Using a Graduate Student Developed Trajectory Generation Program to Facilitate Undergraduate Spacecraft / Mission Capstone Design Projects

Mr. Martin James Brennan, University of Texas, Austin

Martin James Brennan developed a passion for Science and Mathematics at Mississippi State University (MSU), where he met his wife Holly. In December 2008, he received a Bachelor of Science degree in Aerospace Engineering with an emphasis in Astrodynamics, a Bachelor of Science degree in Physics, and a minor in Mathematics. He began his graduate career in Aerospace Engineering with a focus in Orbital Mechanics in January 2009 at the University of Texas at Austin, where he is pursuing his Ph.D. He enjoys flying radio controlled aircraft, tinkering with projects, and loving life with his wife.

Mr. Adam Wayne Nokes, The University of Texas at Austin

Adam Nokes is currently a doctoral student at the University of Texas and resides in Austin with his wife Travis and dog Motley. His educational experience includes a B.S. from Cornell University in Mechanical and Aerospace Engineering, an M.Eng. from Cornell in Engineering Management, and an M.S. from the University of Colorado at Boulder in Aerospace Engineering. His current research focus is trajectory optimization and mission design. Nokes is also a teaching assistant for the undergraduate spacecraft mission design course. He enjoys fishing, hiking and biking in the great outdoors any chance that he gets.

Dr. Wallace T. Fowler P.E., University of Texas, Austin

Wallace Fowler holds the Paul D. and Betty Robertson Meek Centennial Professorship in the Department of Aerospace Engineering and Engineering Mechanics at the University of Texas at Austin, where he has been on the faculty since 1965. His areas of teaching and research are dynamics, orbital mechanics, spacecraft and space mission design, and aircraft flight testing. He is the recipient of several teaching awards, including the AIAA/ASEE John Leland Atwood Award and the ASEE Fred Merryfield Design Award. He is a member of the University of Texas Academy of Distinguished Teachers. He is a fellow of both the AIAA and the ASEE. He served as president of ASEE of 2000-2001.

Using a Graduate Student Developed Trajectory Generation Program to Facilitate Undergraduate Spacecraft / Mission Capstone Design Projects

Introduction

A major stumbling block in undergraduate spacecraft/mission design projects is the early generation of plausible baseline trajectories for missions outside of Earth orbit. Many student teams rely almost totally on the literature for their baseline trajectories, and often do not progress very far beyond these trajectories in their final designs. The problem is that trajectory design for interplanetary missions has many options and senior aerospace engineering students have neither the experience nor the time to explore the options thoroughly.

Trajectory Design in the Classroom

Numerous trajectory design and optimization programs exist. However, none of them are well suited towards teaching trajectory design to undergraduate engineering students. Many FORTRAN-based technical software packages require students to learn a new computer language (current engineering undergrads are usually *not* familiar with FORTRAN) while other, off-the-shelf, software packages have hefty licensing fees. For several years the students in the capstone design course at The University of Texas at Austin have made use of a professional quality software program called MAnE, developed by Dr. Jerry Horsewood.[4] This program is not open-source, so no changes can be made by the user to its functioning or output — a factor that limits the flexibility of the tool in teaching situations. The need for developing a customizable and flexible solution became clear, and this software package is the result.

TRACT (TRAjectory Configuration Tool)

In 2010-2011, the lead author developed a robust preliminary trajectory generation program, TRACT, as his MS thesis project. The goal was to create a program that could be used by undergraduate students to explore the solution space for many interplanetary missions. This program is based on three dimensional conic trajectories and features deep space maneuvers, powered and unpowered planetary flybys, and JPL planetary ephemerides. The program uses sequential patched conics as the basis for rapid complex trajectory prototyping. Sequences of patched conic mission segments are “optimized” to obtain trajectories that are good approximations of converged n-body mission trajectories. In addition to being used to develop sufficiently accurate approximate trajectories for undergraduate capstone design missions, the program output can be used as an initial guess for high precision trajectory optimization programs that require “good” initial estimates to converge on the appropriate solution.

TRACT is a software tool developed in the MATLAB environment to facilitate construction and refinement of solar system mission trajectories by undergraduate aerospace engineers. The decision to use MATLAB for this tool was based on a number of factors. First, most undergraduate aerospace engineering students have already had extensive experience with this language. The plotting functions built into MATLAB provide three dimensional depictions of the trajectories, allowing the students to clearly visualize the entire mission. Additionally, MATLAB is easily accessible and provides an open source environment where students can experiment directly with the functioning of the program. Existing trajectory analysis packages tend to either focus on accuracy or focus on speed and ease of use, but none can bring all these factors together in a software package that our entire target audience can access, use, and adjust.

When aerospace engineering students at our university begin the senior capstone design project, they have already taken a spacecraft dynamics course and an applied orbital mechanics course. A major pedagogical strength of the TRACT program is the fact that most of the orbital trajectory processes employed are taught in these undergraduate classes. The processes include two body motion, impulsive maneuvers, and gravity assists. Most undergraduates will not be familiar with the theory behind the two optimization techniques that are built into MATLAB, as such techniques are most often taught in beginning graduate courses. However, since the code is open source, this allows interested undergraduates to delve further into the theory if they so desire.

The primary trajectory tool that undergraduates might not have seen is Lambert targeting. TRACT uses this procedure to generate conic trajectories. Lambert targeting is the most common tool used for designing realistic baseline ballistic trajectories in mission design. The key feature of Lambert targeting is that only two-body dynamics is assumed, which correlates well with the patched conic assumptions. This procedure uses two positions and the time of flight between those positions to fit a conic section trajectory connecting them. Each segment of the patched conic trajectory is determined by separate Lambert targeted solutions. (Refer to Vallado[11] for a complete discussion of Lambert's problem, and the resulting solutions developed by Gauss, Battin[1], and others.) TRACT uses Lambert targeting subroutines created by D'Souza[2], Izzo[7], and Lancaster and Blanchard.[8] Some Lambert solvers fail to converge when the departure and destination points are exactly 180 or 360 degrees apart. Other solvers cannot create multiple revolution solutions. By applying different Lambert solvers in different situations, the TRACT software program avoids these common difficulties.

In order to construct realistic and efficient interplanetary trajectories, TRACT requires the student teams to specify a mission architecture made up of common trajectory elements (Earth departure, coasting trajectory segments, unpowered and powered gravity assists, deep space maneuvers, insertion into a targeted orbit about a planet, etc.). This demands that the student teams develop mission concepts and then use TRACT to evaluate their suitability for their mission.

When evaluating mission architecture, the MATLAB optimizer attempts to find the best local minimum *near* the assumed arrival and departure dates. By incrementing the departure dates and times of flight forward and backward in time, the user can search for nearby local minima. This allows the mission designer to compare different trajectories and search for one that best meets mission requirements. This also lets the user compute the penalty of launching at times other than the local optimum. Combining these results with information about booster capability helps to determine the launch window for the mission. The optimization routine uses total ΔV (propulsive velocity change) as the main performance criteria to minimize, by varying times of flight. Comparing the total ΔV requirements of each local time optimized solution, the students can perform their own trade-offs of energy versus time.

When using the program, the student provides the initial mission architecture, (estimates of the departure date, maneuver dates, and arrival dates, as well as powered and/or unpowered flyby sequences) and this is used to create a Lambert targeted baseline mission. The program then uses one of the two MATLAB optimizing procedures to move the event dates, while allowing the possibility of inserting deep space maneuvers between flybys. The program also allows the student to add constraints to the problem, such as a minimum flyby radius or a maximum launch V_∞ (hyperbolic excess velocity). The result is a local optimum in the vicinity of the initial estimates that conforms to the given mission architecture (event scenario).

Several teams, working on separate missions, will use the TRACT program for development of their mission trajectories during any given semester. The graduate student who wrote the program (the first author) and the design class teaching assistant (the second author) create a sample trajectory based on one of the logical mission architectures for the chosen mission. The undergraduate students, working under the guidance of the TA, then use TRACT to search over a range of potential departure dates and total transit times associated with the initially chosen mission architecture. TRACT facilitates student understanding of the solution space by plotting the best candidate trajectories side by side, allowing the teams to select the most preferable trajectory (their baseline trajectory) for their chosen mission. Guidelines for baseline trajectory development are included in the tutorial for the students in the design teams who are responsible for trajectory generation. The current tutorial process for a new user takes about an hour. Each team is then given a copy of the program and must develop their own architectures to generate trajectories that accomplish their mission goals.

TRACT Outputs

A major advantage of TRACT is a suite of MATLAB plotting outputs that have been tailored to allow easy visualization of various aspects of the trajectory. The pedagogical goal of these displays is to help the students better understand the details of their solutions through accurate visualizations. A very important concept in interplanetary trajectory design is the gravity assist maneuver (GAM) performed during a planetary flyby. In planet centered coordinates, the unpowered flyby rotates the velocity vector of the spacecraft. In a powered flyby, both the

magnitude and direction of the velocity in planet centered coordinates can be changed. In heliocentric coordinates, the spacecraft velocity vector can be turned, increased, or decreased, depending on the geometry of the GAM and the powered maneuver. Every gravity assist maneuver must be carefully planned; to aid in this plots are generated that contain helpful vectors and different colored segments. Indicated on the plots are the incoming and outgoing velocity vectors of the spacecraft, the position of the planet during closest approach, and the turning angle achieved during the GAM. The incoming and outgoing hyperbolic asymptotes are displayed radiating from the planet and reinforce the ease of interpretation by the student.

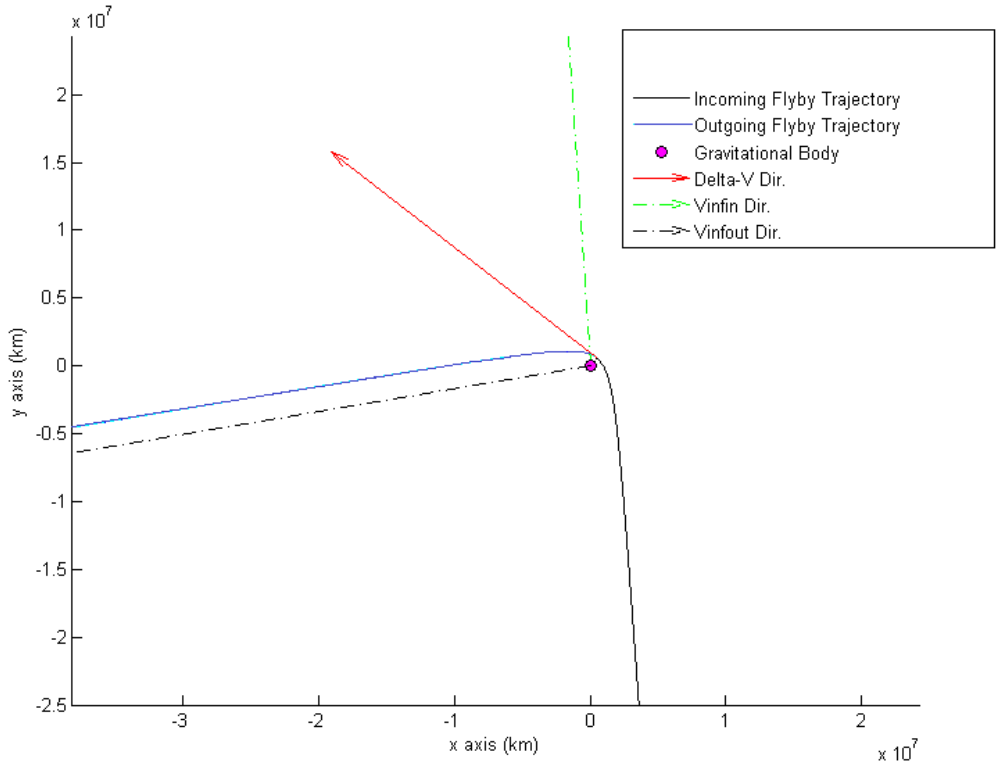

Figure 1: Gravity Assist Maneuver Example[2]

Figure 1, above, shows a sample GAM plot using planet centered coordinates. In the program, all gravity assists are assumed to be powered, and if an unpowered gravity assist is better, the optimization reduces the ΔV magnitude to a negligible value.[2] In the figure, the incoming and outgoing hyperbolic trajectories are shown in black and blue, respectively, with a red powered ΔV direction vector. In this example case, the ΔV direction vector is displayed, but its magnitude is negligibly small (< 1 m/s). Further iteration would likely reduce the ΔV magnitude to zero.

TRACT can also handle departure and arrival parking orbits. An example arrival trajectory and the associated arrival parking orbit are shown in a planet oriented frame (Figure 2). Hyperbolic asymptote vectors radiate away from the planet. The planetary velocity direction vector and a sphere of influence circle are also shown—to give the student references for understanding the

orientation of the trajectory. The ΔV vector shows the location and direction of the impulsive maneuver.

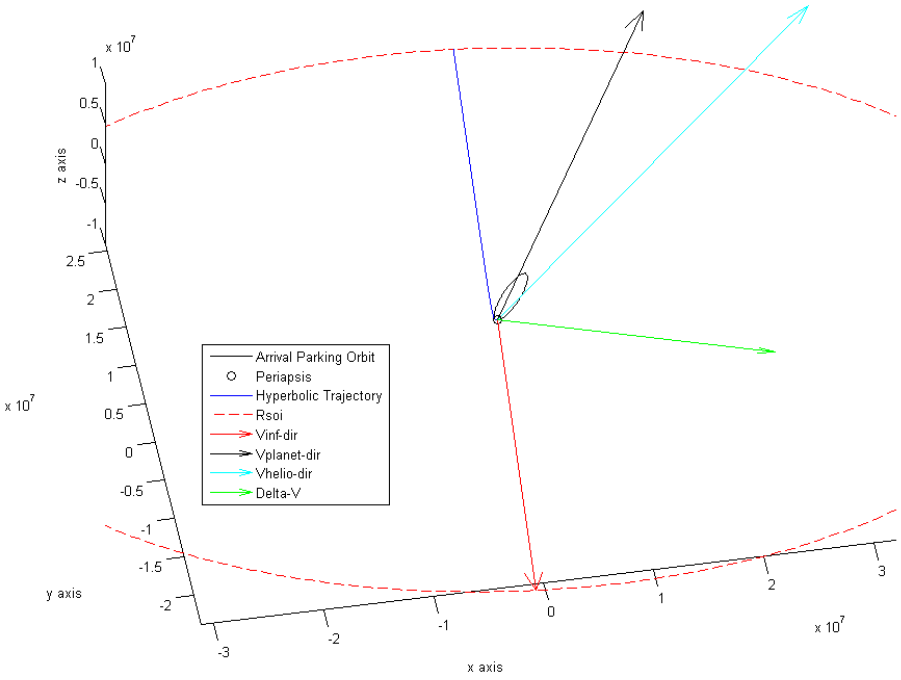

Figure 2: Highly eccentric arrival parking orbit example[2]

The spacecraft's velocity in heliocentric space, before the arrival, is in the direction of the cyan vector, and the final orbit about the planet is the small black ellipse. The planet's heliocentric motion is represented by the black vector. The V_∞ asymptote direction is shown by the red vector, while the nearly parallel arrival hyperbolic trajectory is blue. The ΔV direction is depicted by the green arrow. With the main parameter vectors and trajectory segments displayed in color, the figure perspective can be manipulated in all three dimensions to achieve the desired orientation.

TRACT Performance Verification

The usefulness of TRACT is derived from producing trajectories that are close enough to reality so that student designs, based on them, are realistic. Interplanetary spacecraft encounter numerous gravitational and perturbing forces throughout a mission, such as solar radiation pressure, atmospheric drag, and magnetic forces. Modeling such a complex set of forces requires numerical integration and detailed knowledge of the mission environment to determine a complete, and highly accurate, trajectory solution. The patched conic approximation utilizes analytical two-body equations to simplify the process and obtain a reasonably good approximation of the desired trajectory. Though the dynamics are approximated and simplified

by implementing the patched conic method, there is minimal loss of accuracy. As part of the development process, the program replicated the Voyager missions, the Cassini mission, the Ulysses mission, and the Juno mission in order determine how well its outputs approximate actual space mission trajectories. Results from the modeling of the Voyager 2 mission are discussed below.

Table 1 shows the replicated Voyager 2 mission trajectory parameter values versus the actual values. The replicated trajectory values have less than 5% deviation from the actual parameter values. In the table, TOF1 is the time from Earth to Jupiter, TOF2, from Jupiter to Saturn, etc. The flyby radii at each of the planets are the other values in the table. The flyby radius at Neptune was not computed since there was no stated post Neptune target.

Table 1: Comparison of actual Voyager 2 trajectory parameters to TRACT optimized values[2]

Parameter	Actual	Optimized	Error
Launch V_∞	10.147 km/s	9.642 km/s	4.9 %
Launch Date	2443376.2	2443383.1	6.9 days
TOF1	688 days	692.2 days	0.6 %
TOF2	778 days	784.2 days	0.7 %
TOF3	1,613 days	1,611.5 days	0.1 %
TOF4	1,309 days	1,310.2 days	0.1 %
Jupiter r_p	7.214×10^5 km	7.342×10^5 km	1.8 %
Saturn r_p	1.607×10^5 km	1.607×10^5 km	0.0 %
Uranus r_p	1.072×10^5 km	1.090×10^5 km	1.7 %
Neptune r_p	2.924×10^4 km	—	—

The performance of TRACT was also validated by comparison to actual trajectory values of the Cassini mission. Figures 3 and 4 below show the TRACT-produced trajectory for the Cassini mission. At the scale shown, the trajectory cannot be distinguished from the actual trajectory flown by Cassini. Figure 3 depicts the entire mission, beginning at Earth and ending at Saturn orbit insertion. Figure 4 is a close-up of the initial segments of the trajectory, where the locations of ΔVs 1 through 5 are displayed according to the following sequence of events: Earth departure, the first Venus flyby, a Deep Space Maneuver (DSM), the second Venus flyby, and the Earth flyby. Jupiter flyby and Saturn insertion, on Figure 4, are the ΔV_6 and ΔV_7 locations, respectively.

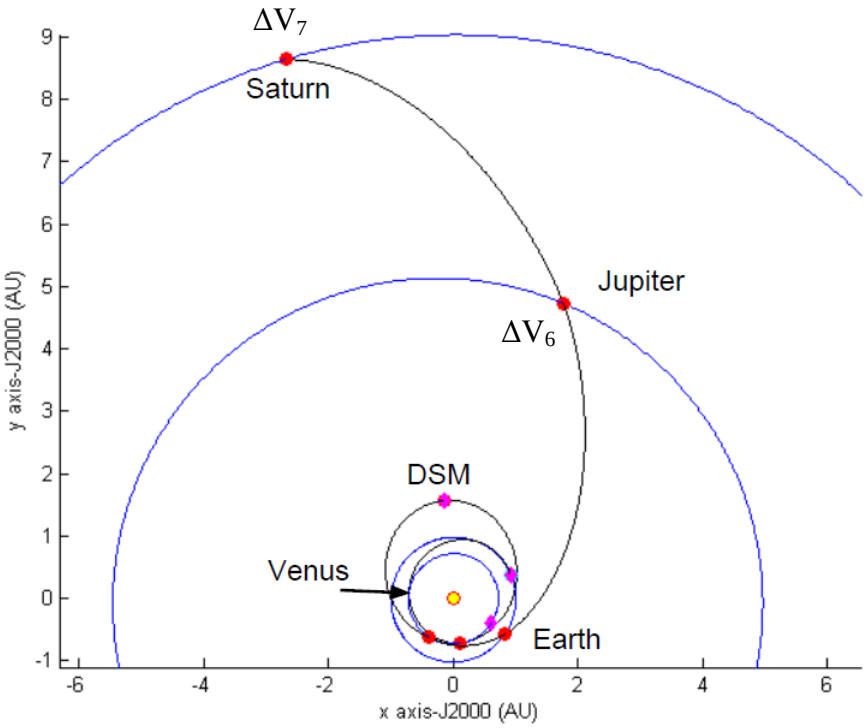

Figure 3: Cassini interplanetary trajectory profile[2]

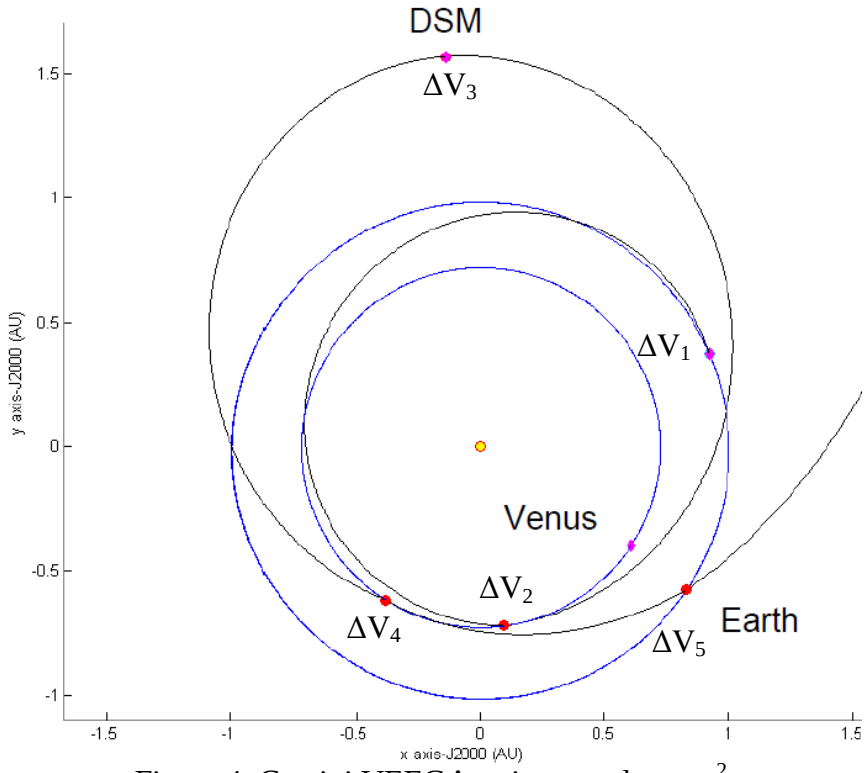

Figure 4: Cassini VEEGA trajectory close-up[2]

The validation goal was to evaluate how the computed trajectory values (flight times, ΔV values, and DSM locations) matched the true values. The actual mission values were used as initial

guesses for TRACT, and then the mission was optimized (using two optimization methods: numerical gradient and Nelder-Mead Simplex); see Table 2 below. In the table, TOF1 is for the Earth-Venus1 leg, TOF2 is for the Venus1-DSM leg, TOF3 is for the DSM-Venus2 leg, etc. The computed trajectory flight times and launch dates are quite close to the actual values. This indicates that the Lambert patched-conic model can converge on interplanetary trajectories that approximate high precision solutions closely enough for preliminary mission design.

Table 2: Comparison of Cassini trajectory flight times and dates[2]

Parameter	Actual	Optimized (gradient)	Optimized (simplex)
Iterations	—	25	1,475
Launch V_∞ (km/s)	4.074	4.053	4.073
Launch Date	2450736.9	2450736.9	2450736.9
TOF1 (days)	193.2	193.3	193.5
TOF2 (days)	220.7	220.7	220.6
TOF3 (days)	203.5	203.6	203.8
TOF4 (days)	54.4	54.5	54.2
TOF5 (days)	500.3	500.3	505.6
TOF6 (days)	1,278.9	1,278.9	1,290.4

Table 3, below, compares the TRACT-produced DSM and GAM values with those actually flown by Cassini. For this case, the MATLAB Nelder-Mead Simplex results are closer to the true mission values than the MATLAB gradient values. The difference in simplex optimized DSM location from the actual location is only 0.00254 AU (3.799×10^5 km), while the difference in gradient optimized DSM location from the actual is 0.0160 AU (2.397×10^6 km). The improvement in accuracy is an order of magnitude better for this study.

Table 3: Comparison of Cassini trajectory DSM and GAM values

Parameter	Actual	Optimized (gradient)	Optimized (simplex)
Venus1 r_p (km)	6,335.8	6,427.6	6,415.1
Venus2 r_p (km)	6,654.8	6,737.5	6,646.4
Earth r_p (km)	7,551.1	7,097.1	7,220.3
Jupiter r_p (km)	9,791,492	10,102,561	10,055,385
DSM x (AU)	-0.1385	-0.1227	-0.1381
DSM y (AU)	1.5722	1.570	1.5697
DSM z (AU)	0.0291	0.0276	0.0293

It should be noted that in dealing with deep space missions, the units we use can sometimes mask differences and similarities in solutions. This can be illustrated from the data in Table 3, above. If we compute the differences between the actual Cassini DSM location and the point determined by the optimizer, and then express the differences in units of kilometers, we get some interesting results. The point determined by the optimized gradient for the DSM is 2.40 million km away

from the actual point where the maneuver took place. The DSM point determined by the simplex optimizer is about 380 thousand km from the actual maneuver point. These position differences, when looked at in terms of kilometers, seem very large. However, in terms of percentage of the magnitude of heliocentric radius vector to the actual maneuver point, the differences are only 1.02% and 0.16%, respectively.

The impulsive maneuver magnitudes determined from the Nelder-Mead Simplex method closely match the ΔV values for all gravity assists and mirrors the actual DSM and orbit insertion values, as shown in Table 4. The benefits between the gradient or simplex methods vary from mission to mission. For this reason, neither optimization method is deemed better than the other and must be investigated on a case by case basis. Yet, when used in combination, these methods strengthen each other's progress and improve the quality of trajectories explored by the student.[2]

Table 4: Comparison of Cassini trajectory ΔV maneuver results[2]

Maneuver	Actual (km/s)	Baseline (km/s)	Optimized (km/s) (gradient)	Optimized (km/s) (simplex)
ΔV_1	0	0.009	0	0
ΔV_2	0	0.166	0.012	4.26×10^{-11}
ΔV_3	0.450	0.478	0.515	0.438
ΔV_4	0	0.123	0.002	1.97×10^{-8}
ΔV_5	0	0.394	1.141×10^{-7}	1.03×10^{-11}
ΔV_6	0	0.286	0.150	1.11×10^{-11}
ΔV_7	0.622	0.601	0.601	0.598

The gravity assists in the Cassini trajectory provide a change in heliocentric velocity (ΔV) and boost the energy of the spacecraft without requiring any propellant. This effective ΔV is quite large, as can be seen in Table 5, and essential in making the Cassini mission feasible. The optimized values, produced by TRACT, are a good approximation for the measured values of Cassini's effective ΔV increase due to the flyby.

Table 5: Comparison of Cassini gravity assists effective ΔV gain

Gravity Assist	Actual Δv_{eff} (km/s)	Gradient Δv_{eff} (km/s)	Simplex Δv_{eff} (km/s)
Venus 1	6.99	6.98	7.02
Venus 2	6.69	6.70	6.71
Earth	5.45	5.84	5.70
Jupiter	2.21	1.97	2.07

Case study: Mission to Uranus

During the fall 2012 semester, TRACT was used by undergraduates in the capstone design course to assist in trajectory design. This was a pilot semester for the tool, and it is worth reporting on the impact it had on the team and their project.

The trajectory design from the Uranus group required a novel approach. A strictly ballistic Earth-Uranus trajectory was not feasible due to the combination of a fourteen year limit on the chosen power supply and the limited capacity of available boosters. The mission had to include one or more gravity assist maneuvers and possibly, deep space maneuvers. The key to moving forward with the design was the determination of a feasible baseline trajectory, as this factored strongly into the choice of the launch vehicle and the payload delivered to the target planet.

Starting with the requirement to keep the mission duration below the expected lifetime of the Advanced Stirling Radioisotope Generators (ASRGs) that the design team planned to implement (~14 years), it was clear that a simple Hohmann-like transfer trajectory would take too long (~16 years). To bracket the departure window, the project team identified a need to schedule launch after 2025, in order to provide sufficient time for the development of the spacecraft, and before 2040, to make the application of current technologies relevant. Next, an evaluation of fly-by opportunities was made by the students. They determined that Saturn would not be properly aligned for a gravity assist maneuver on the way to Uranus. Jupiter, however, would be in a good location for a fly-by in the 2034-2037 time range.

Students, with the help of the TA, identified ballpark transit times from Earth to Jupiter and Jupiter to Uranus to use as a starting point. To explore the solution space, they iterated across multiple departure dates, while keeping the times of flight constant, and across multiple times of flight, while keeping the departure dates constant. Some of the trajectories resulting from this preliminary analysis are shown below (in units of AU's):

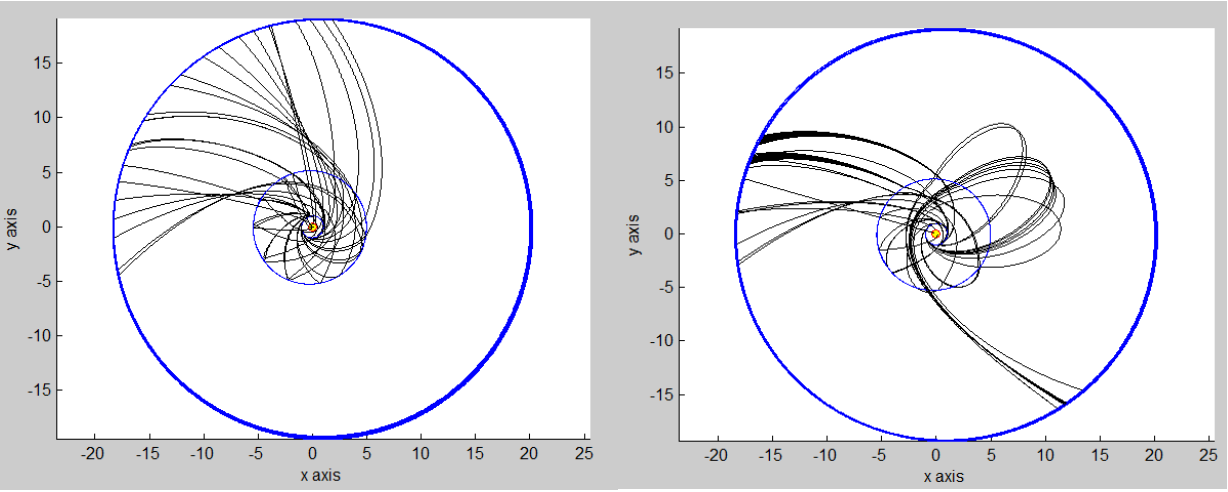

Figure 5: Results from preliminary investigation of possible Earth-Jupiter-Uranus trajectories

In these plots, the orbit of Uranus is the outer circle and the next inner orbit is that of Jupiter. The inner circle (obscured by the trajectories) is that of Earth. After experimenting with many options, the Uranus team identified the 2035-2036 departure time as the most ideal. The plots below show results from the last phase of their trajectory search.

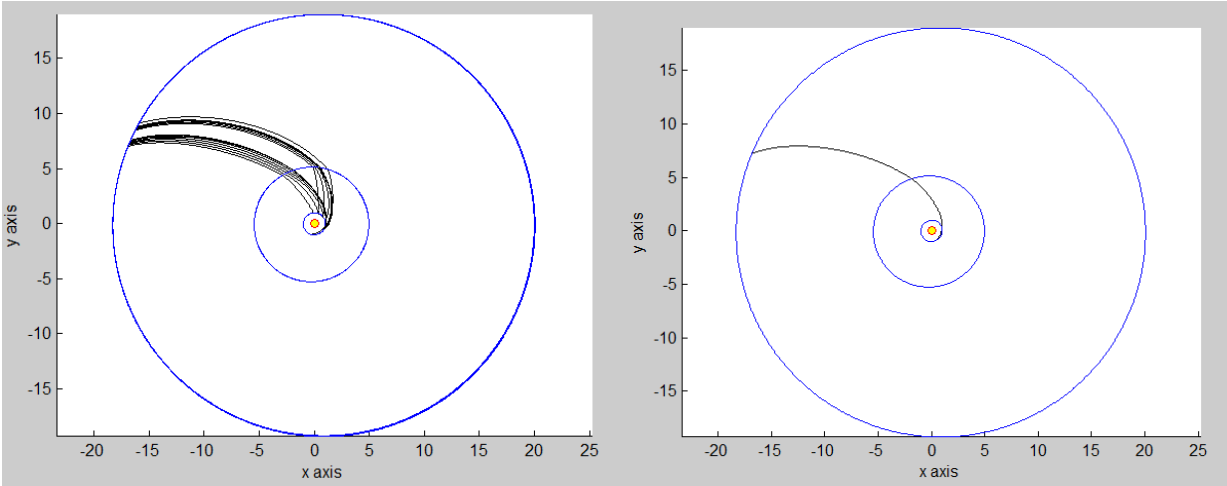

Figure 6: Narrowing down to an optimal solution for Earth-Jupiter-Uranus trajectory

The plot on the right represents their final trajectory solution. By this process, the Uranus mission team was able to determine a good mission trajectory and select a booster two weeks before the mid-semester review. This enabled them to have good estimates of the total propellant required for the mission and, therefore, the mass that they would be able to deliver to Uranus. Having the trajectory and available mass tied down early, the team was able to concentrate on the design of the spacecraft and its subsystems for the last half of the semester. The result was that the overall design was considerably more complete and detailed than those of earlier semesters. The science, communications, power, structure, and thermal aspects of the design were mature much earlier than in previous semesters.

Another benefit from quickly developing a trajectory was having time to learn from the design experience and understand the solution generated by TRACT. A prime example of this is shown in the right-hand part of Figure 6 above. On first inspection, the Jupiter flyby appears to have had little effect on the trajectory. However, on closer inspection, the students learned that the Jupiter flyby produced an increase of 5.51 km/s. They learned that a significant gravity assist can occur with or without a substantial change in the direction of the heliocentric velocity. The dynamics of the gravity assist about Jupiter become clearer by plotting the flyby in planet centered coordinates, as seen in Figure 8.

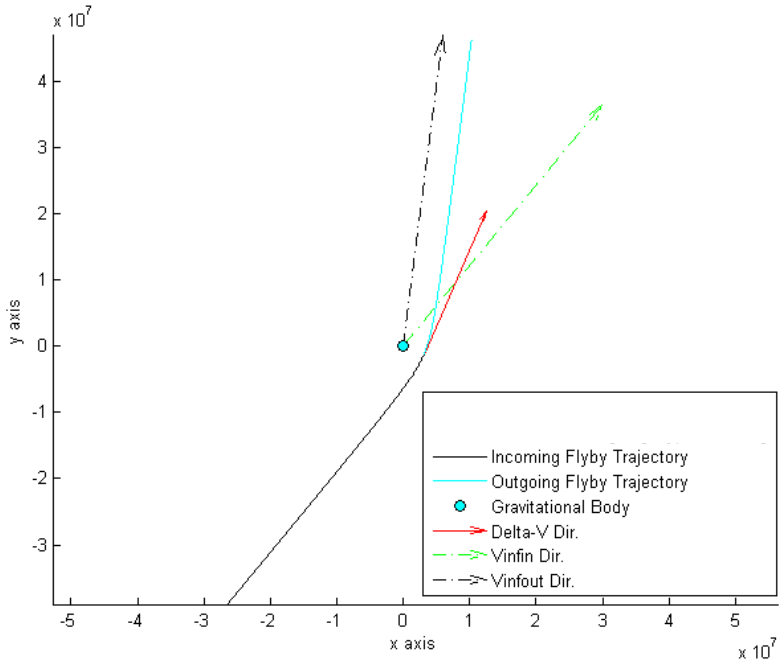

Figure 8: Close up of Jupiter GAM with V_∞ asymptote vectors displayed

Overall, the development of a good trajectory model early in the design process is crucial in spacecraft/mission design. This is especially true in a one-semester capstone design course at a university. The launch vehicle choice, the payload mass, the maneuver sequence, and the mission duration are all intimately tied to the trajectory flown. Getting a good trajectory early is a key to the development of better and more realistic designs–which, in turn, provides them with a better overall design experience.

Evaluation of Student Usage of TRACT

It is difficult to assess the effect of the introduction of a new feature into a learning process in situations where no control group is possible. That is the situation that we faced when introducing TRACT into the design class. Instead of having data from parallel efforts, one using TRACT and one not using it, we have to rely on TA and instructor observations and student comments. The following is a summary of the feedback.

During the spring 2013 semester, three student teams are making strong usage of TRACT. The projects are: (1) a mission to Jupiter's icy moon Europa involving an Earth escape and Earth gravity assist, (2) a mission to Saturn's largest moon Titan that involves a high energy Earth escape, a Jupiter flyby, and orbits around Saturn and Titan; and (3) a mission to a hypothetical space station 60 degrees ahead of Earth in its orbit around the Sun (at the Earth-Sun L4 Lagrange point).

We asked the students about difficulties in learning TRACT. They reported that it took 3 to 4 hours to learn to use TRACT without instruction. Those who sat down with the author of

TRACT (the first author) were very comfortable with TRACT after only two hours. All students recommended that a tutorial be developed and that it would shorten the time between first use and first usable trajectories. One of the students suggested that we develop an online video as the tutorial.

Each of the users reported that TRACT allowed them to better understand the various scenario options that were available to them. This allowed them to focus quickly on the details of the mission itself, rather than spending weeks on developing a program that "might" help them to characterize the mission. The following quote from one of the students is instructive: "I have a real appreciation for trajectory planning now that I have had some exposure to the many variables that can affect a trajectory and how easily it changes with small variations. This is especially true considering there typically isn't a whole lot of room for error with mission trajectory planning."

Student team members who developed trajectories were active much sooner in the semester than before TRACT was available. We told them their jobs were crucial for their teams and the rest of the team members were waiting on them to get initial maneuver (velocity change) estimates. They worked with their teams to develop concepts of operations and then used TRACT to determine the viability of those concepts. The teams were working with viable trajectories within three to four weeks of the start of the design effort. In earlier semesters, most teams used small variations of heritage trajectories, because they did not have the time/ability to model new mission scenarios. For those earlier teams that did attempt to model trajectories of their own mission scenarios, the process usually took eight to twelve weeks to reach the an acceptable level of sophistication. Overall, the student response to TRACT has been very positive. For the current semester (Spring 2013) we expect the end result to be projects that have more detailed and complete designs than before.

Conclusion

As with any active software package, improvements are always forthcoming. An additional benefit from the open-source nature of MATLAB is that the student users can become sources of upgrades. After the project's pilot semester, in the Fall of 2012, several improvements and additions were suggested to aid in the applicability to the mission design classroom. These include: adding low-thrust trajectory capability that functions in a two-body environment, creating additional optimization schemes that solve for near-global optima over a large range of departure dates, and a GUI that allows for a large number of total events. The impact of TRACT on undergraduate spacecraft mission design reports has been substantial. Benefits to the students include: faster acquisition of baseline trajectories, less time spent iterating through slightly better trajectories to find the local optimal solution, less project churn due to unrealistic launch windows, and perhaps most importantly, an increased familiarity with the underlying theory involved in trajectory design. Another benefit that is harder to quantify is the positive effect that having an early detailed mission trajectory has on the rest of the design team. By knowing the

mission trajectory, the planners can define a launch window, choose a booster, choose a staging scenario, and provide the rest of the team with a maximum available payload mass estimate. In this manner, TRACT has been observed to substantially aid in the progress of undergraduate capstone mission design projects. TRACT is unique in the field of trajectory design: it is appropriately accurate, executes rapidly, and since it is written in MATLAB, it allows interested students to customize *any* aspect of the program.

Ongoing Program Development

The TRACT Graphical User Interface (GUI) is still under development. The MATLAB GUI package offers clickable buttons, drop down menus and data input methods that will allow students to quickly and easily explore many candidate trajectories for their mission. The GUI for TRACT is being developed specifically to facilitate use by undergraduates designing interplanetary spacecraft missions. It will not only offer an easier method of learning to use TRACT, but will also provide a better understanding of resulting trajectories. Once we have completed two additional semesters of testing and development, we will offer TRACT to the aerospace engineering educational community for student use.

Bibliography

1. Battin R.H. *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition*. American Institute of Aeronautics and Astronautics, Virginia, 1999.
2. Brennan, Martin James. *Patched Conic Interplanetary Trajectory Design Tool.* (Master's thesis). Retrieved from University of Texas at Austin; Theses and Dissertations database. December 2011. (http://hdl.handle.net/2152/ETD-UT-2011-124849).
3. D'Souza C.D. *lamBattin*, (vers. January 20, 1989). Developed in the MATLAB environment.
4. Gooding, R.H. A procedure for the solution of Lambert's orbital boundary-value problem. *Celestial Mechanics and Dynamical Astronomy*, 48:145–165, 1990.
5. Horsewood, J.L. MAnE, The Mission Analysis Environment (vers. 3.5) and attached documentation.
6. Izzo D. Lambert's problem for exponential sinusoids. *Journal of Guidance, Control and Dynamics*, 29:1242-1245, September 2006.
7. Izzo D., Becerra V.M., Myatt D.R., Nasuto S.J., and Bishop J.M. Search space pruning and global optimization of multiple gravity assist trajectories. *Journal of Global Optimization*, 38:283-296, 2007.
8. Izzo, D. ESA Advanced Concepts team. Code used available in MGA.M, on http://www.esa.int/gsp/ACT/inf/op/globopt.htm. Last retrieved Nov, 2009.
9. Oldenhuis R.P.S. *Robust Solver for Lambert's Orbital-Boundary Value Problem*. Internet. April 2011, MATLAB File Exchange: http://www.mathworks.com/matlabcentral/fileexchange/26348-robustsolver-for-lamberts-orbital-boundary-value-problem.

10. Lancaster E.R. and Blanchard R.C. A unified form of Lambert's theorem. *NASA technical note* TN D-5368, 1969.
11. Oldenhuis R.P.S. Trajectory optimization for a mission to the solar bow shock and minor planets. Master's thesis, Delft University of Technology, 2010.
12. Petropoulos A.E. and Longuski J.M. Shape-based algorithm for automated design of low-thrust, gravity-assist trajectories. *Journal of Spacecraft and Rockets*, 41:787-796, 2004.
13. Vallado D.A. *Fundamentals of Astrodynamics and Applications*. Microcosm Press and Kluwer Academic Publishers, second edition, 2001.