

# Autonomous Navigation System for Spacecraft Using Low-Thrust Trajectories

Erika A. DeBenedictis\*

*California Institute of Technology, Pasadena, California 91126*

DOI: 10.2514/1.B34103

**This research describes a prototype software navigation system that would allow a spacecraft with a small amount of continuous propulsion to navigate low-energy trajectories. First, the desired route is described in terms of basic orbit shapes, such as Lyapunov orbits. This sequence of orbit shapes is converted into an itinerary of spatial boundaries that a spacecraft executing the low-energy maneuver will cross in order. A software system then employs a guided optimization algorithm that identifies the thrust angle that will maintain the desired orbit. Using this software as a research tool, simulations have identified low-energy paths that could be used by a spacecraft with an ion drive to perform a Venus flyby within four or five years of its launch from Earth. This approach makes it possible to identify complex low-energy trajectories that rely on the gravitational effects of different two-body systems (for instance, Earth–moon and Earth–sun) and to study the utility of continuous propulsion in flying such trajectories from Earth.**

## I. Introduction

**T**HE Interplanetary Superhighway (IPS) is a concept in spacecraft navigation where a spacecraft can travel among planets and moons with arbitrarily small amounts of propulsion, i.e., no more than stationkeeping thrust. This research describes an autonomous spacecraft navigation system for a low-thrust spacecraft traveling the IPS.

The IPS was developed to explain observations of asteroids in surprising orbits that resulted from multibody effects [1]. While attaining these orbits is improbable for asteroids that truly have no propulsion, a spacecraft with a thruster only strong enough to overcome the unpredictable effects of solar wind, navigational error, and computational error could choose to fly orbits that asteroids only appear in by chance.

There are several existing types of low-energy orbit planning, each of which focuses on navigating a different area of the solar system. The first type investigates navigating the area near Lagrange points and planets, using differential equations to define manifolds created by the paths of unpropelled particles as they move through the region and are affected by gravity [2–4]. This type of planning is most closely related to this project as it focuses on the same area of space and uses similar orbit shapes. In this project, these orbits have been used to navigate near the Earth and then enter interplanetary space. Another type of low-energy orbit planning focuses on systems with multiple bodies in resonant orbits. This approach creates a kick function to characterize how much a planetary body will affect another objects' orbit depending on how closely they pass each other. Although this works best in a system like the moons of Jupiter, where orbit periods are relatively small and there are many moons with which to interact, the studies of Ross and Grover [5] reveal principles that are useful when low-energy spacecraft initially enter interplanetary space. Their method may be used to adjust the orbit of a spacecraft so that it can reach another planet without propulsion, although this method alone would take a prohibitively long amount of time. A third type of low-energy orbit planning focuses on navigating interplanetary space using gravity-assist maneuvers with several planets. Studies in this area show that it is possible to navigate

the entire solar system through a network of gravity-assist maneuvers at different energy levels. Strange and Longuski demonstrate the general principles [6], and Petropoulos et al. show how to optimize these gravity assists [7]. Since navigation with gravity assists allows for relatively easy low-energy access to the solar system once a spacecraft can reach another planet, this project focuses on planning the low-energy orbits near Earth and connecting these orbits to the larger interplanetary gravity-assist network.

There are several instances where spacecraft have already used low-energy orbits to great effect. The Genesis spacecraft used a series of Earth–sun  $L_1$  and  $L_2$  halo orbits to collect solar wind samples [1,8]. Another example is the Hiten, a Japanese spacecraft designed to relay signals for the Hagaromo, a smaller spacecraft that would detach from the Hiten and orbit the moon. After the Hagaromo failed, a low-energy transfer to the moon was executed, allowing the Hiten itself to gain moon orbit, even though it had 10% less fuel than was believed to have been needed,<sup>†</sup> making the mission a success. The most famous use of low-energy maneuvers are the slingshot gravity assists used by Mariner 10, Pioneers 10 and 11, and Voyagers 1 and 2.

The mission lifetime for a spacecraft with a powerful rocket engine is limited by the propellant that can be carried. A spacecraft using the IPS would only need a thruster strong enough to overcome error. Such a level of thrust is low enough that a spacecraft's operation time could plausibly be limited by parts wearing out due to aging or radiation damage rather than by fuel limitation. This could expand the life expectancy of spacecraft by an order of magnitude. After receiving commands for a 100-year-long mission, the only relevant action of the navigation system on the proposed spacecraft will be to specify the direction to point the thruster (or perhaps turn it off) at a given time. Here, we present an approach for this navigation system and a limited demonstration with planar simulation. The demonstrated navigation system could plausibly control a spacecraft autonomously, but it has been demonstrated only for certain classes of orbits.

We also demonstrate the connection between the IPS and low-thrust trajectories. Spacecraft that have a thruster slightly more powerful than required to navigate the IPS can use this additional thrust to great effect. While this spacecraft would appear to follow the IPS in the short term, it could choose to point its small amount of extra thrust in a direction that could allow it to achieve low-energy orbits that would be otherwise inaccessible. Over long periods of time, the extra thrust could make a significant difference in the time needed to navigate the solar system.

Received 2 September 2010; revision received 6 July 2011; accepted for publication 22 July 2011. Copyright © 2011 by Erika Alden DeBenedictis. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the \$10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0748-4658/12 and \$10.00 in correspondence with the CCC.

\*Undergraduate Student; edebened@caltech.edu.

<sup>†</sup>Data available at <http://nssdc.gsfc.nasa.gov/nmc/masterCatalog.do?sc=1990-007A> [retrieved 8 May 2008].

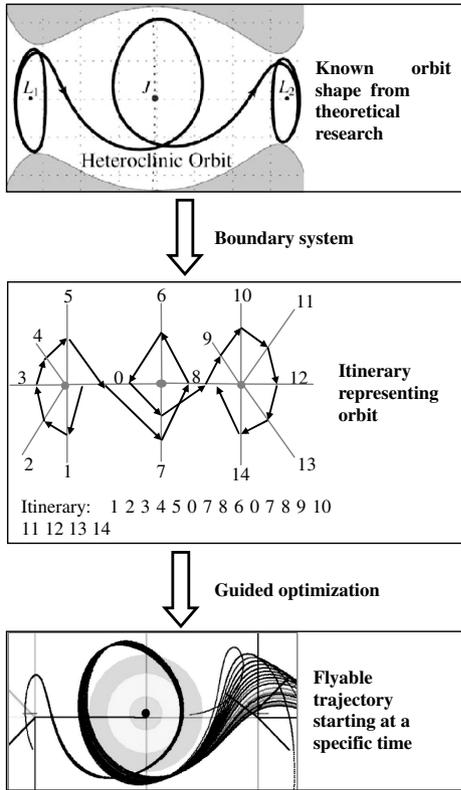


Fig. 1 Overview of method.

II. Method

The proposed navigation method is based on abstract spatial descriptions of orbits, as shown in Fig. 1. The method relies on an abstraction of the description of an orbit. A basic orbit shape is converted into an itinerary using a system of spatial boundaries. This itinerary is then used to compute a trajectory that satisfies the original orbit shape. This trajectory may differ due to  $n$ -body effects or, as shown in the bottom picture, continuous propulsion. Known orbit shapes, such as heteroclinic connections and Lyapunov orbits, are described as an ordered list of boundaries in space that a trajectory should cross. An algorithm then identifies specific initial conditions of a trajectory that satisfies the itinerary. The method is implemented by boundaries that define the supported orbits with sufficient precision to permit reliable navigation but do so abstractly enough that trajectories of the same orbit type that differ by  $n$ -body effects have the same description. Note that a heteroclinic connection is a connection between two unstable orbits. In this paper, a heteroclinic connection refers to a transfer from an Earth–moon  $L_1$  Lyapunov orbit to an Earth–moon  $L_2$  Lyapunov orbit. For further description, see [1].

The algorithm is autonomous in that it could be executed by an onboard computer of the expected modest performance levels and function with sensor data of the expected limited precision. The specific design and requirements for such onboard sensors are potentially a significant research effort and are beyond the scope of this paper. Computational reliability is the key to designing the autonomous algorithm such that it always succeeds in precisely navigating sensitive low-energy paths. The research behind this paper consists of selecting and adjusting boundaries to accommodate the supported orbits, then testing until autonomous navigation proved effective, and incorporating the trajectory-finding algorithm into a system that can use low-thrust propulsion.

The demonstration program developed in this project studies methods of autonomous navigation. The program was written in C++ with multithreading capabilities. The planar positions of the planets are calculated using ellipses and represent a specific date, as given by the equations of Schlyter [9]. The program then uses fourth-order Runge–Kutta, as described by Hut et al. [10], to track the positions of

tracer particles, as they are affected by the gravity and movement of the planets. The user interface consists of a browser displaying various views of the simulation region as well as several data export files.

Note that the simulation in this program is planar. In a three-dimensional (3-D) simulation, each of the boundary lines would become a surface designed to accommodate 3-D orbits such as halo orbits. Some minor correction of the continuous propulsion angle would be needed in order to best exploit the gravity of the planets in question as well as to properly arrive at another planet. While these corrections would be nontrivial, the existing algorithm includes a large portion of the calculations required for a 3-D simulation. A 3-D simulation would also add significantly to the computational intensity of the problem. For these reasons, a planar simulation was used for the purposes of study.

The following sections detail the design of the boundary system and its use in constructing itineraries, the process by which guided optimization finds a trajectory that satisfies this itinerary, and, finally, how this system may be used to continuously calculate the thrust angle required to fly such an orbit with a continuous propulsion craft.

A. Boundary System

There are many possible constraints that could be used to describe orbit shapes. To implement the abstraction of orbits, the algorithm primarily uses boundary lines, which are line segments approximately perpendicular to a trajectory completing the orbit. By placing many boundary lines approximately perpendicular to the path of the orbit shape, it is possible to describe an itinerary for the orbit shape in terms of an ordered list of boundary line crossings. A guided optimization algorithm finds specific initial conditions for some orbit path by simulating the divergence of many tracers, grading each tracer’s path based on how well its path followed the specified itinerary, and then simulating more tracers that have initial conditions close to the one that best satisfied the itinerary. A spacecraft given the resulting initial conditions, in terms of position, velocity, and thrust angle, could fly the orbit.

The number, size, length, and placement of boundaries are crucial to the functioning of the algorithm. Between two successive boundaries, there must be at least a few tracers that reliably reach the next boundary in order for the algorithm to have sufficient information to find the best initial conditions. Also, the relative placement of the boundaries should be such that incorrect trajectories should not only fail to cross the proper boundary but also cross a clearly incorrect boundary. The boundary lines must be general enough to accommodate the orbit at all different times while still being specific enough to sufficiently constrain the algorithm to identify initial conditions that produce the correct type of orbit.

Figure 2 depicts the boundary lines used to construct itineraries for Lyapunov orbits, heteroclinic connections, and flyby maneuvers in both the Earth–moon and Earth–sun systems. The boundary lines

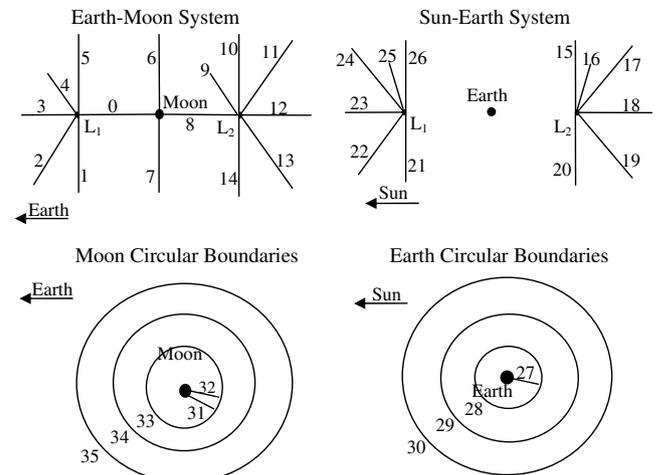


Fig. 2 Boundary positions near Earth.

were heuristically defined to be abstract enough to be valid for highly perturbed versions of the supported orbit shapes but specific enough to keep the paths sufficiently on track. These lines are supplemented by circular boundaries around planets, which detect when spacecraft go near planets and are useful for describing flyby maneuvers. The circular boundaries around the Earth are at distances of 1, 1.5, and 20 times Earth’s diameter above Earth’s surface. Similarly, the moon’s boundaries are at distances of 3.5, 10, and 15 times the moon’s diameter. Circular boundaries in the moon’s area are not implemented at the same time as the straight boundary line systems because they overlap. The top two systems are used to describe Lyapunov orbits and heteroclinic connections and the bottom two are used to describe flyby maneuvers.

When designing boundary systems, one method of creating an effective system is to alternate between long easy-to-cross boundaries and shorter, more precise boundaries. The long boundaries allow the algorithm to more quickly distinguish better tracers during difficult portions, and the short, specific boundaries further refine these rough orbits produced by the long boundaries. Lyapunov orbits are one place where this method is helpful, resulting in the disparities in size between boundaries 3, 4, and 5, as well as 9, 10, and 11; 24, 25, and 26; and 15, 16, and 17. Similarly, the difference in the sizes of the circular boundaries near the moon also helps the algorithm find flyby paths.

**B. Guided Optimization**

Figure 3 shows example cases of identifying the best tracer or tracers for a simulation of a Lyapunov orbit. The best tracer is the one for which the itinerary continuously matches the specified itinerary for the most number of boundary crossings, starting at the beginning. The boundary crossings of each tracer are listed; the highlighted crossings are correct according to the given itinerary. Note that, to save computational time, the algorithm ceases to simulate tracers as soon as they cross an incorrect boundary, as the desired trajectory completes the entire itinerary. In some cases, such as iteration 1, there are multiple tracers that perform equally well. The program identifies these  $c$  correct tracers. The range,  $r = c + 1$ , is the number of

sections of initial condition phase space that the tracers will span on the next iteration. The top left portion of Fig. 3 shows an orbit shape and its corresponding trajectory. The middle column shows 20 tracers, their paths, and the boundaries they crossed. The range indicates the tracers that the algorithm identifies as equally good in terms of following most of the desired itinerary. In the third column, the algorithm refines the positions of the tracers so that all the new tracers lie within the range determined by the previous iteration, allowing further refinement to be made.

The initial conditions of the tracers may be different in position, velocity, or acceleration. When considering the zero-thrust case, the initial conditions are set by positioning the  $n$  tracers  $x$  meters apart along the moon- $L_1$  line and performing a minor adjustment in velocity based on position using the Jacobi integral [11] to ensure that each tracer has the same energy. For example,  $nx = 500$  km centered 48,000 km from the moon. When the program attempts to optimize the direction of thrust, the tracers begin at the same position 48,000 km from the moon along the moon- $L_1$  line and velocity set by the Jacobi integral. The difference in direction of thrust  $\Delta\theta = \frac{\pi}{2n}$ , centered around synodic acceleration angle (see Sec. II.C).

When performing a refinement of initial conditions, the algorithm has  $c$  best tracers and  $r = c + 1$  ranges to fill ( $c - 1$  gaps and one location off each end). Each range is split into two ranges by adding a tracer for which the initial position, velocity, and thrust vector are the average on the range’s original left and right tracers. This process is repeated, splitting the range with the largest position difference between the edge tracers, until the number of total tracers is reached.

The addition of three special case boundaries have been experimentally shown to reduce the time needed for the computer to consistently complete the itinerary. The first type is the optional boundary, denoted by a question mark preceding the boundary in the itinerary. An optional boundary has two interpretations: it is a boundary that should be crossed in some alignments of the Earth-moon and Earth-sun systems but not others, or a boundary that, when crossed, indicates to the algorithm that it is near the correct trajectory (in other words, a hint). In the example shown in Fig. 4, tracer 2 would be considered better than tracer 1 because, in addition to completing the first  $n - 1$  elements of the itinerary, it also crossed the

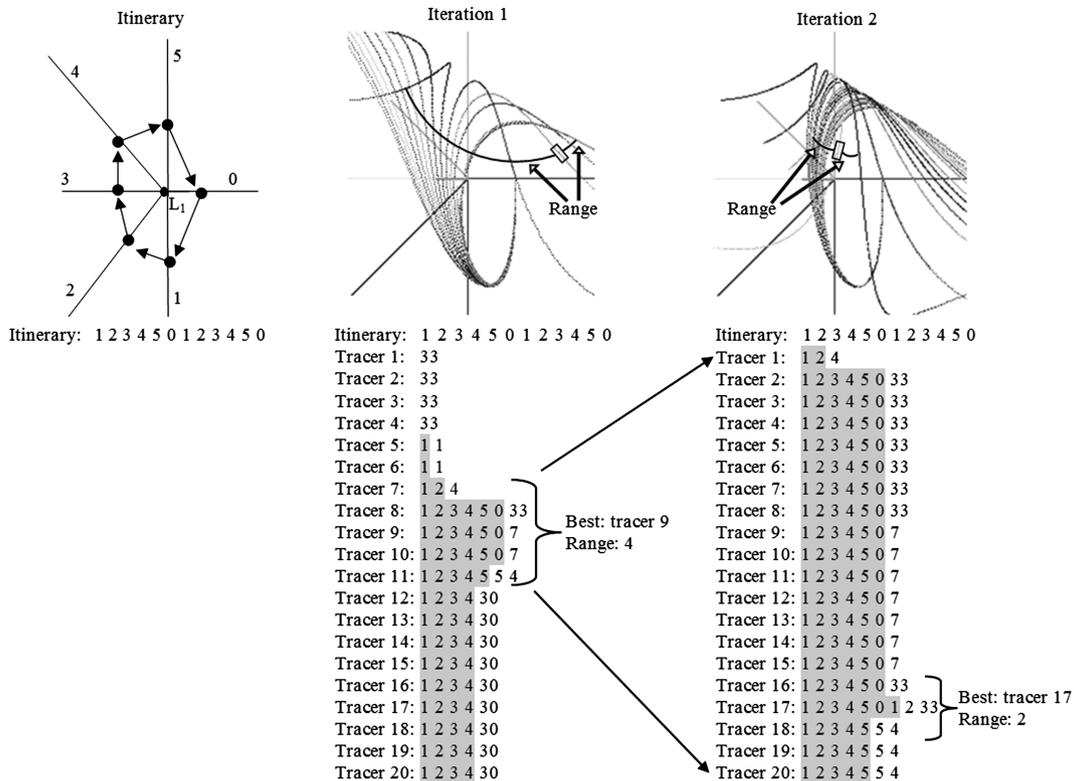


Fig. 3 Example of itinerary-based optimization.

```

Itinerary: 1 2 3 4 5 ?5 0 1 2 3 4 5 0
Tracer 1: 1 2 3 4 5
Tracer 2: 1 2 3 4 5 5
Tracer 3: 1 2 3 4 5 5 0
Tracer 4: 1 2 3 4 5 0

Final trajectory could be
1 2 3 4 5 5 0 1 2 3 4 5 0
or
1 2 3 4 5 0 1 2 3 4 5 0
    
```

**Fig. 4** Optional boundaries.

specified  $n$ th optional boundary. However, tracers 3 and 4 would both be considered equally good, and both are better than tracer 2 because they completed the  $n + 1$  element of the itinerary, at which point the optional boundary is irrelevant. Optional boundaries are used by the software system in two ways. First, boundaries may be specified as optional when it is unclear if they are necessary: some trajectories may be correct both with and without a particular boundary crossing. This is useful, for example, when the path jumps from the Earth–moon system to the sun–Earth system. The last boundary in the Earth–moon system should be optional to account for the relative alignment of the systems. The second use is to assist the algorithm in identifying a trajectory on which to optimize in locations where there tend to be many trajectories that satisfy the itinerary equally well. The itinerary in Fig. 4 (the actual itinerary for a Lyapunov orbit, as shown in Fig. 3) is an example of this. Often, tracers exhibit the itinerary “1 2 3 4 5 33” (completing most of a Lyapunov orbit and then crashing into the moon). If the algorithm at some point optimizes on “1 2 3 4 5 5”, then it will be immediately close to the correct initial conditions to complete the Lyapunov orbit. Note that, in this case, the correct, final version of the itinerary does not include the optional boundary, which is merely added to speed computation.

Sets of boundary lines may be turned on or off when the tracers complete a portion of the itinerary with switch boundaries. For example, if a path includes a heteroclinic connection and then later a moon flyby, the program should not have both the standard Earth–moon boundary lines as well as the circular moon boundary lines on at the same time, as they overlap and would interfere with each other. The only exception to the rule that overlapping boundaries should not be used at the same time is boundary 33, the smallest moon circular boundary, which is turned on at the same time as the linear Earth–moon boundary lines. Figure 3 shows a simulation in which this is the case. As none of the orbits in the area need to go closer to the moon than boundary 33, discontinuing the simulation of tracers as they get too close to the moon saves computational time. Instead, once all the tracers have cleared the heteroclinic connection, the now-unnecessary standard Earth–moon boundaries are turned off and the circular boundaries turned on.

A boundary may also be specified as a reload boundary. For example, when the program detects that all the tracers have

successfully completed the itinerary up to that point, it will save the positions of the tracers as they cross the boundary and then restart the next simulation from that point to save simulation time. This type of boundary is necessary in the zero-thrust case for long trajectories; it prevents the computation of initial conditions from accumulating past the point of computer precision. In the case of nonzero thrust, the spacecraft continuously performs a similar function to a reload boundary in order to begin simulations at its current position, as described in the next section.

The types of boundaries are summarized in Table 1. The table details the types of boundaries used in itineraries and how they are used by the algorithm.

Each basic orbit shape (such as a Lyapunov orbit or a heteroclinic connection) may be described by a specific set of boundary line crossings, a standard itinerary. This standard itinerary may include optional boundaries, switch boundaries, or reload boundaries that have been shown through heuristic experimentation to make it easier for the program to identify the correct trajectory. These standard itineraries, shown in Table 2, are constant, including the special boundaries, and are not changed by user input. Orbit shapes may be converted into itineraries with the above itineraries. These orbit shapes may only be preceded by or followed by certain other orbit shapes.

While the language allows any orbit shape to be combined, not all combinations yield functional orbits (performing an Earth–moon  $L_1$  Lyapunov orbit and then sun–Earth  $L_2$  Lyapunov orbit, for example, would not work). Each orbit shape may be preceded and followed by other orbit shapes, as detailed in Table 2. Because of the relative alignment of the Earth–moon and sun–Earth systems, only some orbits will work at a given time; that is, after exiting the Earth–moon system, the spacecraft can perform either a sun–Earth  $L_1$  Lyapunov or a sun–Earth  $L_2$  Lyapunov.

As previously mentioned, the placement, length, and number of the boundaries have been defined heuristically so that the optimization algorithm can use them to efficiently and reliably find specific trajectories. However, there are very rare cases where the algorithm will fail to find the appropriate path, either because the tracers in question have an unusually low or high amount of energy relative to what the boundary system was designed for or because the boundaries were not specific enough to allow the algorithm to optimize on the correct trajectory at some point. The first case could be solved by making the boundaries more general, while the second would require the boundaries to be more specific. Since, at this point, the boundary systems cannot be further optimized, a different approach to reliability is employed. The program can detect that it is not making progress completing more of the itinerary if it has not completed another point in the itinerary after three iterations of refinement. When this is the case, it will back up to the configuration three iterations before, in which the tracers were less optimized, double the number of tracers, and continue. This is a fairly effective

**Table 1** Boundary types

Boundary type	Representation in itinerary	Meaning to the orbit	Implementation
Linear	Boundary line number	Approximately perpendicular to the correct direction of motion	A spacecraft that crosses the first $n + 1$ boundaries of the itinerary is better than one that crosses the first $n$ boundaries.
Circular	Boundary line number	Describe flyby maneuvers	A spacecraft that crosses the first $n + 1$ boundaries of the itinerary is better than one that crosses the first $n$ boundaries.
Optional	? precedes number.	Used to account for relative phasing of Earth–moon and Earth–sun systems. Used as optimization hint for algorithm.	If the optional boundary is the $n$ th element of the itinerary, a spacecraft that crosses the first $n$ boundaries is better than one that crosses the first $n - 1$ but worse than one that crosses reaches the $n + 1$ element, even if it skips the optional boundary.
Switch	d precedes number to turn off Earth–moon linear boundaries. g precedes number to turn on moon circular system.	Allows appropriate boundary system to be used in areas where two overlap.	After all the tracers have crossed this boundary, the appropriate set of boundary lines is turned on or off.
Reload	> precedes number.	Saves computational time by moving the start of the simulation forward in time.	After all the tracers have crossed this boundary, the program saves their positions, velocities, and thrust angles for use as initial conditions for the following simulation.

**Table 2** Standard itineraries

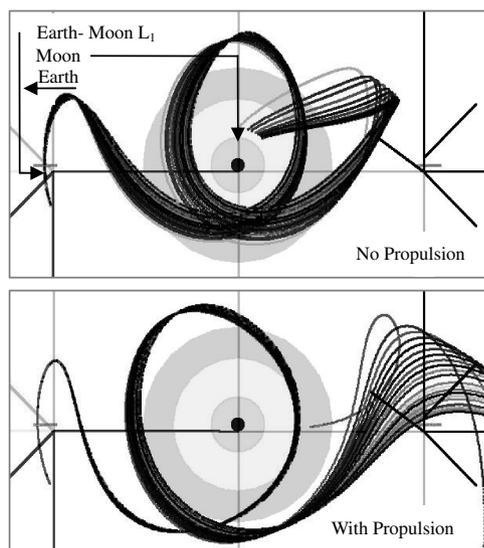
Orbit Shape	Abbreviation	Itinerary	Preceded by	Followed By
Earth–moon $L_1$ Lyapunov	EML1	?0 1 2 3 4 5 ?5	EML1	EML1, EMHC
Earth–moon $L_2$ Lyapunov	EML2	?9, 10, 11, 12, 13, 14, 8	EMHC, EML2	EML2, Trans.
Earth–moon heteroclinic connection	EMHC	0, 7, 8, 6, 0, 7, 8	EML1	EML2, Trans.
Transition between Earth–moon and sun–Earth	Trans.	?9, 10, 11, 12, ?dg13	EML2, EMHC	SEL1 or SEL2
Sun–Earth $L_1$ Lyapunov	SEL1	>21, 22, 23, 24, ?25, 26, ?26	Trans., SEL1, SEL2	SEL1 SEL2, MF, EF
Sun–Earth $L_2$ Lyapunov	SEL2	>15, ?16, 17, 18, 19, 20, ?20	Trans., SEL1, SEL2	SEL1 SEL2, MF, EF
Moon flyby	MF	35, 34, 33, 31, 32, 33, 34, 35	SEL1, SEL2, EF	EF
Earth flyby	EF	30, 29, 28, 27, 28, 29, 30	SEL1, SEL2, EF	MF

way of identifying the correct trajectory. However, on a spacecraft, this is costly in terms of computational power. For a spacecraft under the time constraint of requiring an answer in a set amount of time, this approach to reliability may result in an inability to calculate the answer fast enough. Instead, it may be more efficient to simply disregard the bad simulation and continue with the next time interval, as described in the next section.

### C. Autonomous Spacecraft Navigation System

Any spacecraft must have some propulsion to perform the stationkeeping maneuvers necessary to fly sensitive low-energy paths. If measurements and computation were perfect, and solar wind could be perfectly forecast, simply pointing the spacecraft in exactly the right direction would cause it to fly the orbit. To the extent these assumptions are not met, stationkeeping propulsion will be necessary to keep the spacecraft on the desired orbit. An ion drive is a good choice for a type of continuous propulsion that would provide the minute amounts of thrust required.

There will also be times when the spacecraft does not need to use all available thrust to perform stationkeeping. In these instances, there might be a way to put the continuous propulsion to good use in changing the spacecraft's energy. Accelerating in the direction opposite the spacecraft's velocity in the synodic system is an efficient method of increasing energy and is referred to here as synodic acceleration. This method of increasing energy is useful when the spacecraft is attempting to climb out of a gravity well, as it does when it begins at Earth. Spacecraft with higher energy may be able to execute certain maneuvers that lower energy spacecraft could not. For example, Fig. 5 shows how the addition of a small amount of continuous propulsion can allow tracers to complete a heteroclinic connection that was previously inaccessible. The addition of continuous propulsion can allow spacecraft to execute maneuvers that would not otherwise be possible.

**Fig. 5** Using continuous propulsion to increase energy.

The autonomous spacecraft navigation system acts as the navigation computer for a spacecraft, calculating how to use continuous propulsion for both stationkeeping as well as increasing energy when helpful to the mission. It works with the following steps:

- 1) The spacecraft uses its onboard computer to project where it will be at some time interval in the future. This interval will be  $t$  min, as defined by a safe estimate of how long the simulation will take to run.

- 2) The software uses the itinerary-based algorithm to calculate how the spacecraft could best use its continuous propulsion to fly the desired route. The initial conditions of all the tracers are located at the spacecraft's anticipated position and differ by the direction of the continuous thrust vector. This propulsion angle is used for the first  $t$  min of the simulation, and then the angle reverts to synodic acceleration. The program optimizes the angle for the continuous propulsion that would allow the spacecraft to fly the correct route.

- 3) At the end of the time interval, the spacecraft arrives at the anticipated location and reorients its thruster to the direction of continuous propulsion it just calculated. Meanwhile, the onboard computer repeats steps 1 and 2 for the next time interval.

This allows the spacecraft to perform stationkeeping by updating its continuous propulsion angle to adjust for navigational errors. Also, this method provides a balance between stationkeeping and increasing energy. Since the majority of the path is simulated using the synodic acceleration angle, the small time interval of variable acceleration angle will only differ from the assumed synodic angle by however much is absolutely necessary to correct for the last  $t$  min error.

Algorithmic reliability for an onboard spacecraft navigation system is crucial. There is the possibility that a simulation error would fail to find a proper trajectory. In such a situation, the spacecraft would simply use the synodic acceleration angle for the full time interval, which will be similar to the properly calculated angle. A more serious possibility would be that a temporary hardware malfunction of some kind would send the spacecraft offtrack enough that it would not be possible to recover the correct path with the available amount of propulsion. The likelihood of this occurring can be reduced by increasing the frequency at which the spacecraft updates its path. Using the software system, the simulated spacecraft will go off track with an update time of  $t > 60$  min. However, with an update time of  $t = 30$  min (meaning that the simulated spacecraft updates its propulsion angle every 30 min), a desktop computer could successfully simulate 12 h worth of spacecraft time in about 3 min: a speedup of  $240 \times$  real time. Of course, a deployed spacecraft's hardware and the software simulation program would be completely different. The spacecraft would also be running other systems that would require computational time, making it difficult to determine the computational power required for the software to run with sufficient accuracy in real time. An in-depth study of computer speed, update time, and required navigation accuracy is the subject of future research. However, the current results indicate that the spacecraft could be reasonably expected to safely simulate its own required propulsion angles.

An interesting concept revealed by this study is the tradeoff between computational power and additional fuel. If a spacecraft updates its acceleration angle more frequently, it will be reducing navigation error each time. This results in an acceleration angle

closer to the synodic acceleration angle. Spacecraft with more computational power will therefore make more effective use of available fuel.

While increasing the energy of the spacecraft is, in general, a good idea, it is possible that specific trajectories might benefit more from using their available propulsion to accelerate in a direction other than opposite the spacecraft’s synodic velocity. Analyzing and incorporating this circumstance into the simulation is a topic of future research.

### III. Results

Reaching another planet with low-energy orbits and minimal propulsion is challenging. Using this software system, a number of highly eccentric low-energy orbits were identified that could put the spacecraft on a path for Venus, the transit point for many other gravity-assist maneuvers. As spacecraft exit the Earth’s neighborhood for the first time, they have the unique opportunity to interact with the Earth–moon system in such a way that they diverge from Earth’s orbit as much as possible.

Several unpropelled exit strategies, shown in Fig. 6, were chosen in which the tracers interact with both the Earth and the moon during their departure from the region. The five exit strategies are maneuvers that interact with the Earth–moon system for the last time when leaving from the area. These maneuvers are evaluated for their ability to allow spacecraft to reach other planets. As the tracers enter interplanetary space, the system calculates and records their orbits around the sun. The results of these calculations are shown in Fig. 7.

The goal is for the tracer’s orbits’ perihelion to the sun to be closer than Venus’ aphelion. Note that the most effective exit strategy, the heteroclinic connection, moon gravity-assist, Earth flyby, is able to change a tracer’s orbit by about half of the amount needed to interact with Venus. While this is a significant distance for an unpropelled maneuver, it is nevertheless only halfway to the destination. Consider that in interplanetary space there are few available low-energy maneuvers a spacecraft can use to further change its orbit. However, the research results indicate that small amounts of continuous propulsion could greatly decrease the transit time for the spacecraft to finish changing its orbit.

Consider a small spacecraft propelled by an ion drive and generating power with solar panels. Assume there are 0.25 m<sup>2</sup> of solar panels dedicated to the ion drive, in total yielding 114 W, and 25 kg of propellant lasting for 10 years or 3.15 × 10<sup>3</sup> s. Using the power of the solar panel to calculate the velocity of the propellant exiting the ion drive,  $P = (1/2)(dm/dt)v^2$ , so 114 W = (1/2) (25 kg/3.15 × 10<sup>3</sup> s)v<sup>2</sup> and v = 53 km/s, which is reasonable for an ion drive. The force exerted is

$$F = \frac{dM}{dt} v = \frac{25 \text{ kg}}{3.15 \times 10^3 \text{ s}} \frac{53 \text{ km}}{\text{s}} = 4.3 \times 10^{-3} \text{ N}$$

and for a spacecraft of mass 130 kg, the resulting acceleration is 6.5 × 10<sup>-5</sup> m/s<sup>2</sup>. A spacecraft can quickly decrease its perihelion by directing this propulsion in its direction of motion in the sun–Venus system. These simulations show, as indicated in Fig. 7, that it would take only 1.5 additional years of continuous propulsion for the spacecraft’s perihelion radius to equal Venus’s aphelion radius. With

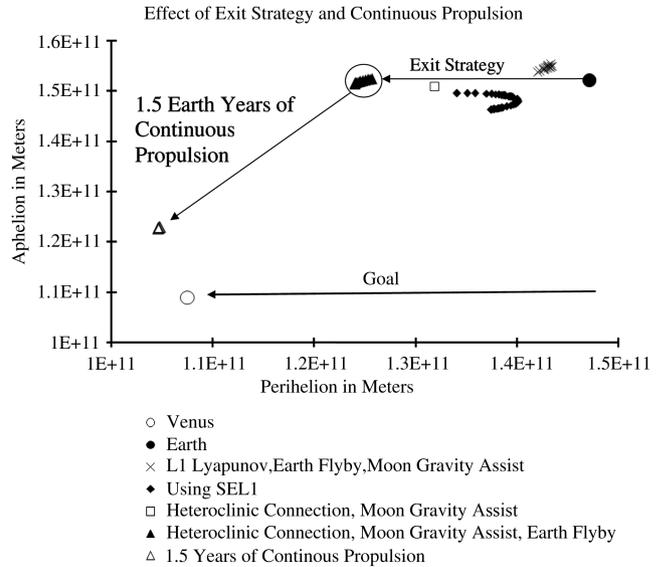


Fig. 7 Reaching Venus with low-energy orbits.

proper phasing, this research indicates that a spacecraft could reach Venus using only low-energy orbits and minute amounts of continuous propulsion within four or five years of its launch from Earth.

Once a spacecraft reaches another planet, it can perform a gravity assist. One example is the famous VEEGA (Venus–Earth–Earth gravity assist) used by the Galileo spacecraft and other maneuvers described by Petropoulos et al. [7], which allow spacecraft to reach the rest of the solar system.

### IV. Analysis of Method

This method can be extended to include any minor effect that can be programmed. This is in contrast with some of the more mathematically intensive methods, where perturbations must be built into the mathematical formulation of the equations or patched in later, which is often difficult and sometimes impossible to do. For example, if signals from a millisecond pulsar are being used for navigation, this system could be coded to terminate tracers where a planet blocks the signal from the pulsar. It would also be possible to test different types of continuous propulsion using this method. For example, solar sails could be simulated by only allowing tracers’ thrust to go downwind. The software could be easily modified to accommodate other constraints such as communication, radiation, orientation, and heat restraints for the spacecraft.

The boundary line systems can be extended to include some, but not all, additional types of orbits. Boundary lines and itineraries could be constructed for heteroclinic connection and Lyapunov orbits through many planet–sun combinations and, probably, a more complex set through the orbital system of Jupiter or Saturn and its moons. However, certain flyby orbits (such as described in [12]) would be difficult for this method to identify because the maneuver

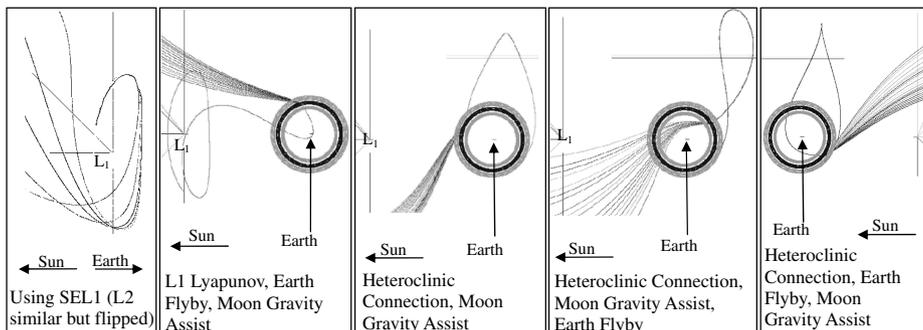


Fig. 6 Exit strategies.

depends on orbital phasing that is not easily represented by spatial boundary lines defined in the rotational framework of a pair of astronomical bodies. These orbits are currently planned using the invariant manifold method, but that method uses human judgment for selecting orbital geometry as opposed to itineraries that could be executed incrementally by a spacecraft computer.

The method described here could become part of a more comprehensive orbit planning environment. Some routes are only possible when certain phase relationships exist between different sets of two-body systems. A more complex navigation program could include this method coupled with automatic phase relationship calculation. The existing method can calculate thrust angle as a spacecraft transitions between different two-body systems, but a method to determine if such a transition exists at the given time has not yet been studied.

Since it is possible to construct itineraries out of different types of constraints, it should be possible to create an autonomous spacecraft navigation system for low-thrust spacecraft flying low-energy orbits that is not based only on boundary lines. One possibility is that sections of invariant manifold could be used as the elements of an itinerary. While it would be difficult, and in some situations impossible, to account for all perturbations, it would expand the maneuvers available in cases that included only limited perturbations. The software system could employ the appropriate method (spatial boundaries or invariant manifold adherence) based on characteristics of the present portion of the trajectory. A system that incorporated both methods could potentially retain autonomy and the ability to calculate low thrust at the same time as expanding the repertoire of available orbit maneuvers.

The itinerary-based algorithm has many advantages when used from the perspective of researching low-energy orbits affected by many perturbations. The method of describing orbits in terms of spatial itineraries is an interesting concept that could be more widely incorporated into other methods of orbit planning.

## V. Potential Applications

This research's approach may enable new types of space missions. One scenario that would benefit significantly by this autonomous navigation algorithm might involve large booster launches of dozens of microsattellites, each of which is propelled by a solar sail or small ion drive. The operations center could give the microsattellites commands to populate the Earth's environment in a specific way. While these commands could be given to dozens of spacecraft in a reasonable time, monitoring and controlling the orbit of dozens of spacecraft could burden the operational resources on Earth. With the autonomous navigation approach, the dozens of spacecraft could navigate for years with only minimal contact with Earth.

Another potential use of low-energy orbits would be to aid in the transit of heavy spacecraft, such as unmanned supply crafts or cargo crafts for asteroid mining. Since these paths take longer to execute than traditional methods, these supply craft could be launched ahead of the primary mission and then wait in orbit near the destination. This method would allow larger-scale missions in remote parts of the solar system because of the increased availability of materials. In the case of asteroid mining, low-energy orbits could provide a cost-effective method for transportation across vast distances with heavy cargos.

## VI. Conclusions

A method has been developed to abstractly describe orbits in terms of their spatial characteristics. This method has been incorporated

into an optimization algorithm capable of identifying the initial conditions of a trajectory that correspond to an input orbit itinerary. The use of continuous propulsion has been explored and incorporated into a concept design for a software system that could operate autonomously on a spacecraft. This system would be capable of identifying the appropriate thrust angle to perform stationkeeping in real time as well as increasing the energy of the spacecraft when useful. This system has been used to investigate both unpropelled and propelled methods of changing a spacecraft's orbit, and initial results indicate that a spacecraft could reach Venus in four or five years of its launch from Earth with minimal propulsion while flying identified low-energy trajectories.

## Acknowledgments

I would like to thank my mentor, Erik DeBenedictis. I would also like to thank Beverly DeBenedictis for her editorial guidance.

## References

- [1] Lo, M., "The InterPlanetary Superhighway and the Origins Program," *Aerospace Conference Proceedings*, Pasadena, IEEE, Piscataway, NJ, 2002, pp. 7-3543-7-3562.
- [2] Howell, K., and Masaki, K., "Transfers Between the Earth-Moon and Sun-Earth Systems Using Manifolds and Transit Orbits," *Acta Astronautica*, Vol. 59, Nos. 1-5, 2006, pp. 367-380. doi:10.1016/j.actaastro.2006.02.010
- [3] Howell, K. C., Barden, B. T., and Lo, M. W., "Application of Dynamical Systems Theory to Trajectory Design for a Liberation Point Mission," *Journal of the Astronautical Sciences*, Vol. 45, No. 2, 1997, pp. 161-178.
- [4] Koon, W. S., Lo, M. W., Marsden, J. E., and Ross, S. D., "Dynamical Systems, the Three-Body Problem and Space Mission Design," *International Conference on Differential Equations*, Berlin, World Scientific, Singapore, 2000, pp. 1167-1181.
- [5] Ross, S., and Grover, P., "Designing Trajectories in a Planet-Moon Environment Using the Controlled Keplerian Map," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 2, March-April 2009, pp. 436-443. doi:10.2514/1.38320
- [6] Strange, N., and Longuski, J., "Graphical Method for Gravity-Assist Trajectory Design," *Journal of Spacecraft and Rockets*, Vol. 39, No. 1, Jan.-Feb. 2002, pp. 9-16. doi:10.2514/2.3800
- [7] Petropoulos, A., Longuski, J., and Bonfiglio, E., "Trajectories to Jupiter via Gravity Assists from Venus, Earth, and Mars," *Journal of Spacecraft and Rockets*, Vol. 37, No. 2, Nov.-Dec. 2002, pp. 776-783. doi:10.2514/2.3650
- [8] Koon, W. S., Marsden, J. E., Ross, S. D., and Lo, M. A., "The Genesis Trajectory and Heteroclinic Connections," *Astrodynamics 1999: Proceedings of the AAS/AIAA Astrodynamics Conference*, Girdwood, AK, Vol. 103, No. 3, American Astronomical Soc., Washington, D.C., 2000, pp. 2327-2343.
- [9] Schlyter, P., "Computing Planetary Positions" [online], <http://stjarnhimlen.se/comp/ppcomp.html> [retrieved 10 Nov. 2007].
- [10] Hut, P., Makino, J., and Heggie, D., "Integration Algorithms: Exploring the Runge-Kutta Landscape," *The Art of Computational Science* [online], Sept. 2007, [http://www.artcompsci.org/kali/vol/runge\\_kutta/title.html](http://www.artcompsci.org/kali/vol/runge_kutta/title.html) [retrieved Jan. 8, 2008]
- [11] Szebehely, V., *Theory of Orbits*, Academic Press, New York, 1967, pp. 7-22.
- [12] Koon, W. S., Lo, M. W., Marsden, J. E., and Ross, S. D., "Shoot the Moon," *Spaceflight Mechanics 2000*, Vol. 105, Part. 2, American Astronomical Soc., Washington, D.C., 2000, pp. 107-1181.

G. Spanjers  
Associate Editor