An Optimization Algorithm for Space Mission Design

Dynamically Simulating Energy-Efficient Trajectories

Erika DeBenedictis
May 14, 2008

Table of Contents

1. Executive Summary

1.1 Introduction

To navigate the solar system effectively, the gravity and movement of planetary bodies can be used to boost a spacecraft's speed and change the spacecraft's position and velocity. Simulating these interactions can reveal the optimal way to launch a spacecraft that minimizes fuel by maximizing the effect of planets in lieu of propulsion.

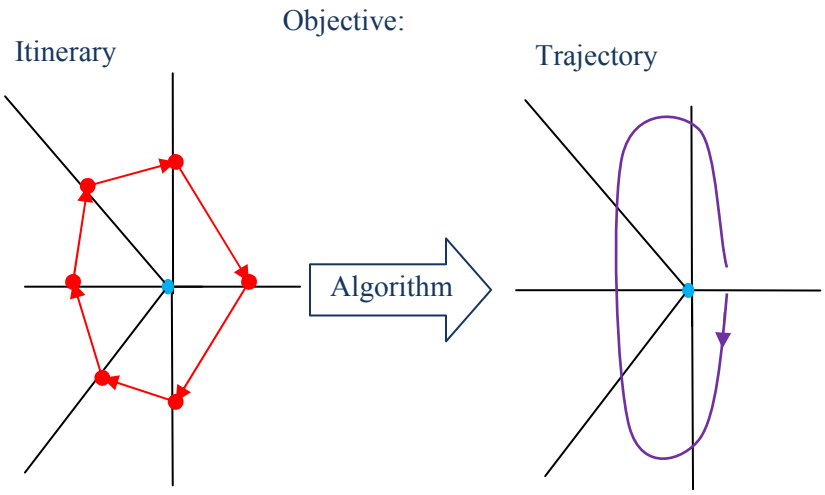
This project focuses on the simulation and optimization of spacecraft trajectory as well as the mathematical theory underlying this process. Its end goal, using the theory of low-energy orbits as a mathematical basis, is to automatically construct trajectories that go from one planet to another with minimal energy expenditure.

1.2 Problem Statement

How can complex low-energy spacecraft paths be automatically planned and efficiently simulated?

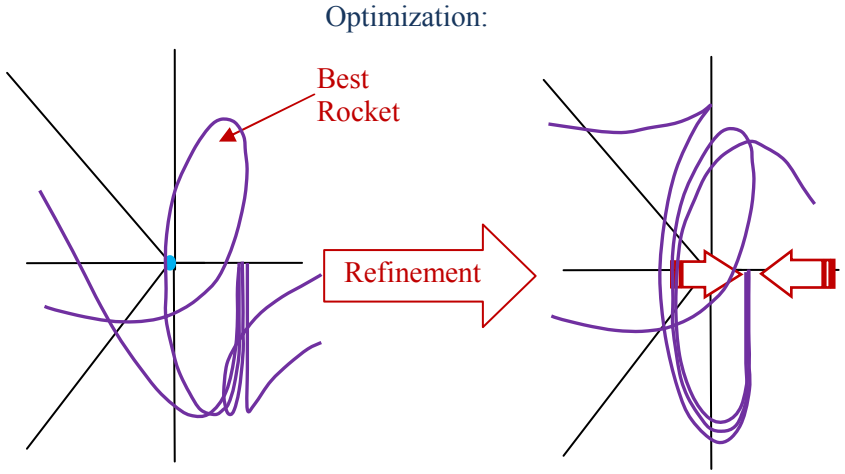
1.3 Novel Method

In this project, a novel itinerary-based algorithm is developed to find spacecraft trajectories for specified routes. To navigate in an area of space, spatial boundaries are established, and as a spacecraft moves through the area without additional thrust, its route may be described by the order in which it crosses these boundaries. The objective is to specify an itinerary of boundary crossings, and have the computer automatically find the trajectory that satisfies the crossings.



Since the optimal launch conditions are unknown, the itinerary of boundary lines should be valid for all potentially optimal paths. The objective of the simulation is to identify the optimal trajectory that satisfies the itinerary.

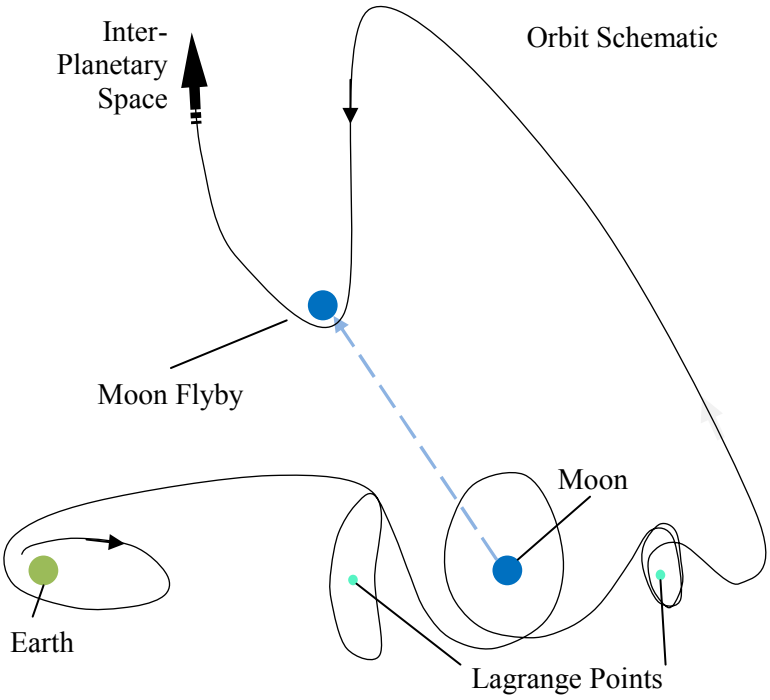
In order to do this, the computer uses the boundary crossings to optimize the initial trajectories. The program simulates the paths of several spacecraft as they are affected by the gravity of all the planets in a 2D simulation and records the boundaries they cross. Then it uses this data to choose the path that was closest to the desired boundary crossing order.



The 'best' path is used to optimize the initial conditions of all the spacecraft and simulate again. This guided optimization is and effective method for finding paths with specified itinerary, and works for complex non-elliptical orbits.

1.4 Results

The program utilizes multi-core processors for optimal performance. It has automatically planned and simulated Lypanov orbits, heteroclinic connections, and Moon gravity assists, thereby escaping the Earth's orbit without any propulsion. Below shows the general plan for the trajectory that involves all of these sections. The actual versions of these maneuvers may be found in Figures 14, 15, and 17.



2. Research

This section presents the concepts underlying orbit construction, focusing on gravity, planet-spacecraft interaction, mathematical methods used in the project, and existing methods used to find spacecraft trajectories. In researching these elements there were several sources that were especially helpful and form the basis of this project, including "Dynamical Systems, the Three-Body Problem, and Space Mission Design" by Koon, Lo, Marsden, and Ross.

2.1 Astrophysics

This section describes the forces a spacecraft experiences when moving through the solar system.

2.1.1 Elastic Collisions

The 'slingshot' interactions between a spacecraft and a planet rely on the movement of the two objects, rather than the planet's gravity, can be described as elastic collisions. In elastic collisions, both momentum and kinetic energy are conserved. In contrast, some kinetic energy is converted into another form of energy during an elastic collision. Should a spacecraft actually crash into a planet, its velocity would be converted to heat energy, it would be considered an inelastic collision. However, we are usually dealing with 'slingshots,' where the spacecraft never touches the planet, and there is no energy conversion.

We consider a spacecraft of mass m, initial velocity v_{1i}, final velocity v_{1f}, and a planet of mass M, initial velocity v_{2i}, and final velocity v_{2f}. Conservation of momentum tells us

$$mv_{1i} + Mv_{2i} = mv_{1f} + Mv_{2f}. \quad (1)$$

Conservation of Kinetic Energy says

$$\frac{1}{2}mv_{1i}^2 + \frac{1}{2}Mv_{2i}^2 = \frac{1}{2}mv_{1f}^2 + \frac{1}{2}Mv_{2f}^2. \quad (2)$$

Rearranging (1) we get

$$m(v_{1i} - v_{1f}) = M(v_{2f} - v_{2i}). \quad (3)$$

From (2) we have

$$mv_{1i}^2 - mv_{1f}^2 = Mv_{2f}^2 - M_2v_{2i}^2$$
$$m\left(v_{1i}^2 - v_{1f}^2\right) = M\left(v_{2f}^2 - v_{2i}^2\right)$$
$$m\left(v_{1i} - v_{1f}\right)\left(v_{1i} + v_{1f}\right) = M\left(v_{2f} - v_{2i}\right)\left(v_{2f} + v_{2i}\right).$$

Because of (3) we can cancel, and find that

$$\left(v_{1i} + v_{1f}\right) = \left(v_{2f} + v_{2i}\right). \quad (4)$$

Using equations (3) and (4) we can solve for the spacecraft's final velocity, v_{1f}

$$-mv_{1i} + mv_{1f} = -Mv_{2f} + Mv_{2i}$$
$$Mv_{1i} + Mv_{1f} = Mv_{2f} + Mv_{2i}$$
$$Mv_{1i} - mv_{1i} + Mv_{1f} + mv_{1f} = 2Mv_{2i}$$
$$v_{1i}(M - m) + v_{1f}(M + m) = 2Mv_{2f}$$
$$v_{1f} = \frac{2Mv_{2f} - v_{1i}(M-m)}{(M+m)} \quad (5)$$

Similarly, we can solve for the planet's final velocity, v_{2f}

$$mv_{1i} - mv_{1f} = Mv_{2f} - Mv_{2i}$$
$$mv_{1i} + mv_{1f} = mv_{2f} + mv_{2i}$$
$$2mv_{1i} = Mv_{2f} + mv_{2f} + mv_{2i} - Mv_{2i}$$
$$2mv_{1i} = v_{2f}(M + m) + v_{2i}(m - M)$$
$$v_{2f} = \frac{2mv_{1i} + v_{2i}(M-m)}{(M+m)} \quad (6)$$

Assuming that M is much larger than m, or that $m \cong 0$,

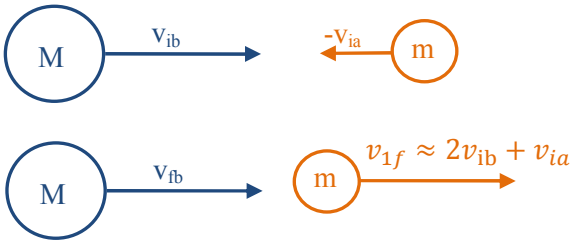
$$v_{1f} \approx 2v_{2f} - v_{1i} \quad (7)$$

$$v_{2f} \approx v_{2i} \quad (8)$$

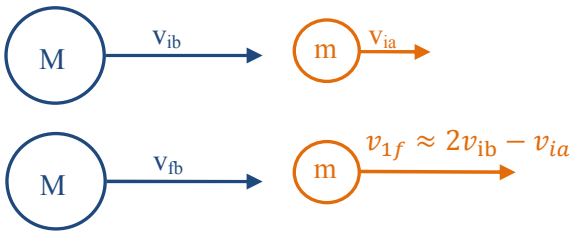
In other words, the planet's velocity is not affected by the spacecraft, while the spacecraft now has twice the planet's velocity.

We can now apply these equations to find the velocity gains for general spacecraft and planet interactions.

First we will consider when the planet and the spacecraft are moving toward each other. Notice the negative sign on the spacecraft's velocity accounts for its direction. We can use the equations to find the velocity of the spacecraft after the collision, which is substantially higher, as it is largely dominated by twice the planet's velocity. This is an example of a slingshot maneuver that would be especially advantageous for gaining velocity.



Now we look at what happens the bodies are traveling in the same direction:



Elastic collisions in two dimensions are essentially the same. Conservation of kinetic energy deals with a scalar, it is unnecessary to include the angle. However, we must change our Conservation of momentum equation to

$$mv_{1i}\cos\theta_{1i} + Mv_{2i}\cos\theta_{2i} = mv_{1f}\cos\theta_{1f} + Mv_{2f}\cos\theta_{2f}. \quad (9)$$

The solution to this equation is much more complicated, but essentially the same as its one-dimensional counterpart.

2.1.2 Frames of Reference

When dealing with systems where there are multiple moving objects it quickly becomes difficult to understand relative movement by looking at absolute movement. In many cases, both mathematically and graphically, we choose a synodic system. In a synodic system, we choose two objects we want to appear stationary, and then only look at the other objects' movement relative to our chosen objects. For example, when we think of a rocket leaving Earth to go to the Moon we would make the Earth and Moon appear stationary (rotating with them), while watching the rocket's progress toward the moon. If we looked at each objects movement around the sun it would be very difficult to gauge the rocket's progress toward the moon. In the same way, mathematically, we often choose a synodic frame of reference in order to simplify equations.

Another useful convention for a system with two massive bodies is to put the center of mass at the middle. This is called the center of mass, or CM, frame. This helps reduce many equations, especially those involving momentum, helping to further simplify our calculations. The combination of synodic and CM is another frequently used frame.

2.1.3 Two Massive Body System

In a two massive body system we can describe the potential energy as a combination of the gravitational potential energy from both bodies and the centripetal speed.

We consider a synodic CM system with total mass m+M units and two bodies. The larger body with mass M, is fixed at (-m,0) and the smaller mass m is fixed at (M,0), making the center of mass at the origin. A spacecraft of negligable mass at position (x,y). For the purposes of the example, we make M=99 and m=1, so the total mass is 100. We can describe the gravitational force on the spacecraft from each planet with

$$PE_g = \frac{99*G}{(-1-x)^2+y^2} + \frac{1*G}{(99-x)^2+y^2}$$, where $G=6.67*10^{-11}$

And the centripetal force as

$$F_c = \frac{x^2+y^2}{2}.$$

These equations can be combined and plotted using Mathematica. Exact commands used for each graph may be found under the title of the graph in the Appendix "Mathematica Equations." Figure 1 is a graph of total potential energy, calculated as shown.

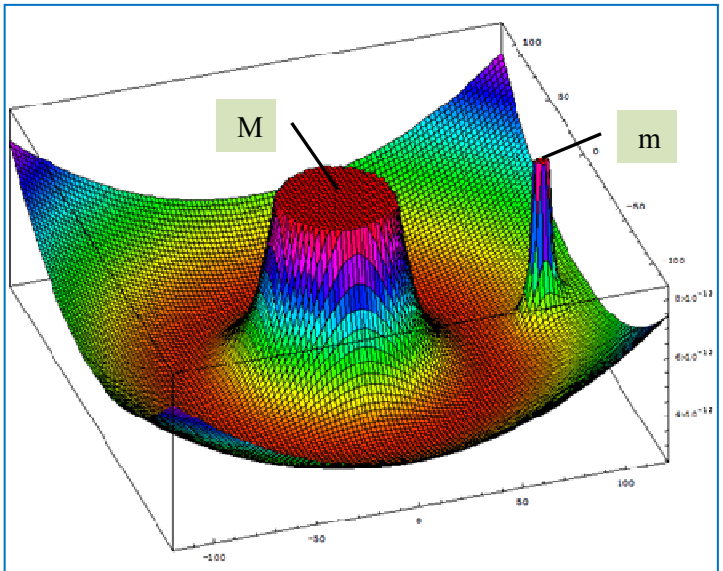


Figure 1: Potential Energy
Synodic, CM system. The larger body is in the middle, while the smaller body is at the top of the spike on the right.

Two areas of interest are the two red colored depressions on the sides of the central body's potential energy hill. By ignoring all the values we can isolate these areas and view them separately. Figure 2 shows these two areas, from the perspective of looking up at the bottom of the previous graph. These two areas are where the least amount of potential energy is acting on an object at that point. We will see later that they are also the location of two Lagrange points, and that objects tend to collect here.

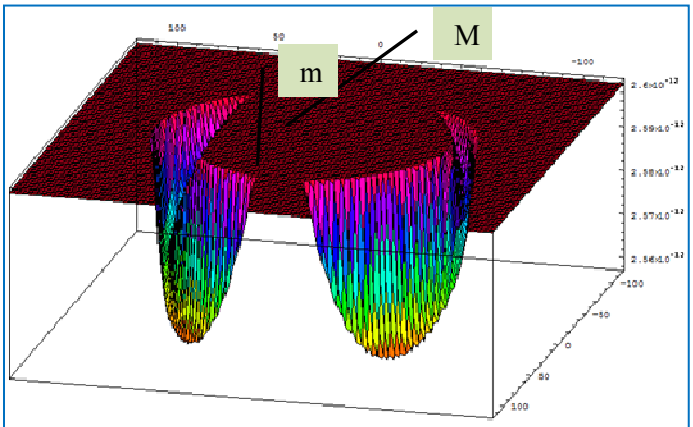


Figure 2: Lowest Potential Energy Depressions
Synodic, CM system. This graph shows the very bottom of the previous graph. The larger body is at the center, and the smaller body is in the break in the graph on the far side.

One of the challenges in launching any spacecraft is escaping the Earth's gravity well. Figure 3 is an inverted and rotated version of Figure 1, showing how much energy a spacecraft would need to escape the gravity well of Earth.

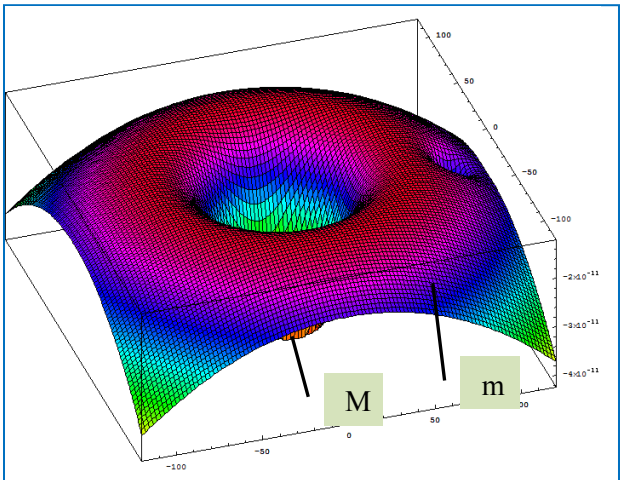


Figure 3: Gravity Wells in a 2-Body System
Synodic, CM system. The graph intuitively shows what a spacecraft must do that exits the system. It expends energy climbing out of earth's gravity well.

Figure 4 shows the very top portion of Figure 3, the bright pink ring. This enlargement shows the benefit of going past the moon on the way out of the system, as the moon's placement causes a depression in the total amount of energy needed to get out of Earth's gravity well.

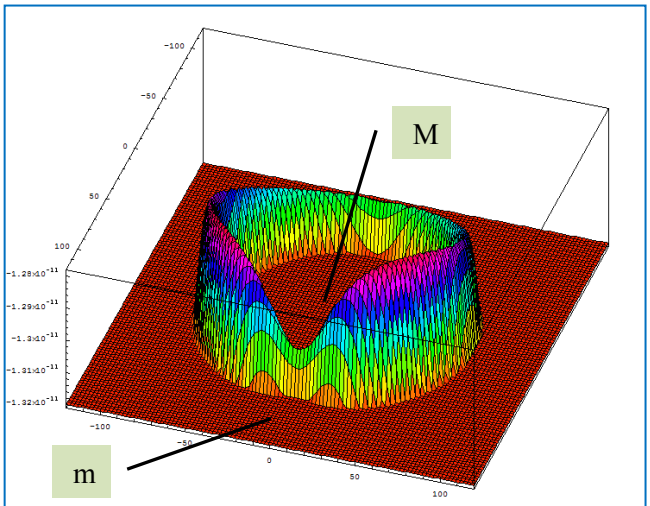


Figure 4: Moon's Effect on Decreased Energy to Leave System
Synodic, CM system. This graph shows the energy needed to overcome the gravity of both planets. As shown, the energy needed near the moon is least.

2.1.4 Lagrange Points

The potential energy maps of the two-body system are extremely important because they later become the basis for calculating low-energy paths. Lagrange points are where all the forces balance, and therefore there is no acceleration on an object at that point. These points have several unique properties, and become the basis for interplanetary space travel. Figure 5 shows the locations of Lagrange points.

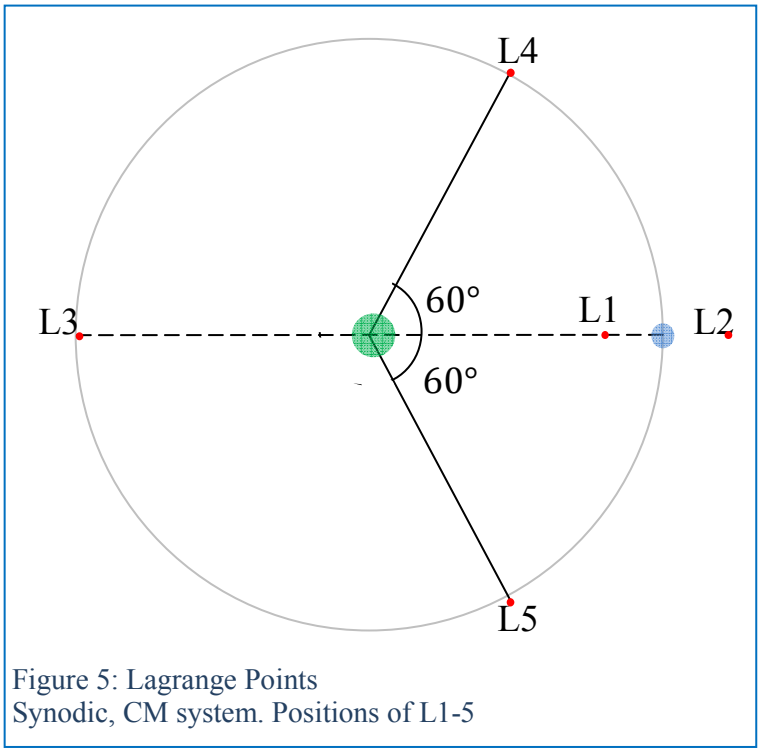


Figure 5: Lagrange Points
Synodic, CM system. Positions of L1-5

Below, we look at the positions and properties of all the five Lagrange points, notated by L and a number, in a two-body system.

2.1.5 L1, L2, and L3

The first three Lagrange points are located on the line between the two massive bodies. L1 is the most intuitive point: it lies between M and m such that the gravity from each body cancels. Usually, an object orbiting M that is closer than m would have an orbital period that is shorter than m, however, m exerts enough force to lengthen the period of the object's orbit and allow it to rotate at the same speed as m.

Similarly, L2 is on the far side of m the same distance away as L1. In this case, m's force is pulling the object along with it, shortening the orbital period to sync with m's.

L3 lies on the far side of M and is a result of both bodies pulling on the point so that it rotates at the same speed as M.

If m is much smaller than M, we can approximate the distance, d, between m and L1 and L2 by the equation

$$d \approx R * \sqrt[3]{\frac{m}{3M}}$$

Where R is the distance between M and m, in our previous example, 100.

Although L3 is not frequently used in orbit planning, L1 and L2 are the primary players in leaving one two-body system to get to another, mostly because of their proximity to the smaller body. As we saw before, the area around the smaller body has a reduced energy requirement to pass into the outer region, putting L1 and L2 directly in the path of an exiting spacecraft.

More importantly, L1 and L2 are highly unstable. This means that the acceleration near these points changes dramatically over very small distances. Therefore, a spacecraft with little fuel would do well to burn its engines at these points because the smallest change in distance has the greatest effect at an unstable Lagrange point. The navigational benefits of these points as well as their proximity to a body that will impart energy to the spacecraft make them perfect for designing low-energy interplanetary paths.

Despite the instability of L1 and L2, it is possible to find a path that circles around these points known as a halo orbit. Since these points have no mass, a spacecraft in a halo orbit is not actually 'orbiting' the point, but rather using the instability of the space to find a very precise path that stays in the vicinity of the point. The planer version of a halo orbit, which is used in this project, is known as a Lyapunov orbit (Contopoulos, 2004). Lyapunov orbits are used extensively in orbit planning as a holding orbit, one that allows the spacecraft to wait to resume its trip until the planets move into a more favorable position. It is identifiable by its distinctive oval shape when shown in a synodic system.

2.1.6 L4 and L5

The last two Lagrange points in a two-body system, although not used in orbit planning, are gravitationally interesting. They lie at the tips of equilateral triangles whose two other endpoints are at the center of mass and the middle of m. They lie in the two 'Potential Energy Depressions' shown in Figure 2. L4 and L5 are unique because they are the only stable Lagrange points. Indeed, objects tend to collect in these potential energy troughs, orbiting the point.

In our solar system, Jupiter's L4 and L5 points are home to the Trojan asteroids. In general, asteroids at L4 are named after Greeks and those at L5 are named after Trojans, accounting for almost all the asteroids near Jupiter's orbit. Other planets exhibit their own collections of moons, asteroids, or dust clouds at their L4 and L5 points.

2.1.7 Forbidden Region for a Specified Energy

By studying potential energy maps, we can determine all the locations a spacecraft cannot get to because it does not have enough energy. When we think of a spacecraft trying to leave earth's gravity well, we intuitively understand that if it does not have enough energy to get over the lip of the well it will fall back down. This concept of falling back down can be applied to the solar system as a whole. We can

view the areas a spacecraft of a particular energy can attain by plotting a potential energy contour map, as in Figure 6. The higher the contour, the more energy a spacecraft would need to be allowed in that region of space.

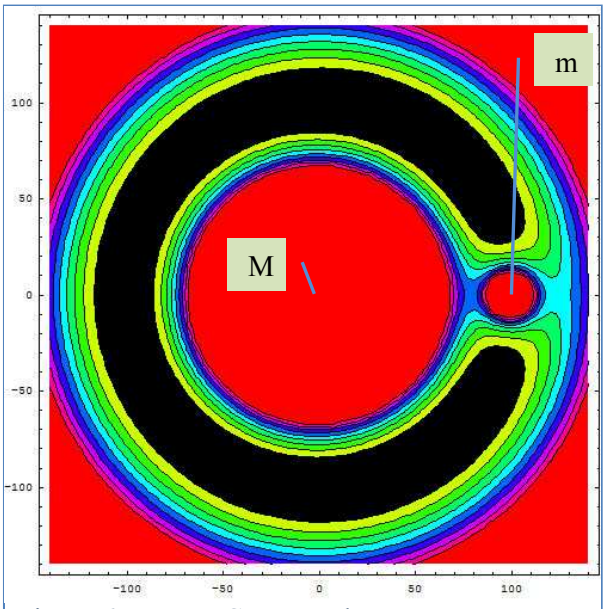


Figure 6: Energy Contour Plot
Synodic, CM system. Shows energy levels required in various regions. The red areas are gravity wells.

Both the circular red regions are gravity wells. The more energy the spacecraft has, the more boundary lines between contours it may cross. For example, a spacecraft with low energy might 'fall back' when it reaches the light blue, while a more energetic spacecraft might attain the yellow region before running out of energy. For this spacecraft, the black region in the picture represents space that it cannot attain- the forbidden region for that amount of energy.

However, if all we need is for our spacecraft to be able to get to the outside red region it is not even necessary to enter the green region. Since the moon lowers the energy requirement for exiting the system, we can slip through with lower energy than would be needed to overcome earth's entire gravity well.

2.2 Mathematical Elements

2.2.1 Jacobi Integral

In a 3- body system (two massive bodies and a spacecraft), it is possible to relate position, speed, and energy, allowing one to be found if the other two are known. This is known as the Jacobi Integral, and is the only known general solution to the 3-body problem (Szebehely, 1967).

When the spacecraft exits Earth's gravity well its velocity decreases while it's potential gravitational and centripetal energy increases, allowing its energy to remain constant. We use a familiar system to express this relation. There are two bodies in this synodic CM system, the larger body with mass M, at $(-m, 0)$ and the smaller with mass m at $(M, 0)$, making the center of mass at the origin. A spacecraft of mass 1 at position (x, y) and with velocity $\dot{x}$ and $\dot{y}$. R is the distance of the spacecraft from the larger body, and r is the distance from the smaller body. n is the mean motion, or $\frac{2\pi}{orbital\ period}$. For some constant energy, C_j,

$C_j = n^2(x^2 + y^2) + 2\left(\frac{M}{R} + \frac{m}{r}\right) - (\dot{x} + \dot{y})$. (Szebehely, 1967)

In other words, the energy equals the centripetal energy, the gravitational energy, and the kinetic energy.

Although it may not be immediately obvious how to use this, it is a powerful tool for constructing low-energy routes because it allows positions to be evaluated for speed and vice versa.

For example: suppose we want to construct an orbit that will spiral onto an unstable Lagrange point, and after an infinite amount of time, will come to rest on the point. Normally, this would be almost impossible because we would have to guess and check different initial velocities and positions and then simulate their paths for an infinite time. However, with the Jacobi integral, we can choose either a starting position or a starting velocity, and then find the corresponding position or velocity that will result in the spacecraft coming to rest exactly on the Lagrange point.

We can solve for the energy constant by using the fact that the velocity at the Lagrange point is zero. Using the point's position, distance from the objects, and zero velocity, we can find C_j. Once we know this constant, it is an easy matter to use our chosen start point's characteristics to solve for velocity. Although slightly more difficult, it is also possible to solve R and r for x and y, and then use a velocity to find the corresponding position.

It is important to remember that the velocity described in the Jacobi integral is the velocity in the synodic CM system, not the total velocity in space. When we solve for a velocity, it is therefore necessary to add back on both the centripetal movement around the center of mass as well as the velocity of the center of mass in the solar system.

2.2.2 N-Body Problem

Beyond the Jacobi integral, a solution to the 3-body problem, there are no further closed-form solutions to problems involving more bodies. This necessitates the need for a timestep integration method. When simulating the movement of a spacecraft, the computer must constantly re-calculate the total acceleration the planets exert on the spacecraft. It also must calculate the force each of the planets exerts on each of the other planets. This allows the planets and the rocket's velocity to be correctly updated as accurately as the timestep will allow.

The N-Body Problem makes a simulation necessary and one that is highly compute intensive. It also renders all the elegant 3-body problem closed form solutions inaccurate, changing the actual positions of Lagrange points, the shape of gravity maps, and the heteroclinic paths. Higher order integration methods become useful to enhance the accuracy of the simulation.

2.2.3 Lagrange Points Move Chaotically

Since there are more than two planets in the solar system, points of zero acceleration in a two-body system come under the influence of the gravity of every other planet. The effects of more planets does not destroy points of zero acceleration, but it moves them. However, since the planets chaotically interact with each other, the higher order coefficients, such as jerk, have value. This means that while it is possible to find a point of zero acceleration, this point will be moving chaotically with the rest of the planets.

For example, Figure 7 shows the effect extra planets have on Lagrange points. Their presence deforms the gravity map, manifesting in asymmetrical spacecraft movement.

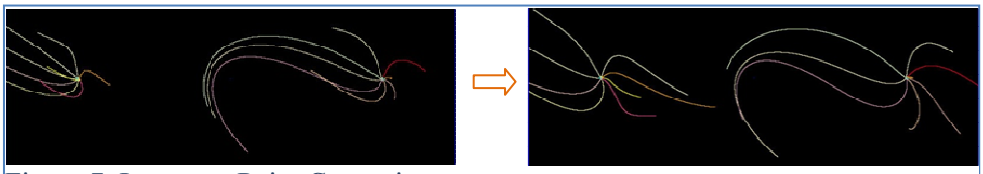


Figure 7: Lagrange Point Correction
Synodic, CM system. Moon-centric with L1 on left. On the left are rockets set in a circle around the Lagrange point as calculated in a two-body system. The rockets on the right were set around the Lagrange point corrected for more bodies, making their orbits more symmetrical.

We notice that the spacecraft on the left do not fan out from the Lagrange point in a symmetrical fashion. On the right, it is more apparent that they are centered on a potential energy saddle point, as their movement fans out from the point.

Having a misplaced Lagrange point can disrupt energy-dependent calculations in many ways. If a spacecraft is at a point where there should be zero acceleration, but where there is acceleration from other planets, then the speed of the spacecraft will be initially changed. When the Lagrange point is adjusted for the gravity of other planets the rest of the energy surface is deformed, but not shifted. This allows the two-body system math to be used as an approximation.

Unfortunately, there is no solution for the chaotic movement of Lagrange points in a system with more than 2 bodies, so approximation such as the one demonstrated in Figure 7 must suffice.

2.2.4 Computing Accurate Planet Positions on an Ellipse

Although it is possible to use an integration method to calculate the movement of the planets, this does not yield a completely accurate result. In order to calculate the true position of the planets at a particular real time it therefore becomes necessary to find an equation for their elliptical orbit.

Primarily, planet ellipse calculations take into account the semi-major axis of a planet's orbit, or it average distance from the body it orbits, as well as the eccentricity, which describes how pronounced the ellipse shape is relative to a circle. Other elements and calculations account for the planet's position in its orbit at a start time and perturbations caused by other planets in their orbit.

For this project, the equations from Paul Schlyter's "Computing Planetary Positions," were used in the program.

2.2.5 Integration Methods

The presence of more than two gravitational bodies in the solar system necessitates an approximation of spacecraft movement. Integration methods calculate the changing acceleration (a), velocity (v), and position (x) of the spacecraft as it moves in time (t). During such simulations, we separate time into small timesteps, where the movement is calculated at each.

The order of the integration method describes its complexity as well as its accuracy. The basic first-order approach we call Newton's Method. In this approach, we calculate the acceleration on the spacecraft at the beginning of the timestep and use this, along with the current velocity of the spacecraft and its position to calculate the position and velocity at the end of the timestep with the basic equations $\Delta x = v_i t + \frac{1}{2} a t^2$ and $v_f = v_i + at$. However, if acceleration is changing rapidly with position (the planets are moving) then this method gives inaccurate results with large timesteps. For the sake of runtime, it is not favorable to run this calculation for every spacecraft every .1 simulated second or so, so instead we try to come up with a more accurate calculation that will give more accuracy for longer time periods.

This is accomplished by the Runge-Kutta Method, which evaluates the acceleration in the middle of the timestep and using it to adjust the velocity mid timestep. The adjusted velocities are weighted into

the final movement. This creates a calculation of the form $\Delta x = \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ Where Δt is the length of the timestep, $v(t,x)$ is the velocity at time t and position x, and

$k_1 = v(t,x)$

$k_2 = v(t + \frac{\Delta t}{2}, x + \frac{\Delta t}{2}k_1)$

$k_3 = v(t + \frac{\Delta t}{2}, x + \frac{\Delta t}{2}k_2)$

$k_4 = v(t + \Delta t, x + \Delta t k_3)$.

In the program, the velocity function also contains Runge-Kutta calculations of acceleration, so this is coupled with a similar equation for Δv, whose k terms are calculated with an $a(t,x)$ function. The derivation of the coupled form of Runge-Kutta used in the program may be found at Hut, Makino, and Heggie's "Integration Algorithms: Exploring the Runge-Kutta Landscape," section 5.2.

Although Runge-Kutta methods involve more calculations, they make up for this extra compute time in accuracy. While the error of Newton's Method is on the order of Δt, with fourth-order Runge-Kutta the error is on the order of Δt^4.

2.2.6 Simulation Accuracy

One frequently asked question is "how accurate is the model?" or "have you compared this model to other programs." The program written for this project uses both Runge-Kutta integration and the elliptical positions of a planet at a given time to make the simulation accurate and realistic. However, since the orbits that are being planned here are unique (not everyone simulates spacecraft going 213m/s on Jan 12, 2010) they are not directly compared to commercial software.

While it is important for the program to be realistic, it is not a perfectly accurate model of the solar system for several reasons. Foremost, the simulation is 2D, meaning that some of the orbital elements of the planets are not taken into account. Also, the moons of the planets as well as asteroids are not included in the simulation. Although these elements are obviously crucial in actually launching a spacecraft, their absence makes spacecraft paths much easier to research, and adding them back in would not add complexity to the algorithm in a significant way.

Although each trajectory is not compared to an outside source, the maneuvers may be compared. Koon, Lo, Marsden, and Ross found a method of transferring to the Moon that uses less energy than the normal method (Koon W. , Lo, Marsden, & Ross, Low Energy Transfer to the Moon, 2001). This maneuver was executed by an early version of the program used for this project and yielded the same result (Cordwell, DeBenedictis, & Lott, 2007), as shown in figure 8.

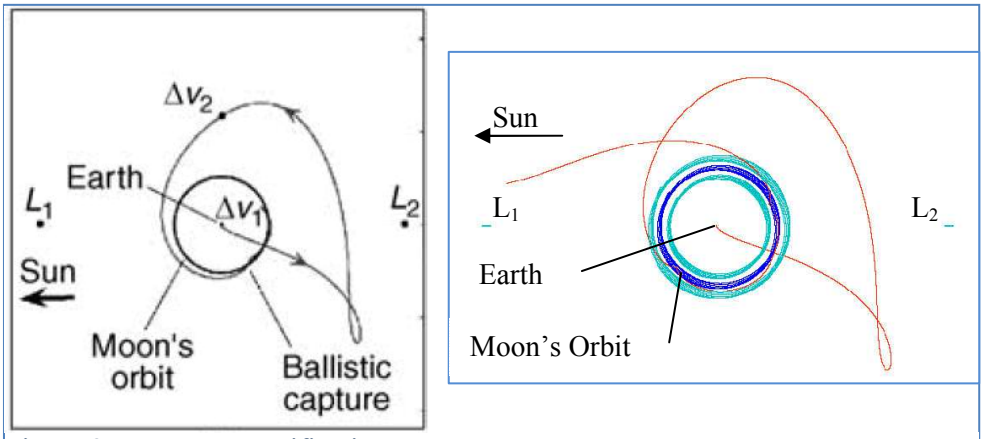


Figure 8: Maneuver Verification
Synodic, CM system. Earth-centric with Earth-Sun L1 on left. Here the program is executing a low-energy maneuver. The results match other studies.

This example shows that although the model is not directly comparable to other software, it may act as a realistic basis for research.

2.3 Existing Methods

Most orbit projection and planning research to date is based on mathematical models describing spacecraft movement in conjunction with methods of simplifying the problem with fewer planetary bodies and approximating the solution with elliptical orbits. This view is hardly surprising, as Poincare's work on the n-body problem was the first major study of how orbits work, and it was largely from a chaos theory point of view.

Currently, there are two major branches of orbit planning. One is the kind NASA uses for their missions, which involve elliptical orbit approximations and corrections and disregard many available low-energy maneuver opportunities. The other is the mathematical study of orbits as differential equations (Olds, Kluever, & Cupples, 2007) or 'invariant manifolds (Koon W. , Lo, Marsden, & Ross, Dynamical Systems, the Three-Body Problem and Space Mission Design, 2000).' The two are separated primarily by a difference in goals: one wants to get to other planets while the other is interested in how it happens.

This research project differs from both these approaches in that while it is mathematically grounded, it employs a computer science prism on the problem of orbit planning, which creates a compromise between the theoretical mathematical basis and the practical goal of navigating the solar system.

2.3.1 Invariant manifolds

As we have seen, the Jacobi integral (page 12) allows us to find positions and their corresponding velocities that create a total constant energy. The collections of these points and their velocities for a particular energy can be called an energy surface. However, unless we restrict our surface to certain initial conditions it would include every point with a velocity vector pointing in every direction. By adding more qualifications, such as 'all orbits that wind onto a Lagrange point,' we get what is known as an invariant manifold.

Like its name, and invariant manifold has invariant energy, but changing position. We can arbitrarily assign it enough conditions that it is a continuous surface, rather than a vector field. Although these manifolds may seem highly mathematical, they actually describe the way we compute trajectories.

When calculating the position of an invariant manifold, we assign 'tracer objects' at intervals along the initial conditions. For example, if we want to see the invariant manifold that starts at one point but has a velocity vector pointing in every direction, we would need to take test cases of the velocity vector direction at intervals with all the tracers at the same place. By simulating the movement of the tracers, we can see the outline of the invariant manifold.

These manifolds are used constantly in orbit planning. Their highly mathematical nature as well as the ease at which they are described by computer simulations make them the perfect medium for simulating constant energy spacecraft.

2.3.2 Heteroclinic Connections

When attempting to find the most energy efficient method of exiting a two-body system, we pick the lowest possible energy level required to maintain a path out of the system. In this case we can see three main regions of space we must pass. In the Earth-Moon system, they are three primary gravitational regions: the Earth's gravity well, the Moon's gravity well, and the Outside region. Although it is easy to launch a rocket so it gets to L1, it is a more delicate matter to get the same rocket to L2. A heteroclinic connection is exactly that: a path between L1 and L2, or a connection between Earth's gravity well and the outside region (Koon W. , Lo, Marsden, & Ross, Dynamical Systems, the Three-Body Problem and Space Mission Design, 2000).

2.3.3 Finding Heteroclinic Connections

To find a Heteroclinic Connection we start by defining two equal-energy invariant manifolds around L1 and L2 that wind asymptotically onto the Lagrange point. We simulate the movement of the tracer particles as they enter the Moon's gravity well.

What we want to find is the intersection of these two manifolds. The intersection will be made of two tracers that have the same position but opposite velocity vectors. By splicing together these two halves of the path between L points, we for a heteroclinic connection.

We find the intersections of the invariant manifolds with a Poincare map. This map is a cross-section of the tracers as they pass by the moon, showing the positions and velocities of tracers from both sides as they pass over this middle line. In a two-dimensional simulation, one dimension of the Poincare map is distance from the moon and the other is speed. Because all the tracers are of the same energy, the direction of their velocity vector is unimportant beyond which side of the moon, in terms of L points, it is going. The Poincare map reveals the hidden relationship between the tracers that were launched, and the intersection of the two invariant manifolds reveals a heteroclinic connection between L1 and L2.

By choosing different two-body systems, we can find heteroclinic connections between the Earth and Moon, and then the Earth-moon and another planet.

2.3.4 Using a Poincare Map to Find the Intersection of Invariant Manifolds

In a two-body system the following method, which we will call the exact method, may be applied to a simulation to discover a trajectory that will lead to a heteroclinic connection. This section describes how the exact method could be used. The basic reason the exact method, even with optimization, does not work for a complex system is that time must be reversible for the exact method to work. Both methods use the Jacobi Integral to set a Lyapunov orbit. The exact method uses a Poincare map to find the intersection of two spacecraft manifolds, while the actual method uses boundaries to track the progress of the spacecraft.

Every time the rockets are moved the program checks to see if they have crossed the boundary line for the Poincare map. These intersections are compiled at the end of the program into a map that shows distance from the moon and velocity. A heteroclinic connection would form when two rockets have the same position but opposite velocity on the Poincare map.

Although scientists such as Martin Lo have achieved heteroclinic connections, they disregarded the other planets in their calculation. Figure 9 shows a heteroclinic connection Martin Lo found using only two planets and the Poincare map method in his paper "Dynamical Systems, the Three-Body Problem and Space Mission Design."

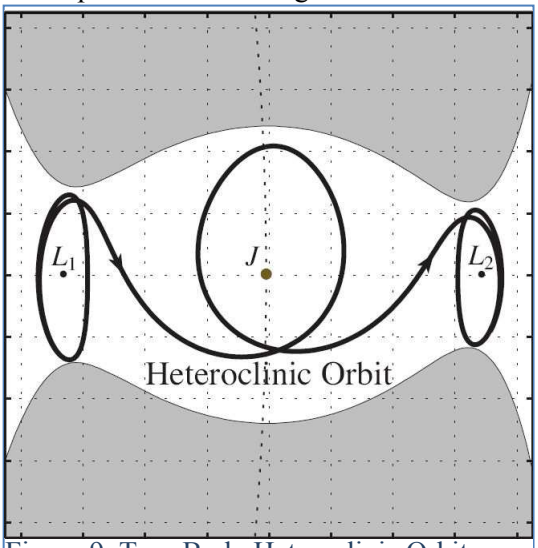


Figure 9: Two-Body Heteroclinic Orbit Synodic, CM system. Moon-centric with L1 on left. A heteroclinic connection found by Martin Lo and the Poincare Map method.

By adding the other planets the Lyapunov orbits become increasingly unstable and difficult to work with. In actuality, the exact method for finding a heteroclinic connection may not be applied to a complex system because the Poincare Map reveals rockets whose paths would connect if time was reversible. Thus, although this method involves optimization, it still does not account for the chaotic nature of planetary movement in the real solar system. However, the concept underlying this process may be applied to itinerary-based optimization, which automatically reveals heteroclinic connections.

2.3.5 Applications of Low-Energy Paths

Although NASA and other space organizations do not regularly use low-energy paths for spacecraft, they have been effective when utilized. The Genesis spacecraft used a series of Earth-Sun L1 and L2 Lyapunov orbits to collect solar wind samples (Koon W. , Lo, Marsden, & Ross, 1999). Another example was the Hiten, a Japanese spacecraft designed to be a relay signals for the Hagoromo. After the Hagoromo failed, a low-energy transfer to the moon was executed, allowing the Hiten to gain moon orbit even though it had 10% less fuel than was traditionally needed (Grayzeck, 2008), making the mission a success.

Other possible applications include the movement of heavy spacecraft, such as unmanned supply crafts for space colonies. Since it requires a large amount of fuel to initially launch such heavy crafts, the spacecraft would have little fuel still available when in space to maneuver, making it the perfect application for gravity-assist propulsion. Since these paths take longer to execute than traditional methods, these supply crafts could be launched even ahead of the initial mission and then wait in orbit around another planet.

3. Novel Method for Finding Spacecraft Trajectories

3.1 Itinerary-Based Optimization

This method is based on a model which takes into account all of the planetary bodies and uses a guided optimization algorithm to identify orbit paths. It does not rely on creating an approximation that works for fewer bodies and then correcting the approximation. In this way, the program is not restrained by elliptical orbit approximations and can easily find more complex non-elliptical orbit patterns.

Below is a picture of the primary boundary lines used to construct itineraries. These lines are supplemented by circular boundaries around planets, which detect when spacecraft go near planets, as well as other boundaries specific itinerary, such as boundaries around the Earth Sun L2 point.

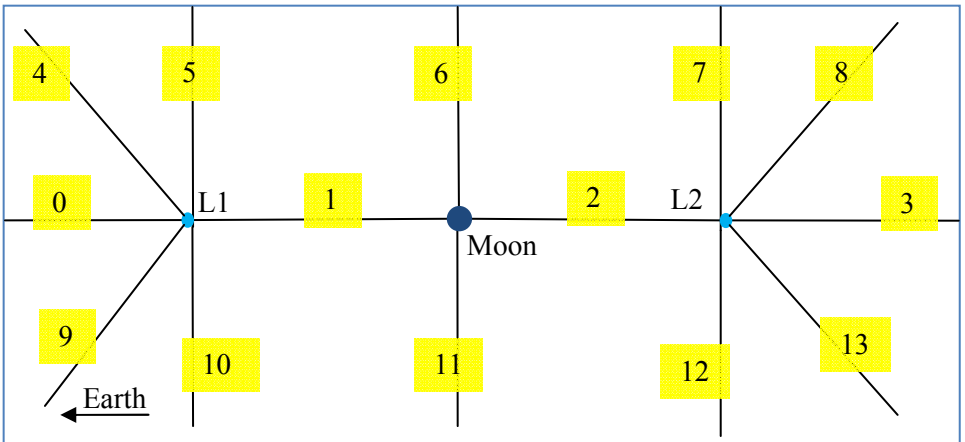


Figure 10: Boundary Positions for Earth Moon System Synodic, CM system. Moon-centric with L1 on left. This is the layout of the boundaries (boundary numbers included) which are used to analyze the paths of the spacecraft.

The program tracks the boundary crossings of each spacecraft so that it can refine on spacecraft trajectories that yield the most correct list of boundary crossings. These crossings make up the itinerary for the flight.

Itinerary is described by a list of spatial boundaries each spacecraft must cross in order. For example, an L1 Lyapunov orbit, shown in dotted green in Figure 11, would satisfy the itinerary 1, 10, 9, 0, 4, 5.

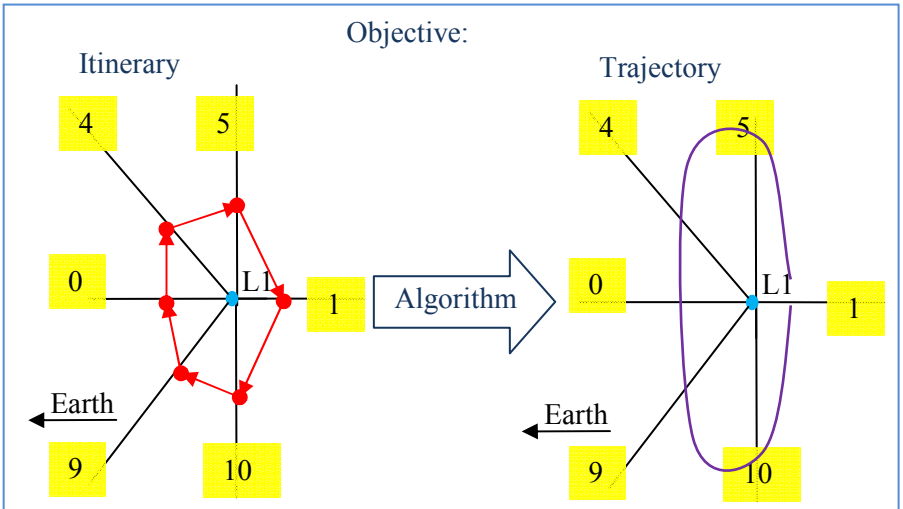


Figure 11: Interpetation of Itinerary
Synodic. L1- centric. The left shows the itinerary as it is described to the computer for computing a Lyapunov orbit. The right shows the actual trajectory the computer would find that meets the boundary specifications.

Since the optimal launch conditions are unknown, the itinerary of boundary lines should be valid for all potentially optimal paths. The objective of the simulation is to identify the optimal trajectory that satisfies the itinerary. In order to do this, the computer uses the boundaries that have been crossed to optimize the initial trajectories.

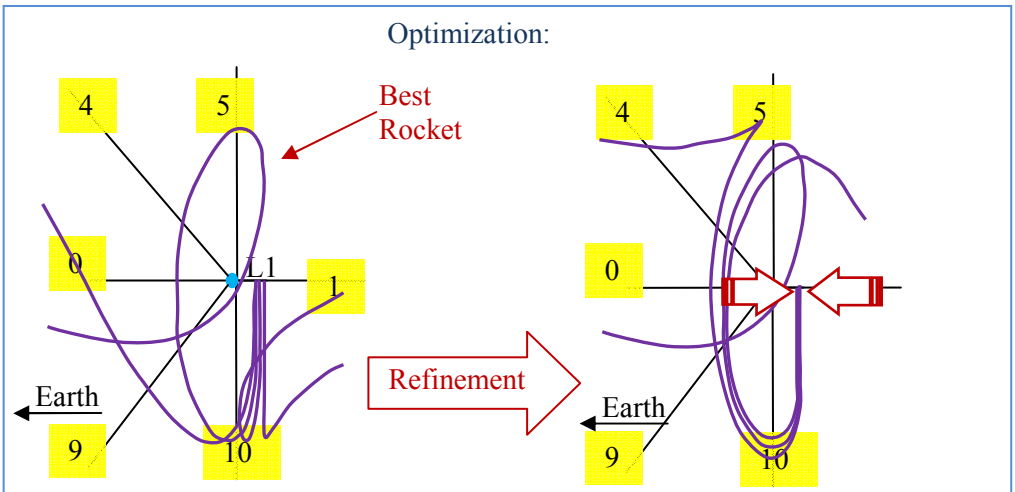
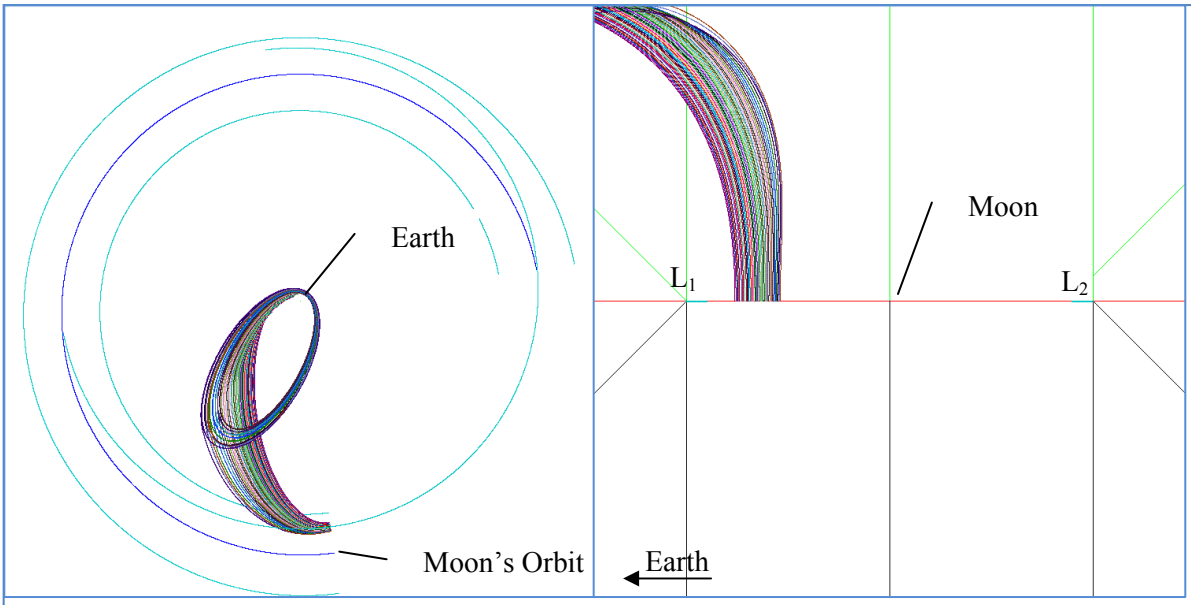


Figure 12: Optimization of Spacecraft Trajectories
Synodic. L1- centric. When the program finds a spacecraft that is close to the specified itinerary, it moves all the other spacecraft closer, thereby refining their trajectories.

By continuously refining the initial positions to fit the requirements for a long list of itinerary, the computer may find the appropriate path.

3.2 Part I- Launch From Earth to a L1 Lyapunov Orbit

Although this is the first step of a trajectory to another planet, it is calculated last, after the rest of the spacecraft paths have been determined. By running time backwards, it is possible to find the spacecraft's intersection with Earth. This section does not require complex itineraries because it takes place exclusively within Earth's gravity well, although it does require a burn. One possible path is shown below.



Not Synodic, Earth-centric showing Moon and L points' orbits.

Synodic, CM system. Moon-centric

Figure 13: Launch into L1 Lyapunov Orbit
This shows one possibly method of going from Earth to L1. The spacecraft are launched, and circle Earth once, do a slight burn right before getting to L1, and enter a Lyapunov orbit.

3.3 Part II- Finding Heteroclinic Connections

3.3.1 Using Jacobi Integral to Set Lyapunov Orbit

The first step in finding a heteroclinic connection is to find a Lyapunov orbit around the L1 point. This is done automatically by the program, by finding the Jacobi integral's value for a range of positions along the L1-Moon line. As spacecraft are refined and reset, the positions and their velocities are recalculated according to the Jacobi integral for their new position. Although the Sun changes the Jacobi integral constant, this can be manually adjusted. Variance between consecutive spacecraft paths in the end result is minimal because the spacecraft are eventually placed so close together.

Figure 12 shows this process of refining a Lyapunov orbit over several simulation cycles.

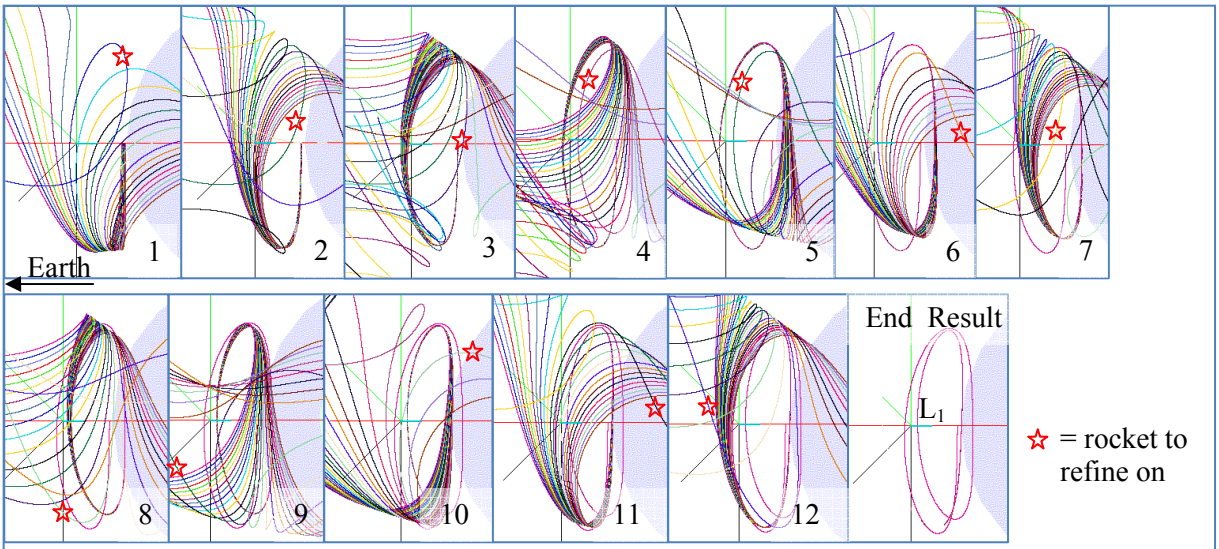


Figure 14: Lyapunov Orbit Iterations
Synodic, CM system. L1 in center. This shows a few iterations in which the computer is automatically adjusting the initial spacecraft conditions to achieve Lyapunov orbits at L1.

Notice that in the last picture there are two Lyapunov orbits, but they do not line up entirely. This is because the orbit of the Earth and Moon are elliptical.

3.3.2 Using Itineraries to find Heteroclinic Connections

We apply the same boundaries to find a heteroclinic connection between an L1 and an L2 Lyapunov orbit.

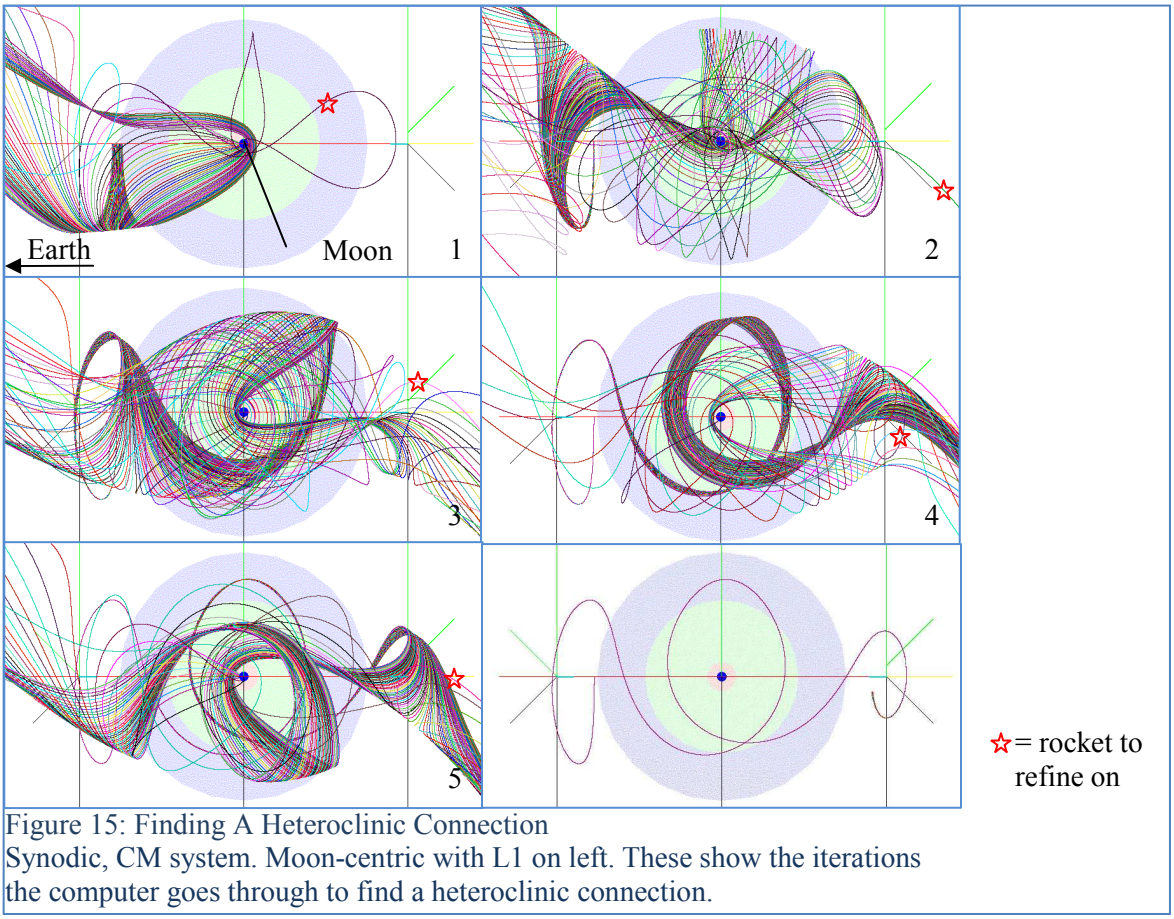


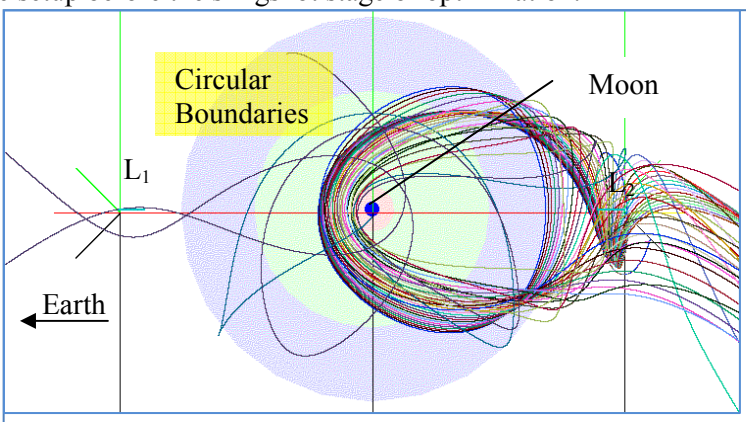
Figure 15: Finding A Heteroclinic Connection
Synodic, CM system. Moon-centric with L1 on left. These show the iterations the computer goes through to find a heteroclinic connection.

Each boundary has a side that it 'favors,' meaning it prefers spacecraft that pass nearer to one endpoint more than the other. This allows for the computer to find paths more quickly.

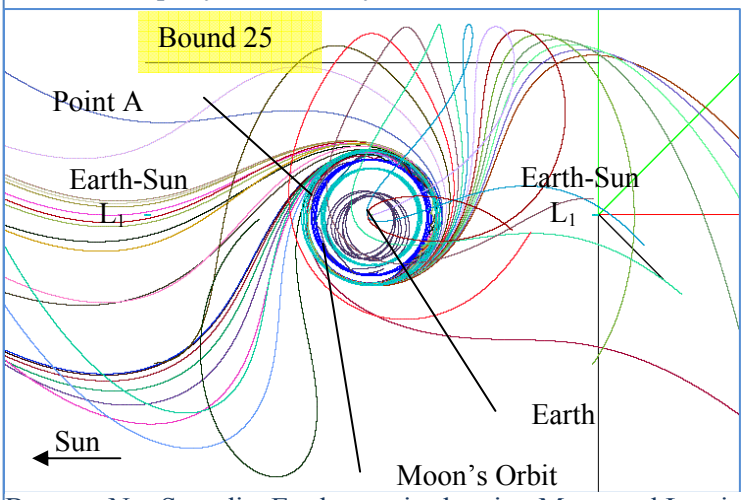
3.4 Part III- Transfer From L2 Lyapunov Orbit to Planet

Once a spacecraft has achieved an L2 Lyapunov orbit, it is outside the strong gravitational pull of the Earth and its movement is largely unconstrained. Before leaving Earth's vicinity it is possible to gain extra energy or maneuverability off the Earth-Moon system. One option is to go into an Earth-Sun L2 Lyapunov orbit, and exploit the instability of this point to gain maneuverability. However, Earth's low mass makes the Earth-Sun L2 weak, and therefore it is better to perform a gravity-assist off the Moon.

In order for this to work, the Moon must be in the correct position when the spacecraft approach it, so they perform an extra Lyapunov orbit before attempting the slingshot. Below are two pictures that show the setup before the slingshot stage of optimization:



Top: Synodic, CM system. Moon-centric



Bottom: Not Synodic, Earth-centric showing Moon and L points' orbits.

Figure 16: Slingshot Setup
The first picture shows the simulation of the second Lyapunov orbit. As these spacecraft leave the Earth-Moon L2 Lyapunov orbit they cross over bound 25, as shown in the bottom picture. As they come back down, they should intersect with the Moon from behind somewhere near point A.

At this stage in the simulation, the program saves all the positions of the spacecraft as they pass over bound 25 and starts re-optimizing their initial positions from there. The program switches from checking for boundaries 1-13, and instead only checks for the circular boundaries around the Moon, shown in the first slingshot picture. By switching boundaries, the program does not need to satisfy the rectangular boundaries as well as the circular ones in some set order, allowing it to get the greatest gravity boost from the moon possible.

This shows the optimization the program performs to find a slingshot off the moon.

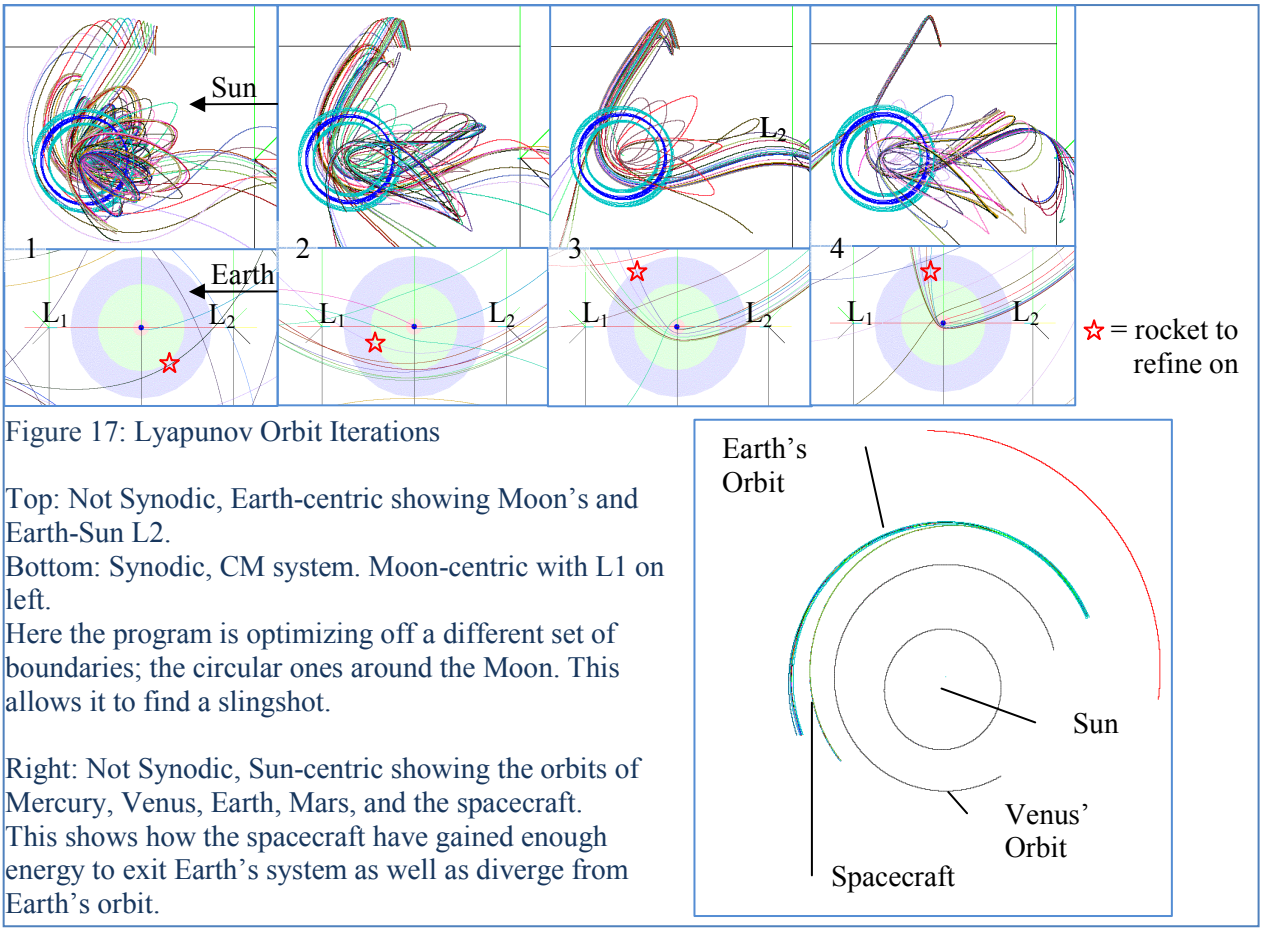


Figure 17: Lyapunov Orbit Iterations

Top: Not Synodic, Earth-centric showing Moon's and Earth-Sun L2.
Bottom: Synodic, CM system. Moon-centric with L1 on left.
Here the program is optimizing off a different set of boundaries; the circular ones around the Moon. This allows it to find a slingshot.

Right: Not Synodic, Sun-centric showing the orbits of Mercury, Venus, Earth, Mars, and the spacecraft.
This shows how the spacecraft have gained enough energy to exit Earth's system as well as diverge from Earth's orbit.

The program has successfully optimized the positions of the spacecraft so they pick up energy from the Moon and exit the system with considerable energy of their own to diverge from Earth's orbit.

3.5 Effect of Spacecraft Energy

The amount of energy a spacecraft has determines the size of the Lyapunov orbit and shape of the heteroclinic connection. Below is a picture of several heteroclinic connections with the same shape and at the same time, but with different amounts of energy. This slight energy change results in a difference in the size of the Lyapunov orbit.

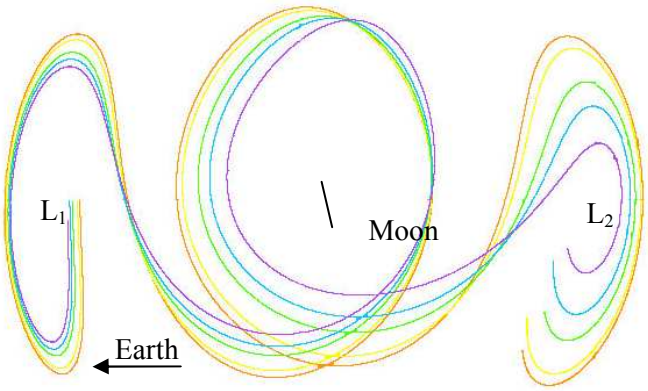


Figure 18: Heteroclinic Connections with Slightly Different Energy
Synodic, CM system. Moon-centric with L1 on left.
This shows multiple heteroclinic connections that have slightly different energy. As the energy decreases, the Lyapunov orbits on the sides become smaller and the Moon fly-by becomes increasingly asymmetrical.

Larger changes in energy can result in changes in the shape of the heteroclinic connections that may be achieved. Too little energy may result in an inability to achieve a maintainable L2 Lyapunov orbit.

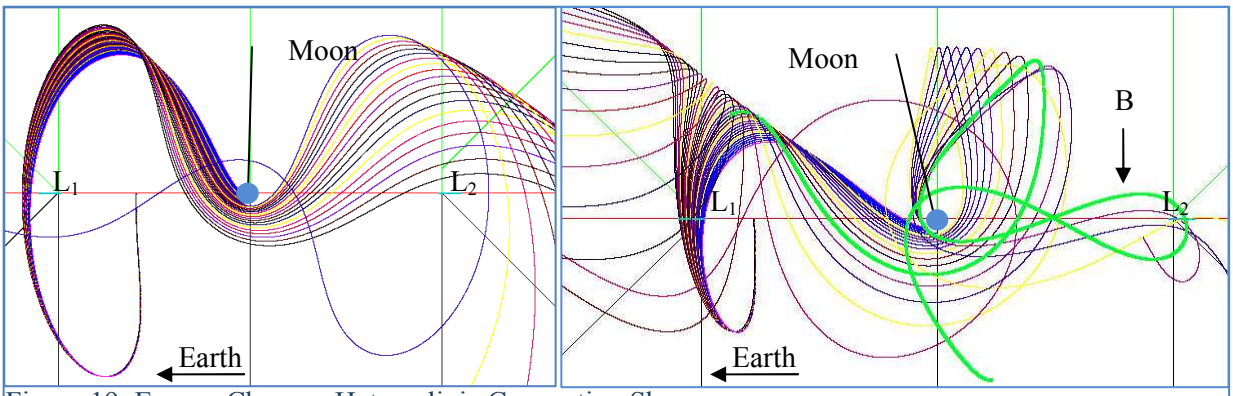


Figure 19: Energy Changes Heteroclinic Connection Shape
Synodic, CM system. Moon-centric with L1 on left.
Both pictures are simulations taking place at the same time index.
The left picture shows a simulation that has too much velocity energy. The spacecraft do not need to gain the available energy from the Moon.
The picture at right shows a simulation that does not have enough energy: here rocket B skimmed the Moon's surface but yet was not able to complete a halo orbit around L2, falling back into the system.

Since the velocity is set off of the Jacobi integral, which does not take into account the gravity of the sun, the initial energy must be adjusted either up or down. After the Sun's gravity is accounted for, it can lower the energy requirement for the spacecraft even below the system's Jacobi integral constant for L2. The simulations achieve the lowest energy possible when they are lined up so that the sun helps pull them out of the Earth Moon system.

There are other heteroclinic boundary patterns that could be used, but they require more energy. For example, Figure 20 shows and alternate boundary pattern, where the spacecraft do not fully circle the moon.

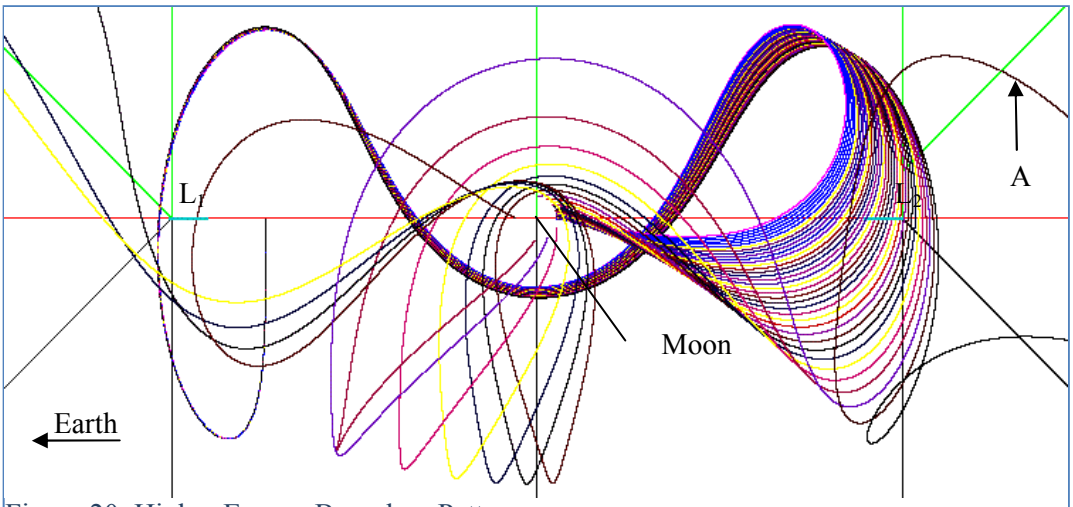


Figure 20: Higher Energy Boundary Pattern
Synodic, CM system. Moon-centric with L1 on left.
Spacecraft A is has left the system after only a half pass by the moon.

Although this is a path out of the system, it requires more energy because the spacecraft do have as much assistance from the moon. This simulation also shows the dynamic nature of the program, as it can be applied to various different orbit patterns.

4. Computation Structure Analysis

4.1 Program Structure

The program is written in C++ to give full access to the capabilities of multithreaded optimization for this compute intensive problem. The "DOS screen" approach seperates compute from interface, which minimizes load imbalance of cores. The following will act as a summary of the basic program functions and as a guide to the code itself. Function and variable `names` have been emphasized to allow the reader to better compare with the code, which may be found in the appendix.

The program is separated into several parts. The major sections are as follows: setting the positions of the planets, initializing the rockets, moving the rockets and planets, analyzing the positions of the rockets, and displaying data. Figure 19 shows a flowchart of the program, and each of these major functions will be described below.

All functions of the program act on three major data structures; `body`, which describes the planets, `rocket`, which describes the spacecraft, and `Lagrange`, which describes Lagrange points. All three contain position (pos) and velocity (v or vr for rockets) data. Planets and rockets also have synodic positions in some cases. Rockets also carry data such as which Lagrange point they were launched from, the color they are printed in, if they have crashed into a planet, which Lagrange point they are currently closest to, their original launch position and velocity, and if they have used their rocket boosters.

- The planet position calculator uses an actual date to set the parameters for the ellipse function, some of which change over time, such as mean anomaly. `niweam` is called to re-set these parameters for each planet according to the day, and then `CalcPlanetPos` uses these parameters to calculate the position on an ellipse. Because of the later use of inter-timestep positions in the Runge-Kutta 4 calculation, pos[1]-pos[3] are set for each planet. At the start of the program some of the planets are set a few extra times to discover their velocities, allowing the Lagrange points to be calculated with the `CalcL12Point` and `CalcL45Point` functions.

- Setting the initial positions of the rockets is perhaps the most complex action of the program. `LAUNCHLOOKUP`, defined at the beginning of the program, describes the method of launch. For a launch around Earth, `LAUNCHLOOKUP` is less than 1, and the rockets are set tangent to a 200 km parking orbit. For finding a heteroclinic connection, the rockets are set in a Lyapunov orbit around the Lagrange points. This method brings to bear the energy-dependent math that this report has discussed. The planet's positions are converted to synodic coordinates with `BodyLgConvertSynodic` and the function `lghalolaunch` sets the rockets in a Lyapunov orbit around both Lagrange points. To get this Lyapunov orbit, the program runs several times, refining the initial guess of position for the rocket to go around the most times. The nature of a Lyapunov orbit is that it passes perpendicular to the line between the Lagrange point and the moon. The program picks a base distance away from the moon (`hsd.synodiccpl1 &2`) and a distance between each rocket (`hsd.changedistl1 &2`) and sets the rockets on the line between the Lagrange point and the Moon (hsd stands for halo set data). It calls the function `jacobiveloofpos`, which uses the Jacobi integral to find the appropriate velocity for that point with the constant set for being zero at L2. All of these energy-dependent calculations have been done in synodic coordinates, and they are converted back into sun-centered rectangular coordinates with `RocketConvertRec`. If `LAUNCHLOOKUP` is more than 5, the program loads the saved positions and velocities of previously used rocket, which will be explained later. If it is 6, it reloads positions from some intermediary point into a new iteration through `datasave`, also to be explained later.

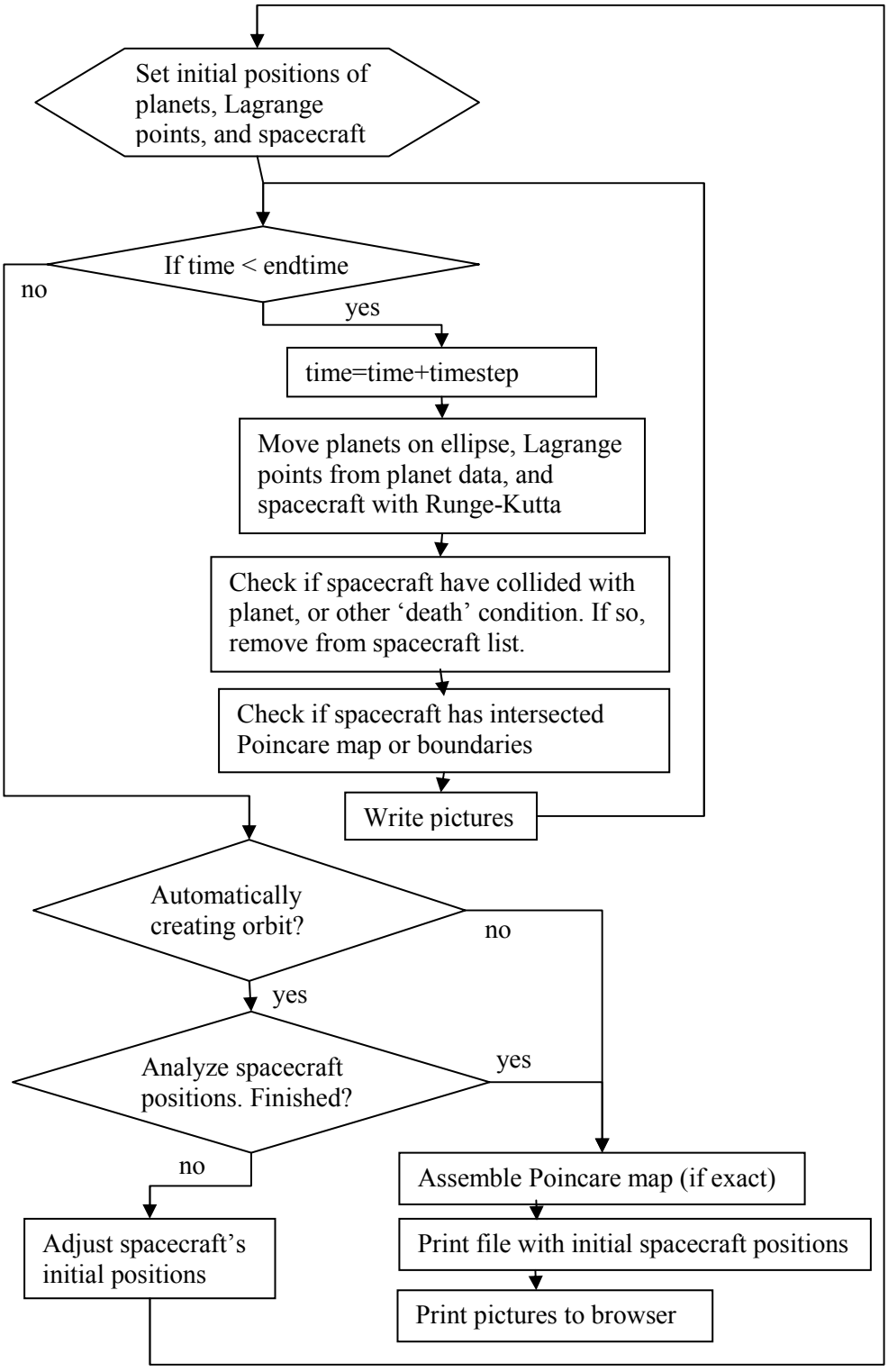


Figure 21: Program Structure

This flowchart shows the general structure of the program. Depending on how the program is being used at the time, actions such as setting the spacecraft's initial positions and analyzing their final positions may be changed according to the current goal of the program.

- The rockets are moved using a Runge-Kutta algorithm (page 14) and the planets are re-calculates using the same method for each timestep. The timestep itself is varied depending on how close the rockets are to planets; the closer they are, the more resolution the program provides by decreasing the timestep.

- The program prints out the date, the timestep, and general information into command prompt every few simulated days. The function `drawpic` draws pictures with a variety of options: synodic or not, centered or de-rotated to any planet, with variable magnification. These pictures are saved using `GifImageStoreAdd` and printed in a html page. The interface between the browser and the program is largely contained in `Osim.cpp`, and not relevant to the functioning of the simulation. The program also converts the final positions of the rockets into synodic coordinates and prints them to a file. The rocket number, position, velocity, `launchlookup`, and time are printed within curly brackets. This file can be used to pick a simulation up where if left off, as `readfilerock` can take the file and restore the positions, velocities, and time of the simulation to those described by the file. This allows the program to separate an longer orbit into pieces which are easier for the computer to identify and design without human intervention.

4.1.1 Region Method Program Structure

One major function of the program is to automatically adjust the original parameters so as to get a better rocket configuration. This is done in two ways. The system, uses `hsd,` contains the information needed to set the rockets, and the program modifies it after each run automatically.

After all the timesteps have been calculated, `PoincareFind` compiles `PoincareData` to create a Poincare map. `resethaloparam` analyzes the rockets final positions to decide where to set them for the next run if the program is dynamically creating an orbit with specified itinerary. In the case of looking for a Lyapunov orbit, the program analyzes which gravitational section the rocket ended up in. It looks for the place where the rockets diverged, some going into one gravitational region and the others to a different one, as shown in Figure 20. `halofate` keeps track of this. If it has already exhibited itself in the moon region, this parameter is already set, otherwise its distance from the earth is analyzed. If the rocket is closer to the earth than is L1, then it is in the moon region. If farther than L2 it is in the sun region. The function then goes through the `halofate`s of all the rockets, in order of how they were set on the line. It finds the mid-point between the two desired rockets, sets `hsd.synodiccpl1`to this desired new center point, and decreases `hsd.changedistl1`. This makes the launch area closer to a Lyapunov orbit that orbits several times. The program repeats until `hsd.changedistl1` is less than a desired amount, or that the rockets are set so close to each other.

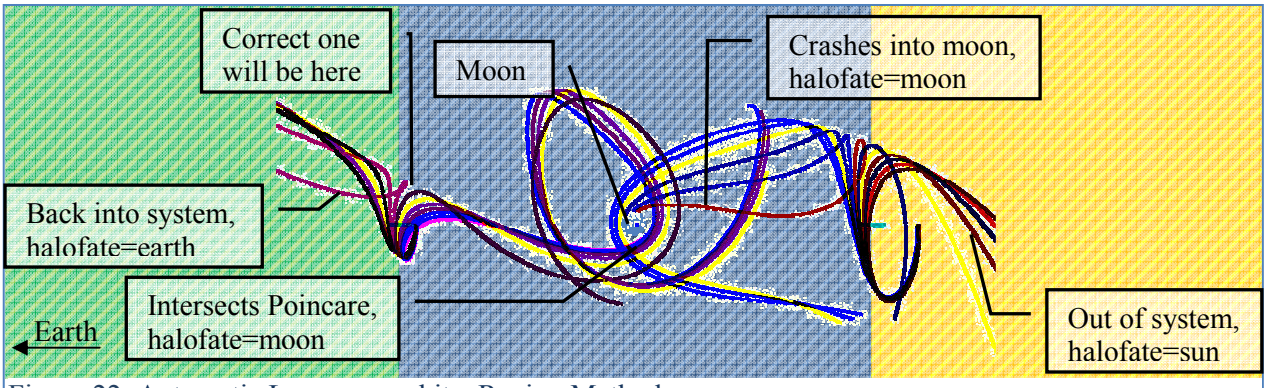


Figure 22: Automatic Lyapunov orbits- Region Method
Synodic, CM system. Moon-centric with L1 on left. Here the computer has completed a few iterations and is trying to find a Lyapunov orbit. The green, blue, and yellow show general gravitational regions.

`PoincareMap` checks to see if a rocket has crossed the boundary line that is perpendicular to the line between the moon and the earth at the moon. If so, it records the position and velocity of the rocket when it passed in `PoincareData`, to be used later to construct the Poincare map. Depending on the function of the program, it may check the rocket's position in relation to some target position or if it is straying too far from the moon. It checks to see if it is within the radius of a planet, and if so sets the rocket's `deadbit` so it is no longer calculated. In the case of finding a Lyapunov orbit, the program sets the rocket's `halofate` to the moon if it crashes into the moon or passes the midline.

Although this system is automatic, it does not provide the computer with enough information to successfully pick the correct spacecraft. For example, when trying to find an orbit around L1 (Figure 21) the method would fail to identify the best spacecraft if some went around the moon and then continued to Earth because only final positions are analyzed.

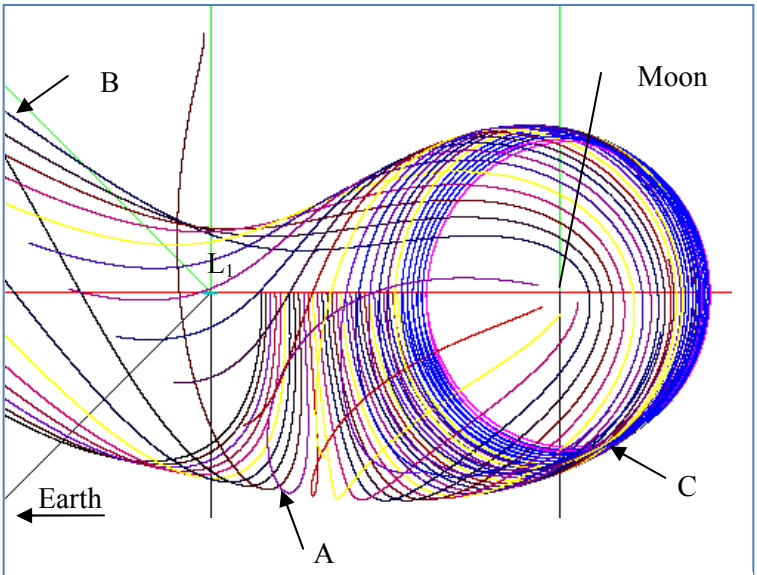


Figure 23: Region Method Error
Synodic, CM system. Moon-centric with L1 on left. Although the correct point for a spacecraft to pass through to get to a Lyapunov orbit is at point A, the region method would optimize toward point C because of the spacecraft at point B.

The following method solves this problem by focusing on where the spacecraft paths have gone rather than where they end.

4.1.2 Itinerary Method Program Structure

The second method of automatic optimization involves checking to see if spacecraft have crossed boundaries and then analyzing the order of boundaries they have crossed.

Boundaries are defined in terms of their relationship to planets, and are stored as a `line1 bound#,` which describes their position on that timesetp in the form Ax+By=C. Each rocket has an array that stores the 'side' (as in 1 or -1) of the boundary it's on. `Boundarysidefunc` is called on each timestp, and checks to see if the rocket has changed sides, or passed over, any of the boundaries. If the rocket has passed over a boundary it sets its `setcrossdata` with the boundary crossed, position, velocity, and time.

`Resethaloparam2` initially refines the `hsd` structure in the same way that the region method does, but because the boundary method requires twice as long a simulation time it must reload the spacecraft's positions halfway through the simulation to preserve accuracy. Thus, halfway through the

simulation it uses `datasave` to record the positions of the spacecraft from some intermediary point, and then optimizations are executed by moving the spacraft closer to the best rocket. In `resethaloparam2` the boundary crossings of all rockets are analyzed in order- all the first crossings, the second crossings, and so on. The best rocket is defined as the first rocket that triggers the most of a specified pattern of boundaries in order.

This wide variety of tools creates a powerful program which can dynamically find a variety of orbit patterns.

4.2 Parallel Code Analysis

In order to simulate more complex and accurate maneuvers the program has been adapted for use on multi-core processors. Unfortunately, this program is relatively difficult to parallelize because the 10 bodies must move between each calculation of the movement of the spacecraft. Also, movement of spacecraft and refinements on their initial positions are cumulative, each depending on the previous steps. Both of these restrain the parallel section of the code to be the movement and force calculations of each rocket with all the planets, reducing the amount of parallel steps available.

For presentation and debugging purposes, the program writes the positions of the spacecraft on pictures each timestep, taking up a significant amount on non-parallelizable time. If this was adapted for standard use, data would be exported in some other less compute intensive form for later analysis. Therefore the performance analysis was done without writing as many pictures.

However, despite these constraints, the program still spends a large portion of its time calculating the movement of the spacecraft and planets, so any speedup in this area is much needed and shows an effect on the runtime of the model. In the parallel model, the planets' positions are calculated on separate processors, the program synchronizes, and then the spacecraft are calculated on separate processors. From 1 to 4 cores, the speedup is about 3. However, this changes with the area being simulated for several reasons.

1. The lower the timestep the more often the program must re-calculate the positions of the planets and then synchronize, which results in wastes cycles. One possible method to largely remove this bottleneck would be calculate planetary positions for many timesteps ahead of time. However, this would mean that the timestep would have to mesh with the available planet positions. Some compromise between the two might result in higher speedup.
2. For some simulations, such as trying to find Lyapunov orbits, many of the spacecraft crash into the moon. This results in a lowered computer time for the processor that is responsible for the movement of these spacecraft. By dynamically creating lists of spacecraft names, this problem could be avoided.

Despite these bottlenecks, running the program on four cores results in increased performance. If these issued were addressed, the program would show further speedup with more cores.

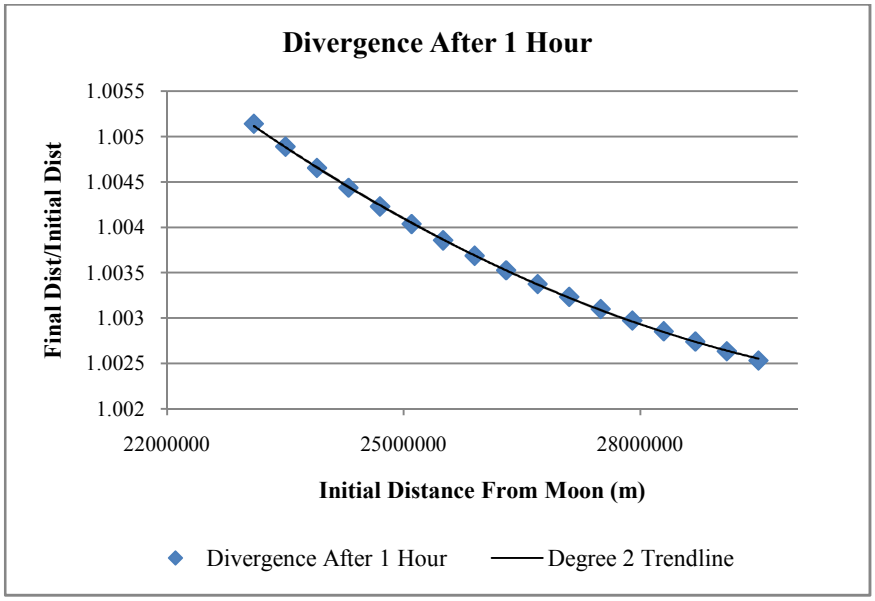
4.3 Algorithm Analysis

This section examines the effectiveness of the algorithm, which is dependent on the specifics of the problem being calculated.

The basic principle behind the itinerary based algorithm is that spacecraft at different positions will experience divergence in their positions over time because of the differences in gravitational forces acting upon them. The algorithm projects the divergence of many spacecraft, and then selects the spacecraft whose path was most correct in terms of satisfying the sought itinerary. The more divergence associated with the region of space an object is traveling through, the more opportunity for low-energy maneuvers, and the more 'work' the computer must perform to correct for and exploit this divergence.

For two spacecraft with the same velocity vector but different initial positions, $divergence = \frac{difference\ in\ final\ positions}{difference\ in\ initial\ positions}$, over some time period. Since divergence is caused by a difference in force acting on an object, it is logical to assume that when these forces are high, as they are near planets, the

divergence will also be high. This was experimentally tested by setting spacecraft along the Earth-Moon line with the same velocity and measuring their divergence.



As shown, when distance from the Moon increases, the divergence decreases. The trendline shows that this seems to happen at a d^2 rate in terms of distance from the planet.

The time a specific itinerary takes to calculate is primarily dependent on two things:

1. As spacecraft travel closer to planets, the program decreases the timestep in order to maintain accuracy, causing each iteration to be more compute intensive and take longer to calculate.
2. Traveling through divergent regions makes the program require more iterations, also taking up more compute time.

This means that going near planets results in iterations that are more numerous and compute intensive. Ironically, low-energy paths thrive in exactly these areas, perhaps explaining why they are so difficult to calculate.

What does this say about the performance of the program? Since the paths of spacecraft are described by itineraries, the entire route must be dependent on the divergence in the areas around the itinerary points. For example, requiring spacecraft to pass over a boundary part way to Jupiter in interplanetary space presumably will not add much compute load to a fixed-time simulation. Conversely passing a few kilometers over the surface of Jupiter, where forces are high, would be very compute intensive. Of course, most of the boundaries are very near the Moon, and therefore in divergent space, and it is not known before hand which part of the boundary will be crossed. However, in theory, it should be possible to create a ‘worst-case scenario’ for how long the program would run given the itinerary specified. This would be based off the product of the spacing between two itinerary points and the highest divergence possible to satisfy both points.

5. Conclusion

An algorithm for automatically finding gravity-assist spacecraft paths has been developed and used in conjunction with a solar system model that takes into account all of the planets. This novel itinerary-based algorithm optimizes spacecraft launch conditions to achieve effective gravity-assist trajectories. It has successfully simulated Lypanov orbits, heteroclinic connections, and moon gravity assists, escaping the Earth's orbit with no burns. The program utilizes multi-core processors for optimal performance.

6. Acknowledgements

I would like to thank my mom, for helping me show the excitement of science to everyone.

My dad has always been patient enough to help me develop my programming skills. You're amazing Dad!

My teachers, especially Mr. Carrie, Mrs. McKernan, Mrs. Sena, and Mr. Carry, have been an amazing support to my continuing interest in science.

To my school, Saint Pius X High School, in general, for being so great!

7. Bibliography

Contopoulos, G. (2004). *Order and Chaos in Dynamical Astronomy.* New York: Springer.

Cordwell, K., DeBenedictis, E., & Lott, B. (2007, April 4). *Optimization of Trajectories.* Retrieved May 1, 2007, from Supercomputing Challenge: http://www.challenge.nm.org/archive/06-07/finalreports/38.pdf

Courant, E., Olbert, S., Bennett, W., Herget, P., Whipple, F., Ehricke, K., et al. (1959). Orbit Theory: Proceedings of Symposia in Applied Mathematics. *Amerian Mathematical Society* .

Grayzeck, D. E. (2008, April 2). *NASA- NSSDC- Spacecraft Details*. Retrieved May 8, 2008, from NASA: http://nssdc.gsfc.nasa.gov/nmc/masterCatalog.do?sc=1990-007A

Hut, P., Makino, J., & Heggie, D. (2007, September 14). *Integration Algorithms: Exploring the Runge-Kutta Landscape.* Retrieved January 8, 2008, from The Art of Computational Science: http://www.artcompsci.org/kali/vol/runge_kutta/title.html

Koon, W., Lo, M., Marsden, J., & Ross, S. (2000). Dynamical Systems, the Three-Body Problem and Space Mission Design. *Control and Dynamic Systems* (pp. 1167-1181). Pasedena: World Scientific.

Koon, W., Lo, M., Marsden, J., & Ross, S. (2001). Low Energy Transfer to the Moon. *Celestial Mechanics and Dynamical Astronomy* , 63-73.

Koon, W., Lo, M., Marsden, J., & Ross, S. (1999). The Genesis Trajectory and Heteroclinic Connections. *Astrodynamics* , Vol. 103, Part III, 2327-2343.

Lo, M. (2008, May 1). Interview With Martin Lo. (E. DeBenedictis, Interviewer)

Lo, M. (2002). The InterPlanetary Superhighway and the Origins Program. *Aerospace Conference Proceedings* (pp. 7-3543- 7-3562 vol.7). Pasadena: IEEE.

Olds, A., Kluever, C., & Cupples, M. (2007). Interplanetary Mission Design Using Differential Evolution. *Journal of Spacecraft and Rockets* , 1060-1070.

Schlyter, P. (n.d.). *Computing Planetary Positions*. Retrieved 11 10, 2007, from Paul Schlyters: http://stjarnhimlen.se/comp/ppcomp.html

Szebehely, V. (1967). *Theory of Orbits.* New York/London: Academic Press.

8. Appendices

8.1 Mathematica Equations

"Potential Energy" and "Lagrange Points":

Plot3D[$\left(\frac{99*6.67*10^{-11}}{(-1-x)^2+y^2+.00001}+\frac{6.67*10^{-11}}{(99-x)^2+y^2+.00001}+.5*10^{-15}*\left(\frac{x^2+y^2}{2}\right)\right)$, {x, -120, 120}, {y, -120, 120}, PlotPoints -> 110, ColorFunction -> Hue, Lighting-> True, ViewPoint-> {-40, -120, 80}, ImageSize -> 700]

"Lowest Potential Energy Depressions":

Plot3D[Min[$\left(\frac{99*6.67*10^{-11}}{(-1-x)^2+y^2+.00001}+\frac{6.67*10^{-11}}{(99-x)^2+y^2+.00001}+.5*10^{-15}*\left(\frac{x^2+y^2}{2}\right)\right)$, $2.6*10^{-12}$], {x, -120, 120}, {y, -120, 120}, PlotPoints -> 110, ColorFunction -> Hue, Lighting-> True, ViewPoint-> {-150, 40, -50}, ImageSize -> 700]

"Gravity Wells in a 2-Body System":

Plot3D[-5*$\left(\frac{99*6.67*10^{-11}}{(-1-x)^2+y^2+.00001}+\frac{6.67*10^{-11}}{(99-x)^2+y^2+.00001}+.5*10^{-15}*\left(\frac{x^2+y^2}{2}\right)\right)$, {x, -120, 120}, {y, -120, 120}, PlotPoints -> 110, ColorFunction -> Hue, Lighting-> True, ViewPoint-> {-40, -120, 80}, ImageSize -> 700]

"Moon's Effect on Decreased Energy to Leave System":

Plot3D[Max[-5*$\left(\frac{99*6.67*10^{-11}}{(-1-x)^2+y^2+.00001}+\frac{6.67*10^{-11}}{(99-x)^2+y^2+.00001}+.5*10^{-15}*\left(\frac{x^2+y^2}{2}\right)\right)$, $-1.32*10^{-11}$], {x, -120, 120}, {y, -120, 120}, PlotPoints -> 110, ColorFunction -> Hue, Lighting-> True, ViewPoint-> {90, 30, 70}, ImageSize -> 700]

"Energy Contour Plot":

CountourPlot[Min[$\left(\frac{99}{(-1-x)^2+y^2+.00001}+\frac{1}{(99-x)^2+y^2+.00001}+2*10^{-6}*\left(\frac{x^2+y^2}{2}\right)\right)$, $2.7*10^{-2}$], {x, -140, 140}, {y, -140, 140}, PlotPoints-> 100, ColorFunction -> Hue, ImageSize -> 500]