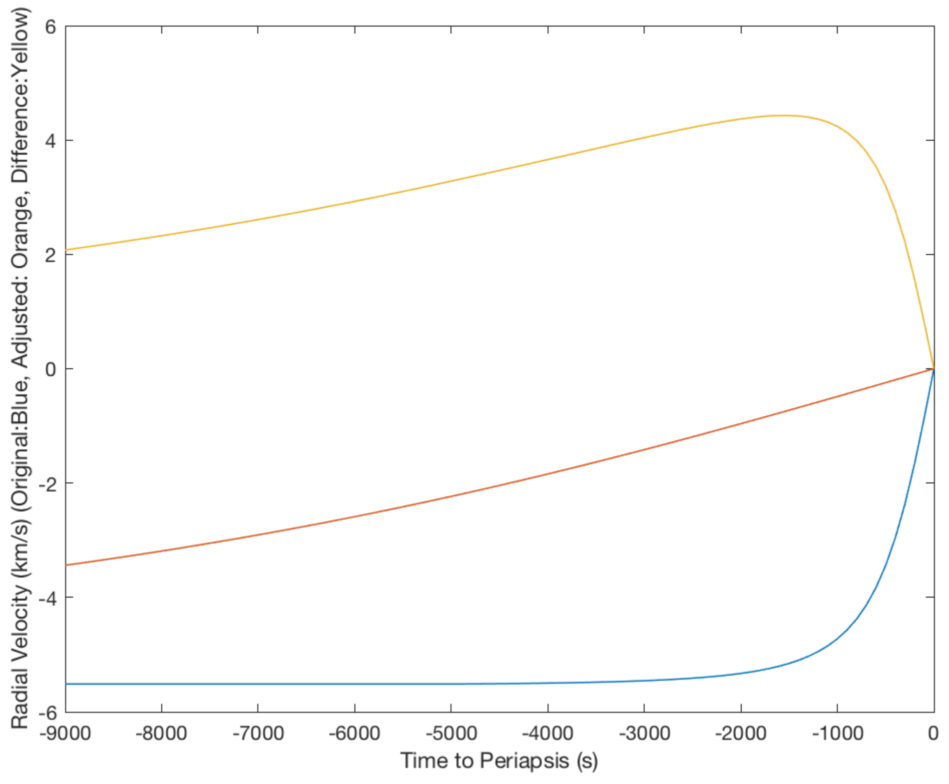
```
%approaching with inbound velocity equal to outbound velocity at
 beginning
%of orbit segment 4
%v_infavgT1=6.1;
%v_infavgT3=5.4;
v_infavg=v_infavg2;
%define various periapses
rpT1=3775; %original planned periapsis
%rpT2=11000;
rpT3=62575; %actual, modified periapsis
h1=rpT1*sqrt(v_infavg^2+(2*mu_T)/rpT1);
%h2=rpT2*sqrt(v_infavg^2+(2*mu_T)/rpT2);
h3=rpT3*sqrt(v_infavg^2+(2*mu_T)/rpT3);
e1=1+(rpT1*v_infavg^2)/mu_T;
%e2=1+(rpT2*v_infavg^2)/mu_T;
```

```
e3=1+(rpT3*v_infavg^2)/mu_T;
thetainf1=-asin((v_infavg*h1)/(mu_T*e1));
%thetainf2=-asin((v_infavg*h2)/(mu_T*e2));
thetainf3=-asin((v_infavg*h3)/(mu_T*e3));
theta1T=thetainf1:0.01:-thetainf1;
%theta2T=thetainf2:0.01:-thetainf2;
theta3T=thetainf3:0.01:-thetainf3;

rT1=(h1^2/mu_T)./(1+e1*cos(theta1T));
%rT2=(h2^2/mu_T)./(1+e2*cos(theta2T));
rT3=(h3^2/mu_T)./(1+e3*cos(theta3T));

[x1T,y1T]=pol2cart(theta1T,rT1);
%[x2T,y2T]=pol2cart(theta2T,rT2);
[x3T,y3T]=pol2cart(theta3T,rT3);
figure(2);
for i=1:length(x3T)
    plot(0,0,'ko');
    hold on;
    if i<=length(x1T)
        plot(x1T(1:i),y1T(1:i),'b',x3T(1:i),y3T(1:i),'r');
    %elseif i<=length(x2T)

  %plot(x1T(1:end),y1T(1:end),'b',x2T(1:i),y2T(1:i),'b',x3T(1:i),y3T(1:i),'r');
    else
        plot(x1T(1:end),y1T(1:end),'b',x3T(1:i),y3T(1:i),'r');
    end
    axis([-1 6.5*10^4 -5*10^5 5*10^5]);
    ylabel('Cassini Distance Along Huygens Approach (km)');
    xlabel('Cassini Distance Perpendicular To Huygens Approach (km)');
    drawnow;
end

vrT1=(mu_T/h1)*e1.*sin(theta1T);
%vrT2=(mu_T/h2)*e2.*sin(theta2T);
vrT3=(mu_T/h3)*e3.*sin(theta3T);
%vperpT1=(mu_T/h1)*(1+e1.*cos(theta1T));
% figure(3);
% plot(theta1T,vrT1,'b',theta3T,vrT3,'r');
% xlabel('True Anomaly (rad)');
% ylabel('Radial Velocity (km/s)');
F1T=2*atanh(sqrt((e1-1)/(e1+1)).*tan(theta1T/2));
Mh1T=e1.*sinh(F1T)-F1T;
nh1T=(h1^3/mu_T^2)*(e1^2-1)^(-3/2);
t1T=Mh1T.*nh1T;
F3T=2*atanh(sqrt((e3-1)/(e3+1)).*tan(theta3T/2));
Mh3T=e3.*sinh(F3T)-F3T;
nh3T=(h3^3/mu_T^2)*(e3^2-1)^(-3/2);
t3T=Mh3T.*nh3T;
figure(4);
plot([t3T(1) t1T t3T(end)],[vrT1(1) vrT1 vrT1(end)],'b',t3T,vrT3,'r');
axis([-0.2*10^6 0.2*10^6 -6 6]);
xlabel('Time Since Titan Periapsis (s)');
ylabel('Radial Velocity (km/s)');
```

```
t=-3600*2.5:100:0; %time period from 2.5 hrs until periapse up to
 periapse
vrT3interp=interp1(t3T,vrT3,t);
vrT1interp=interp1([t3T(1) t1T t3T(end)],[vrT1(1) vrT1 vrT1(end)],t);
figure(5)
plot(t,vrT1interp,t,vrT3interp);
hold on;
%find average difference in radial velocity over course of 2.5 hrs
 into
%periapse
radialvelocitydifference=vrT3interp-vrT1interp;
plot(t, radialvelocitydifference);
xlabel('Time to Periapsis (s)');
ylabel('Radial Velocity (km/s) (Original:Blue, Adjusted: Orange,
 Difference:Yellow)');
averagevelocitysaved=mean(radialvelocitydifference);
```




Radial Velocity (km/s)
Time Since Titan Periapsis (s)
$\times 10^5$


Radial Velocity (km/s) (Original:Blue, Adjusted: Orange, Difference:Yellow)
Time to Periapsis (s)

Published with MATLAB® R2017a

```
%clear all;
close all;
clc;
%Grand Finale
G = 6.674e-11;
mtitan = 1.3452e23;
atitan = 1221870;
rtitan = 2574.73;
msaturn = 5.6834e26;
muSaturn = 37931000;
muTitan = G*mtitan / 1e9;
muEarth=398600;
rsaturn = 60270;
SOIsaturn = 54800000;
import = readtable("StateVectorResults (3).csv", 'HeaderLines', 12);
titan = readtable("Titan_Info.csv", 'HeaderLines', 12);

%% Final Orbit Plot Raw Data
%plot3(import.X_km_(1150:2100),import.Y_km_(1150:2100),import.Z_km_(1150:2100))
plot3(import.X_km_(21250:22204),import.Y_km_(21250:22204),import.Z_km_(21250:22204))
%plot3(import.X_km_(2100:3100),import.Y_km_(2100:3100),import.Z_km_(2100:3100))
%plot3(import.X_km_,import.Y_km_,import.Z_km_)
hold on;
plot3(titan.X_km_,titan.Y_km_,titan.Z_km_);
hold on;
[x,y,z] = sphere();
r = rsaturn;
surf( r*x, r*y, r*z )
axis equal

%% Delta V calculations
%Find orbital parameters in original orbit
%Then find delta V required to go to Grand Finale orbits
%Find delta V imparted by the atmosphere during the final 5 orbits to crash
%into  saturn

%% Interpolating data to get more accurate
%figure;
x=1:1:length(import.Distance_km_);
distance = import.Distance_km_(1:length(import.Distance_km_));
dx=0:1/(10):length(import.Distance_km_); % Resolution of 1 minute
distanceint = interp1(x,distance,dx,'spline');
speed1=import.Speed_km_s_;
speed = interp1(x,speed1,dx,'spline');
X_km = interp1(x,import.X_km_,dx,'spline');
Y_km = interp1(x,import.Y_km_,dx,'spline');
Z_km = interp1(x,import.Z_km_,dx,'spline');
dX = interp1(x,import.dX_dt_km_s_,dx,'spline');
dY = interp1(x,import.dY_dt_km_s_,dx,'spline');
dZ = interp1(x,import.dZ_dt_km_s_,dx,'spline');
%plot3(X_km(1:(1035*600)),Y_km(1:(1035*600)),Z_km(1:(1035*600)))
% R = [X_km; Y_km; Z_km];
% V = [dX; dY; dZ];
% [r,v,vr,H,h,i,N,n,Omega,omega,E,e,theta] = orb_elements(R,V,muSaturn);

%% Finding intial and final orbits from the given data
%min and max radii
%
```

```
[ra, indexra] = max(distanceint(1:(1035*10)));
[rp, indexrp] = min(distanceint(1:(1035*10)));

Ra = [X_km(100),Y_km(100),Z_km(100)];
Rp = [X_km(indexrp),Y_km(indexrp),Z_km(indexrp)];
% scatter3(X_km(indexra),Y_km(indexra),Z_km(indexra),30, 'red')
% hold on;
% scatter3(X_km(indexrp),Y_km(indexrp),Z_km(indexrp),30, 'blue')
ra = norm(Ra);
rp = norm(Rp);

Va = [dX(100),dY(100),dZ(100)];
va = norm(Va);

Vp = [dX(indexrp),dY(indexrp),dZ(indexrp)];
vp = norm(Vp);

[ra,va,vra,Ha,ha,ia,Na,na,Omegaa,omegaa,Ea,ea,thetaa] = orb_elements(Ra,Va,muSaturn)
[rp,vp,vrp,Hp,hp,ip,Np,np,Omegap,omegap,Ep,ep,thetap] = orb_elements(Rp,Vp,muSaturn);

%desired orbit
[ra1, indexra1] = max(distanceint(500:2500));
[rp1, indexrp1] = min(distanceint(1:2100*10));
scatter3(X_km(indexra1),Y_km(indexra1),Z_km(indexra1),30, 'red')
hold on;
scatter3(X_km(indexrp1),Y_km(indexrp1),Z_km(indexrp1),30, 'blue')
va1 = [dX(indexra1),dY(indexra1),dZ(indexra1)];
vp1 = [dX(indexrp1),dY(indexrp1),dZ(indexrp1)];
Va1 = [dX(1500*10),dY(1500*10),dZ(1500*10)];
Ra1 = [X_km(1500*10),Y_km(1500*10),Z_km(1500*10)];

 [ra1,va1,vra1,Ha1,ha1,ia1,Na1,na1,Omegaa1,omegaa1,Ea1,ea1,thetaa1] = orb_elements(Ra1,↙
Va1,muSaturn)
 rp5 = ha1^2 / muSaturn * (1/(1+ea1))
[rp,vp,vrp,Hp,hp,ip,Np,np,Omegap,omegap,Ep,ep,thetap] = orb_elements(Rp,Vp,muSaturn);

%% delta v with no grav assist

hxfer = sqrt(2*muSaturn)*sqrt((ra1*rp)/(ra1+rp));
vxferp = hxfer/rp;
dvp = vxferp - vp;

h1 = sqrt(2*muSaturn)*sqrt((ra1*rp1)/(ra1+rp1));
va2 = h1/ra;
dva = va2-norm(va1);

dvtotal = abs(dvp + dva)
Isp = 310;
mass = 2150+3132/4;
deltam = mass * (1-exp(dvtotal*1000/(310 * 11)))
%Not possible to complete this manuever via a Hohmann transfer: not enough
%fuel

%% Finding DV needed to get into Grand Finale Orbits
% Titan's SOI
rsoiTitan = atitan * (mtitan/msaturn)^(2/5);
%Arbitrary departure: 4/21/17 0 UTC
% pages257 454
```

```
depindextitan = 9721;
depindexcas = 973*10;
i=depindextitan;
j=depindexcas;


Rcdep = [X_km(j),Y_km(j),Z_km(j)];
Vcdep = [dX(j),dY(j),dZ(j)];

% Finding arrival time

Rcarrive = [X_km(j),Y_km(j),Z_km(j)];
%Rarrive = [import.X_km_(j),import.Y_km_(j),import.Z_km_(j)];

Rtarrive = [titan.X_km_(i),titan.Y_km_(i),titan.Z_km_(i)];
dis = norm(Rcarrive - Rtarrive);
while dis > rsoiTitan
    i=i+1;
    j=j+1;
    %Rarrive = [import.X_km_(j),import.Y_km_(j),import.Z_km_(j)];
    Rcarrive = [X_km(j),Y_km(j),Z_km(j)];
    Rtarrive = [titan.X_km_(i),titan.Y_km_(i),titan.Z_km_(i)];
    dis = norm(Rcarrive - Rtarrive);
end
titan.UTCCalendarDate(i) %arrival date and time
%via wolframalpha...
dt = 1662; %minutes
Vtarrive = [titan.dX_dt_km_s_(i),titan.dY_dt_km_s_(i),titan.dZ_dt_km_s_(i)];
Vcarrive = [dX(j),dY(j),dZ(j)];

rtarrive = norm(Rtarrive);
rcdep = norm(Rcdep);

[rac,vac,vrac,Hac,hac,iac,Nac,nac,Omegaac,omegaac,Eac,eac,thetaac] = orb_elements↙
(Rcarrive-Rtarrive,Vcarrive-Vtarrive,muTitan);
turnangle = 2*asind(1/eac)
a = hac^2 / muTitan * (1/(eac^2-1));
rp = hac^2 / muTitan * (1/(1+eac));
flybyalt = rp -rtitan;
percentdiffalt = (abs(flybyalt - 979)/((flybyalt + 979)/2))*100
vp = sqrt(2* (muTitan / rp + muTitan/(2*a)));
percentdiffspeed = (abs(vp - 5.8)/((vp + 5.8)/2))*100

while dis < rsoiTitan
    i=i+1;
    j=j+1;
    %Rarrive = [import.X_km_(j),import.Y_km_(j),import.Z_km_(j)];
    Rcarrive = [X_km(j),Y_km(j),Z_km(j)];
    Rtarrive = [titan.X_km_(i),titan.Y_km_(i),titan.Z_km_(i)];
    dis = norm(Rcarrive - Rtarrive);
end
Vcdepart = [dX(j),dY(j),dZ(j)];
V = Vtarrive

vinf1 = V - Vcarrive

vinf2 = V - Vcdepart
```

```
dV = (vinf2) - (vinf1)
norm(dV)

%% Final Titan Flyby analysis
%Last orbit
R3=[import.X_km_(22000),import.Y_km_(22000),import.Z_km_(22000)];
V3 =[import.dX_dt_km_s_(22000),import.dY_dt_km_s_(22000),import.dZ_dt_km_s_(22000)];
[r3,v3,vr3,H3,h3,i3,N3,n3,Omega3,omega3,E3,e3,theta3] = orb_elements(R3,V3, muSaturn);
r3p = h3^2 / muSaturn * (1/(1+e3));

%Second to last orbit
%Va1 = [import.dX_dt_km_s_(21279),import.dY_dt_km_s_(21279),import.dZ_dt_km_s_(21279)];
%Ra1 = [import.X_km_(21279),import.Y_km_(21279),import.Z_km_(21279)];
[ra1,va1,vra1,Ha1,ha1,ia1,Na1,na1,Omegaa1,omegaa1,Ea1,ea1,thetaa1] = orb_elements(Ra1,↙
Va1,muSaturn)
rp5 = ha1^2 / muSaturn * (1/(1+ea1))
rp5-r3p
```

```
%Orbital Parameters
function[r,v,vr,H,h,i,N,n,Omega,omega,E,e,theta] = orb_elements(R,V,mu)
r = norm(R);
v = norm(V);
vr = dot(R,V)/r;
H = cross(R,V);
h = norm(H);
i = acosd(H(3)/h);
N = cross([0,0,1],H);
n = norm(N);
Omega = acosd(N(1)/n);
if N(2)<0
    Omega = 360 - Omega;
else
    Omega;
end
E = 1/mu * (cross(V,H) - mu*(R/r));
e = norm(E);
omega = acosd(dot(N/n,E/e));
if E(3)<0
    omega = 360 - omega;
else
    omega;
end

theta = acosd(dot(E/e,R/r));
if vr<0
    theta = 360 - theta;
else
    theta;
end
end
```