A Novel Tool for Multi-Gravity Assist Trajectory Optimization

Tom Ginsberg, David Black, Bereket Guta, Jeremy Wong, and Calum Macdonald
University of British Columbia
Applications of Classical Mechanics
Professor: Kris Sigurdson
(Dated: April 2019)

Space is the final frontier, and exploring its vast reaches is one of the greatest challenges mankind has undertaken. However, moving around in our Solar System requires frequent, large velocity changes, which in turn require expensive and heavy fuel, rendering the whole process almost economically infeasible. Because of these costs, it is advantageous to explore more efficient methods of moving around the cosmos. In this project, one such method of maneuvering in space, called a gravity assist, will be examined. Gravity assists allow a spacecraft to use a planet's kinetic energy to its advantage, obtaining desired velocities by interacting with the planet's gravitational field rather than wasting fuel to thrust. Here we treat the objective of flying from one planet to another as an optimization problem and search for the spacecraft trajectory involving any number of gravitational assists that minimizes the required fuel for a given start and destination planet. To determine the optimal trajectory, software was be developed, incorporating a solver of Lambert's Problem, a Genetic Algorithm, and a fitness function for each planetary interaction, as well as an animation to visualize the trajectory. The spacecraft is assumed to use a chemical engine, so its maneuvers can be treated as impulsive. In between maneuvers, the spacecraft follows the natural conic section trajectory only under the influence of the sun's gravity. The spacecraft's overall trajectory can be formed by patching together these conical orbits from the different phases of the transfer.

Contents

I. INTRODUCTION

In the 1970's, Voyager 1 brought its payload into space at a cost of close to $1M per kg. While this price has dropped substantially with improved technology, the cost of bringing 1kg of payload into space is still approximately $10k - $20k. Considering the huge amounts of fuel required for long distance interplanetary travel, this makes any such missions prohibitively expensive. In the past we have therefore only sent light probes and spacecraft with no intention of returning or carrying payloads such as people to other planets. In order to expand the horizons of human space travel, it is essential to explore more efficient methods of flight. A particularly useful and promising technique is called the gravitational assist.

In a gravity assist, a spacecraft enters a planet's sphere of influence (SOI) at a certain velocity relative to the planet. Depending on the magnitude of the velocity and how close it takes the spacecraft to the surface of the planet, the spacecraft will move through a certain hyperbolic orbit around the planet, changing its direction but not the magnitude of its velocity relative to the planet. However, in the sun's reference frame the outgoing velocity of the spacecraft is now essentially the magnitude of the incoming velocity in the new direction plus the velocity of the planet. Though this gain in energy may seem to violate conservation laws, one must consider the planet-spacecraft system as a whole, in which case it becomes apparent that the spacecraft is tapping into the planet's orbital energy to achieve its velocity change. Since the spacecraft is small relative to the planet, this change is negligible for the planet but can have a great impact on the spacecraft.

II. PROBLEM STATEMENT

By making specific combinations of gravity assists with various planets it is possible for a spacecraft to attain any number of interesting and beneficial trajectories to efficiently travel from one planet to another. However, without further constraints this problem has an enormous solution space. In order to find the best possible trajectory in terms of fuel consumption, one would have to consider which planets to rendezvous with, the times at which these maneuvers should occur, exactly where and at what angle to enter each planet's sphere of influence, and when to thrust while inside the planet's SOI.

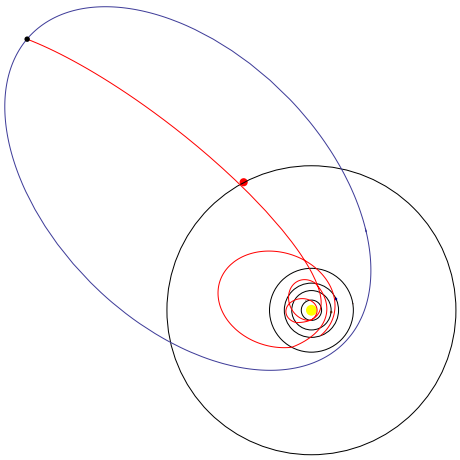

FIG. 1. A sample trajectory to comet 67P generated by the MGA optimizer

The solution to the Muti-Gravity Assist (MGA) problem is given by two vectors each of dimension $N+2$ where N is the number of flyby planets. The first vector $\vec{P}$ characterizes the order of the planetary flybys; it must always start with Earth and end with the destination object.

$$\vec{P} = [\text{Earth}, \ldots N \text{ flyby planets} \ldots, \text{ Destination}]$$

The second vector indicates the time at which each planetary encounter occurs.

$$\vec{t} = [t_{earth}, t_{flyby\,1} \ldots, t_{flyby\,N}, t_{destination}]$$

We will soon see by the application of Lambert's equation that these two vectors uniquely characterise any interplanetary trajectory.

With this in mind, we set out to create a general tool to find the best possible trajectory between any two planets, approaching the design of the trajectory through the lens of an optimization problem. In order to do this, certain assumptions and simplifications have to be made to reduce the dimensionality of our solution space, as mentioned above. We then propose a model for characterizing trajectories in a

physically representative yet sufficiently simplistic manner which allows them to evolve through a Genetic Algorithm into a final, optimal trajectory.

III. ASSUMPTIONS AND SIMPLIFICATIONS

The first step in modeling trajectories within the Solar System is logically to make a model of the Solar System itself. In doing this it is important not only to represent the Solar System in a realistic manner, but especially to do so in a way that facilitates the later addition of spacecraft trajectories into the model. For example, given that the gravity assists will occur at specific times and are dependent on the planets' velocities, it is important to be able to find each planet's velocity and position at a given time very efficiently. Thus numerical integration from some initial condition to find the planets' positions is for instance not a good method.

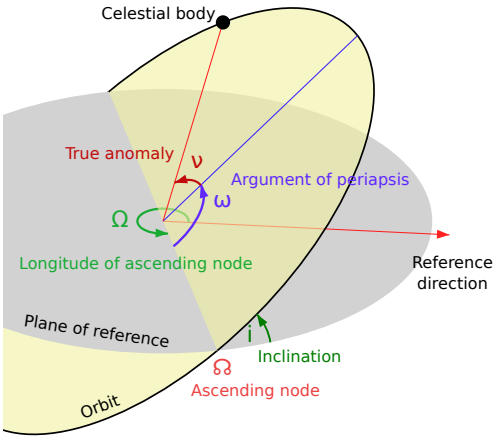

FIG. 2. The Anatomy of a Celestial Orbit (from Wikipedia)

Instead we started off by calculating the exact ephemerides of the planets based on data obtained from JPL's HORIZONS System. The data includes a number of parameters and their time derivatives for each planet, most of which are shown in figure 2. Using the time derivatives, which are approximately constant within a reasonable time frame of about a century before and after J2000, it is possible to update the parameters to their value at a given time past J2000. After some simple calculations with these values, one can find the Eccentric Anomaly of a planet and then use Newton's Method to quickly find the Mean Anomaly. These two 'anomalies' are simply angles that describe a planet's position along its elliptical orbit.

Using the Mean Anomaly, one can find the planet's position in a heliocentric coordinate system with the P axis pointing to the perihelion. We obtain:

$$P = a(\cos E - \epsilon);$$

$$Q = a\sqrt{1-\epsilon^2}\sin E$$

Where a is the semimajor axis of the planet's orbit, E is the mean anomaly, and ϵ is the eccentricity. We then rotate our coordinate system by the orbit's argument of the periapsis, inclination angle, and longitude of the ascending node to obtain the final 3-dimensional planetary orbits in a single (x, y) coordinate system. This is done in 3 steps:

Rotation by the argument of the periapsis (ω)

$$x_1 = P\cos\omega - Q\sin\omega$$

$$y_1 = P\sin\omega + Q\cos\omega$$

Rotation by the inclination angle (i)

$$x_2 = x_1\cos i$$

$$z_2 = x_1\sin i$$

Rotation by longitude of ascending node (Ω)

$$x_{final} = x_2\cos\Omega - y_2\sin\Omega$$

$$y_{final} = x_2\sin\Omega + y_2\cos\Omega$$

$$z_{final} = z_2$$

The orbits found using this method are shown in figure 3.

Although this method is very fast and accurate, there are some issues with integrating it into a

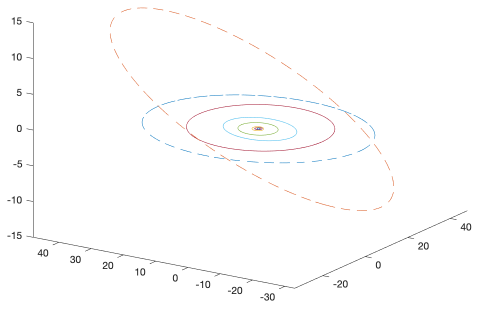

FIG. 3. Actual Orbits using JPL Data

trajectory optimizer. These issues are twofold. First, in terms of animation it is much easier to show the planets moving in circular orbits. In fact, if we look at the eccentricities of the orbits we see that they are all very close to 0, meaning they are all essentially circular. This is illustrated in figure 4. Thus we can make this approximation without losing much accuracy.

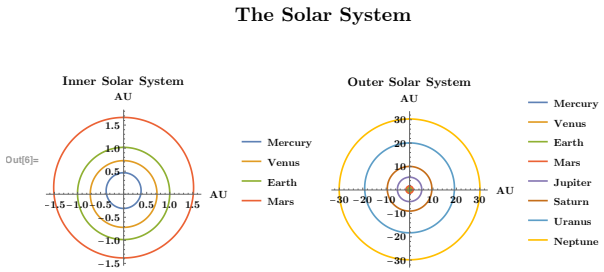

FIG. 4. Low Eccentricity of Planetary Orbits

The second simplification we can make here is related to another assumption which is central to the patched conic approximation described in section V. Namely, within the Sphere of Influence of each planet, the spacecraft feels only that planet's gravitational pull. This simplifies the n-body problem to a 2-body one at each planet without sacrificing accuracy. In order to calculate the gravity assist then, the algorithm must choose the optimal point and direction in which to enter each planet's sphere. During the calculation of the orbits plotted in Figure 2, we rotated each ellipse by an inclination angle to put in in 3 dimensions. However, every angle of inclination is around 2° or less, meaning the planets essential orbit in a single plane. Thus, if we simplify the model to 2D, we immediately massively simplify the problem of choosing a point and direction in 3D on a sphere at each planet to choosing a point in 2D on a circle.

The only planet that deviates from these two approximations is Pluto, which lies 17° outside the plane and has by far the largest eccentricity. Mercury, though not perfect, is close enough to the approximation, and all other planets are nearly perfect. We therefore eliminate Pluto from the project, with the justification that it is, after all, just a dwarf planet.

With a working model of the solar system in place, designed to allow for easy integration of a trajectory optimizer, we can now move on to calculating some trajectories.

IV. LAMBERT'S PROBLEM

Before we can understand how to model interplanetary trajectories we must first revisit one of the most famous problems in celestial mechanics. Posed originally in the 18^{th} century by Johann Heinrich Lambert the formal problem statement is the following.

Given two different times t_1, t_2 and two position vectors $\vec{r}_1$, $\vec{r}_2$, find the solution $\vec{r}(t)$ satisfying the differential equation with the given boundary conditions

$$\ddot{\vec{r}} = -\mu \cdot \frac{\hat{r}}{r^2} \quad \vec{r}(t_1) = \vec{r}_1 \text{ and } \vec{r}(t_2) = \vec{r}_2$$

To solve this problem we use a clever combination of math and physics. We start with Kepler's equation which relates the time of flight to the relative eccentric anomalies E, the gravitational parameter μ, the semi-major axis a and the orbital eccentricity ϵ.

$$\sqrt{\frac{\mu}{a^3}}\Delta t = E(t)\Big|_{t_0}^{t_f} - \epsilon \sin E(t)\Big|_{t_0}^{t_f}$$

The transition from Kepler's equation to Lambert's equation is purely a challenge in geometry whose full derivation is more suited for a text-

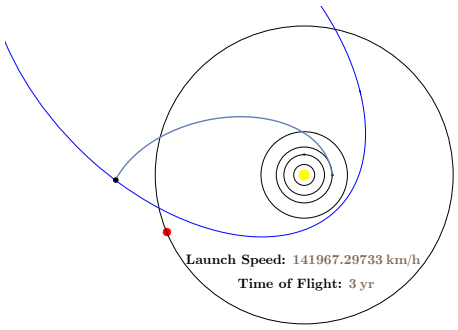

FIG. 5. An example solution to Lambert's Problem (LP) is shown here. Given an exact 3 year time interval and set initial and final positions solving LP gives the exact departure velocity required to reach comet 67P. However, lacking any gravity assists, this path is far from optimal.

book than this report. Only the final form and relevant physics will be presented. Lambert's equation gives an elegant relationship between time of flight and orbital parameters:

$$\Delta t = \sqrt{\frac{a^3}{\mu}}(\alpha - \beta - (\sin\alpha - \sin\beta))$$

Where

$$\sin\left(\frac{\alpha}{2}\right) = \sqrt{\frac{s}{2a}} \text{ and } \sin\left(\frac{\beta}{2}\right) = \sqrt{\frac{s-c}{2a}}$$

The constants c and s are called the chord and semi-perimeter of the transfer.

$$c = \|\vec{r}_1 - \vec{r}_2\| \;\; s = \frac{c + r_1 + r_2}{2}$$

Lambert's equation shows that the transfer time of a body moving between two points on a conic trajectory is a function only of the sum of the distances of the two points from the origin of the force, the linear distance between the points, and the semi-major axis a of the conic. Once we have a it is a straight forward energy calculation to compute the initial velocity required for the transfer. We will forgo writing out the final formula to avoid defining four more geometric constant first.

Recapping, using only geometry and physics a highly non-linear coupled system of differential equations has turned into a root finding problem. A simple bisection algorithm can be written to find which value of a yields the specified value of Δt. Bisection is easy to implement but unfortunately very slow, so for this project we have used a new, unpublished, and extremely fast algorithm created by the European Space Agency's (ESA) Advanced Concepts Team which utilizes a transformation that deforms the time of flight curves into lines, thus allowing uniform, consistent convergence properties for all possible geometries. With the speed and efficiency of this algorithm, a standard laptop running 8 parallel threads can solve Lambert's Problem millions of times in a few seconds. We will later see why this ability is crucial in order to solve the MGA problem.

V. MODELING TRAJECTORIES

So far we have a working Solar System and a method to calculate the spacecraft's path between 2 planets. However, what we are looking to model is a trajectory encompassing not only multiple paths between planets, but also interactions with the planets.

As mentioned briefly in section II, trajectories can be described uniquely by two vectors, each of dimension $N + 2$ where N is the number of flyby planets. The first vector $\vec{P}$ characterizes the order of the planetary flybys; it must always start with Earth and end with the destination object.

$$\vec{P} = [\text{Earth}, \ldots N \text{ flyby planets} \ldots, \text{ Destination}]$$

The second vector indicates the time at which each planetary encounter occurs.

$$\vec{t} = [t_{earth}, t_{flyby\,1} \ldots, t_{flyby\,N}, t_{destination}]$$

These vectors uniquely describe an interplanetary trajectory because they also form the inputs to our Lambert Solver, and thus provide not only which planets we meet and when, but also the the path the spacecraft follows between the planets.

Since we assume no thrust during the interplanetary travel time - an assumption that is also important in solving Lambert's Problem - we model any abrupt changes in velocity as oc-

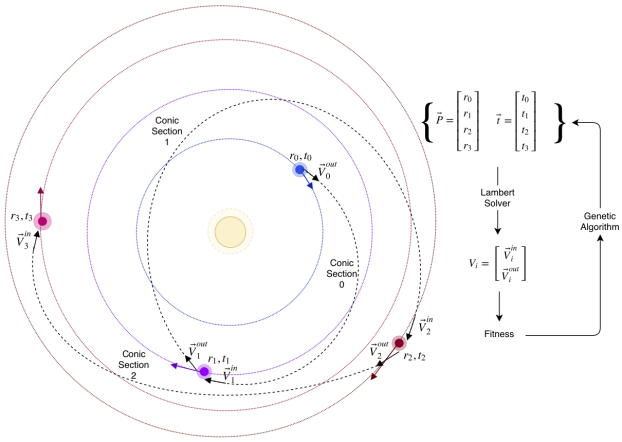

FIG. 6. Splitting up the trajectories in this way, a complex n-body problem turns into N gravity assist problems and $N+1$ conic sections calculated using the Lambert Solver.

curring solely due to either the gravity assists or the impulses delivered by the spacecraft during the gravity assists. These are dictated by the required incoming and outgoing velocities of the spacecraft at each planet, which are in turn provided by the solutions to the Lambert Problem. Thus, in calculating the efficiency of a trajectory, we have to look at the outputs of the Lambert Solver to ascertain how much fuel must be burned during the planetary interactions.

Given the efficiency of a trajectory, the algorithm can try to improve it by either visiting different planets or changing the times of the maneuvers. Altering either one of these parameters causes the Lambert Solver to output different velocities, which changes what the spacecraft has to do during the assist maneuvers. Ultimately, the algorithm tries to find the best possible $\vec{P}$ and $\vec{t}$ to minimize the amount of fuel the spacecraft has to burn and instead maximize the work that the gravity assists do for the spacecraft. The whole process is illustrated in figure 6.

VI. THE FITNESS FUNCTION

In order to perform this optimization, each trajectory must be given an efficiency score based on how much fuel it burns. Since we are using a Genetic Algorithm for the optimization, this score is found using a so-called Fitness Function.

One of the most important aspects of this project is finding a fitness function that represents the reality of a gravity assist and gives high scores for assists that most benefit the efficiency of the trajectory. At the same time, it should not actually calculate the full physics of a gravity assist. This would require optimizing not only the two vectors mentioned in the previous section, but also a vector of positions of entry into each planet's sphere of influence and of impact parameters for each maneuver. Since both of the latter vectors sample from infinite spaces of possible values, this would increase the complexity of the algorithm exponentially and is unfortunately simply outside the scope of this project given the time and resource constraints.

As we cannot exactly model the physics, we adopted an alternate strategy. We first studied the gravity assist problem - what makes a given maneuver fuel efficient, and how we can gain the desired velocity and direction from an assist. Based on this, we created a fitness function that takes the spacecraft's incoming and outgoing velocities as well as the planet's velocity and calculates normalized values of fitness based on 4 important criteria. The total fitness is the weighted sum of the 4 criteria scores, with the weights experimentally derived by running the algorithm and comparing it with known optimal trajectories given by the ESA, trajectories calculated by hand, and trajectories selected by real spacecraft in the past. Finally, we ran the algorithm on new combinations of planets and validated the realism of the assist maneuvers using hand calculations. In the end, though not entirely physical, the fitness function provides excellent and realistic trajectories.

Let us now look in detail at what this fitness function actually entails. Figure 7 will be helpful for this part. The first of the 4 criteria is a score based on the angles of incidence and 'diffraction' of the spacecraft relative to the line spanned by the planet's velocity vector. These are shown as ϕ_{in} and ϕ_{out} in the diagram. In general, during a celestial interaction with no applied thrust and sufficient energy that the orbit is unbounded (i.e. usually hyperbolic), ϕ_{in} will approximately equal ϕ_{out}. Therefore, any change in this angle requires fuel and results in a negative angular

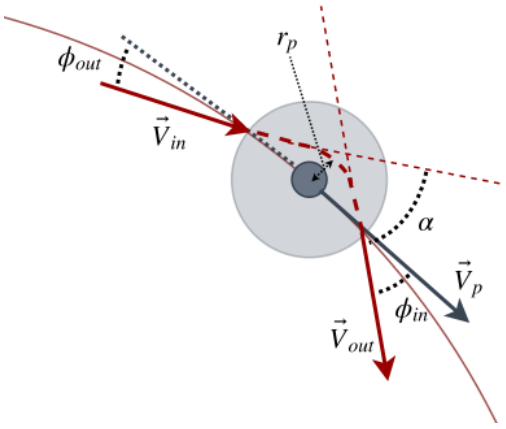

FIG. 7. The anatomy of a gravitational assist. r_p is the impact parameter, $\vec{v}_{in}$ and $\vec{v}_{out}$ the incoming and outgoing spacecraft velocities, and $\vec{v}_p$ is the planet velocity

fitness score given by:

$$F_\phi = \frac{\vec{v}_{in} \cdot \vec{v}_{out}}{|\vec{v}_{in}||\vec{v}_{out}|} - 1$$

The dot product is normalized by the magnitudes of the two vectors so that this term only scores the angle and is unaffected by the amplitudes. Note also that for this term as well as the others, corner cases are considered and constraints placed upon the values in Python code to ensure best possible performance.

The second criterion is also related to the incoming and outgoing angles, but not to their relative change. Gravity assists most effectively impart energy upon the spacecraft if the final and initial spacecraft velocities are not opposed to the planet's velocity. In other words, the dot products of the spacecraft and planet velocities should positive and as large as possible. This is expressed mathematically as:

$$F_v = \frac{1}{2}\left(\frac{\vec{v}_{in} \cdot \vec{v}_p}{|\vec{v}_{in}||\vec{v}_p|} + \frac{\vec{v}_{out} \cdot \vec{v}_p}{|\vec{v}_{out}||\vec{v}_p|}\right)$$

Again, this term is normalized to a value between 0 and 1 so it can easily be weighted and added to the other terms.

The third criterion is perhaps the most obvious and directly measures how much thrust the spacecraft has to give. As explained before, the magnitude of the spacecraft's velocity relative to the planet should not change during the maneuver. Hence, any relative change in velocity is due to a thrust impulse and is negative in our fitness function:

$$F_T = \left|1 - \frac{|\vec{v}_{out} - \vec{v}_p|}{|\vec{v}_{in} - \vec{v}_p|}\right|$$

The final criterion, F_t, is actually related to time rather than fuel consumption. It is simply the total time it takes from the start planet to the destination. The idea here is that a longer mission will require more supplies and therefore more weight to carry. A beneficial side effect of this term is that it tends to prioritize slightly simpler, less convoluted trajectories, which are perhaps more realistic and achievable.

For each gravitational assist manoeuvre, we add the 4 terms in a weighted sum:

$$F_i = c_\phi F_\phi + c_v F_v + c_T F_T + c_t F_t$$

The fitness of each interaction is then averaged to find the final fitness of the trajectory:

$$F = \frac{1}{N}\sum_{i=1}^{N} F_i$$

VII. GENETIC ALGORITHM

We now have a model of the Solar System and any arbitrary spacecraft trajectory in it, as well as a fitness function that tells us the effectiveness of the trajectory. What remains to be done is to somehow use this fitness function to find the optimal trajectory from one planet to another. With the physics done, this becomes an optimization problem. Specifically, it is a relatively unconstrained problem with a vast solution space, so an effective method is the Genetic Algorithm.

The Genetic Algorithm is a very natural and intuitive algorithm as it closely follows Darwin's evolutionary theories. We start with a population of randomly generated individuals - in our case, 160 random trajectories, which are split into 8 groups of 20. These groups are called islands and represent isolated populations that

will evolve separately in parallel. The benefits of such a split will be explained later. Each individual is characterised by its chromosomes, which in this case are the two vectors used to represent a trajectory. The different values found in each individual's vectors are called that individual's alleles and determine how fit the individual will be.

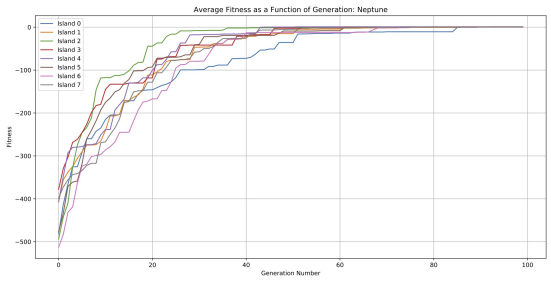

FIG. 8. Average island fitness over 100 generations. In all of our trials, the solutions converged to a maximum after at most 100 generations, allowing for fast runtimes.

The algorithm simulates evolution by repeating the following steps until it converges to an optimum:

1. Calculate the fitness of each individual in the population using the fitness function
2. Eliminate the least fit individuals
3. Evolve the most fit individuals using mutation and crossover. In mutation, random changes are made to the individuals' alleles, while in crossover, also known as recombination, two fit individuals' alleles are combined into a new trajectory. This is completely analogous to how biological populations evolve. At each step the mutations and crossovers occur with a given probability depending on the algorithm used.
4. The new generation of improved individuals restarts the cycle. The fitness of a series of generations is plotted in figure 8.

The specific implementation of the Genetic Algorithm we used is the PyGMO library from the ESA's Advanced Concepts Team. This has a few interesting characteristics.

First, this algorithm evolves each individual island in parallel, meaning it is very fast. Combined with our exceptionally fast Lambert Solver, this allows us to optimize entire new trajectories in around 10 seconds. Another benefit of the concurrency of the algorithm is that it is very effective for finding global maxima. Even if one or more islands become stuck at a local maximum, the stochastic nature of the mutations in addition to the isolation of each island invariably allow the algorithm to find something close to the global maximum of the solution space. This is illustrated well in figure 9.

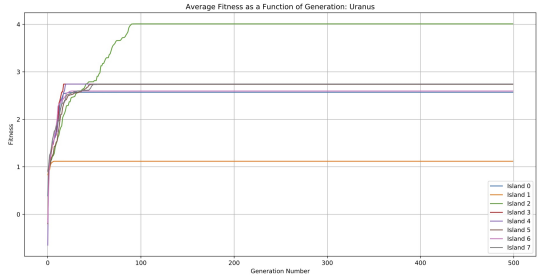

FIG. 9. Average fitness of each island evolving over 500 generations. This displays the ability of the algorithm to find near global maxima even when some populations are held up at a local maximum. Here, the island in green achieves a high fitness score while the others converge to a lower value.

Another feature of the PyGMO library is that it performs some inter-island 'migration' of individuals, which adds more randomness and increases the probability of achieving a global maximum. The migration is carried out through the Barabási-Albert model.

Finally, the overall algorithm used to run the Genetic Algorithm is called Self-Adaptive Differential Evolution. This essentially dictates how likely mutation and cross-over are to occur and how they are actually implemented. It decides these parameters based on the system it is trying to optimize, hence the name self-adaptive. This allows the Genetic Algorithm to be run effectively without spending huge amounts of time trying to optimize parameters like mutation frequency for our specific application.

VIII. SOFTWARE

As a part of this project, a fully 3D interactive simulation was developed in order to best present our work. The application, freely available on GitHub here, allows for the real-time generation of new interplanetary trajectories that can be viewed and controlled dynamically via mouse and keyboard. Many sample trajectories are also provided to spare the user from building the full backend (instructions provided in README). The application, built in processing, launches with the following GUI, allowing the user to select their destination, or chose a pre-generated sample.

FIG. 10. Application GUI

After selecting an option, pressing the enter key advances the program into the simulator, which may take a few seconds to a few minutes depending on the chosen target as well as system threads and processing speed. Once the simulator starts the user will be able to view the trajectory progress and is given full speed control using the left and right arrow keys. Mouse control allows the user the smoothly pan around the camera to view the trajectory from different angles. At any point, the enter key can be pressed to restart the current trajectory or the application can be quit and reopened to go back to the GUI.

The inner workings of the software are slightly complicated and require many dependencies. Upon selection of a non-demo trajectory, a new Python background process will be started and is passed the name of the planet it must reach. It will then chose random orbital

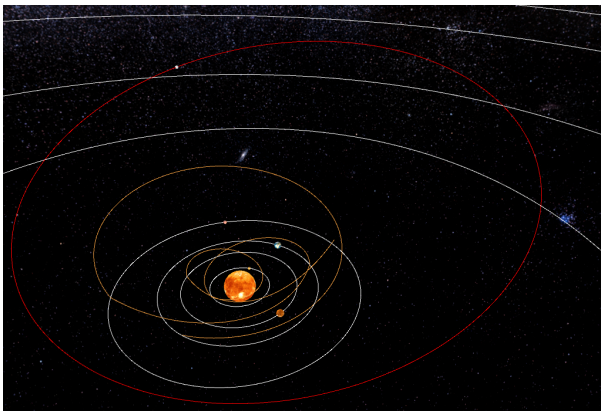

FIG. 11. Screenshot of simulator

phases for all of the planets in order to increase the variety of generated trajectories. Then the optimization will begin. The evolutionary algorithm will evolve 8 populations of 20 in parallel for 100 generations using methods described in this paper. Upon selection of the optimal solution, a text file will be generated containing the initial velocities of each leg, and the encounter times. Next, a new Mathematica process will be called which numerically solves the Kepler initial value problem on each leg using an Explicit Runge Kutta Method with varying time step to ensure convergence and accuracy. The numerical solution is further enhanced using Hermite Interpolation between each time step. Next, the trajectory location every $\frac{1}{100}$ years is written to a CSV file. This file is read inside processing and drawn using cubic splines. In order to allow an arbitrary program time step, further linear interpolation is done between data points when the animation time falls out of sync with the trajectory data.

IX. ANALYSIS

With this fully functional trajectory optimizer in hand, the question of whether it is actually accurate remains. To get a better sense of the efficiency of the gravitational encounters in a generated trajectory, we can study a plot (12) of solar distance and velocity throughout the trajectory to comet 67P. An interesting conclusion from figure 12 is that the optimizer has chosen to engage in gravity assists near perigee, a surely effective heuristic as perigee is the location of maximum

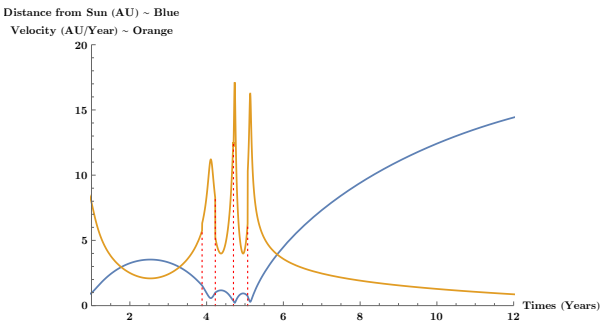

FIG. 12. Heliocentric distance and velocity over the full trajectory to comet 67P. Dotted lines indicate encounter times

velocity. This heuristic was not explicit within the code. Next we observe figure 13, a plot of the specific orbital energy over the trajectory, so we can observe the energetic effects of each encounter. We notice that the first encounter leads

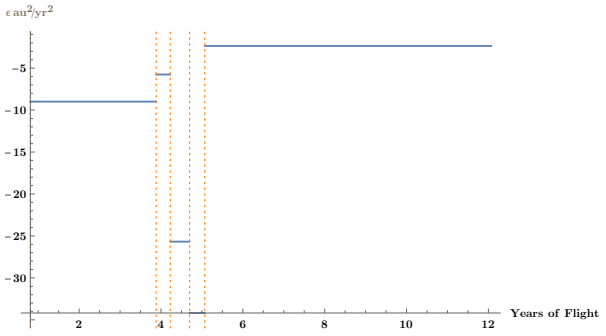

FIG. 13. Specific orbital energy as a function of time. Dotted lines show energy changes during encounters.

to an increase in energy, while the next two result in a drop. A drop in energy, while perhaps globally optimal, should be locally un-optimal. Hence, an effective validation of the fitness function will be to observe if the calculated scores for the second and third assist are less optimal than the first.

As a result of the oversimplified fitness function, the optimizer was unable to compute exact thrust factors for each assist. Thus, to validate the simplified approach, the trajectory to 67P was studied in further depth. Each assist was solved numerically for different impact parameters, and optimized for the smallest difference in planet frame velocities. This method provides an accurate measure of of the thrust correction factor at each leg by computing the square of the difference of the required velocity for the next Lambert arc and the velocity naturally produced by the encounter.

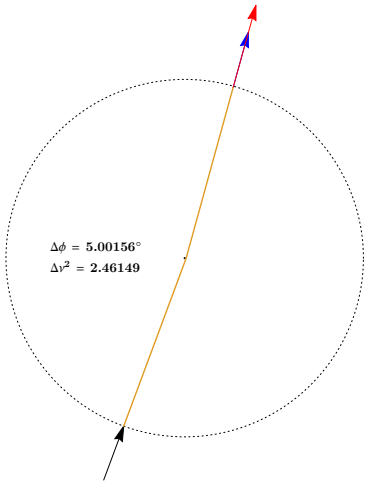

FIG. 14. Mars assist on trajectory to 67P

Figures 14-17 show the first three gravity assists on the journey to 67P. Black arrows represent incoming velocities into the SOI, blue arrows show the natural output velocity leaving the SOI, and red arrows show targeted velocities required for the next Lambert arc. All velocities are taken in the planets frame of rest. Additionally a scale black dot, and dotted circle are drawn to show the planet and SOI respectively.

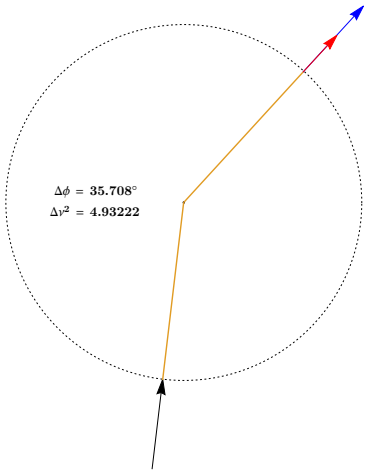

FIG. 15. Attempted Earth assist

In figure 14 we see a fairly successful assist that is able to position the craft in line with the Lambert velocity, differing only by a reasonable thrust factor.
In figure 15 we see the result of the optimal impact parameter chosen for the second assist on Earth. However, looking closer, it is evident that to achieve this transfer, the craft must have passed through the planet. This issue was resolved by adding further constraints, and the solution is shown in figure 16.

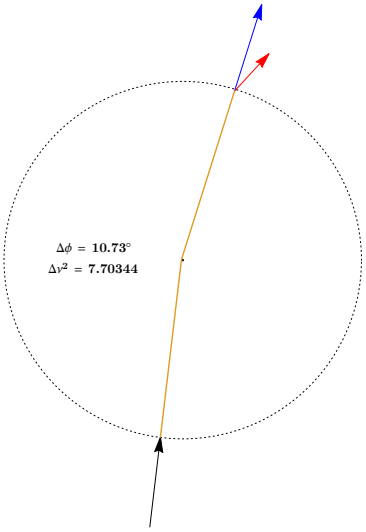

FIG. 16. Earth assist on trajectory to 67P

The final maneuver on Mercury shown in figure 17 seems fairly non-optimal since it requires a significant burn to correct the velocity.

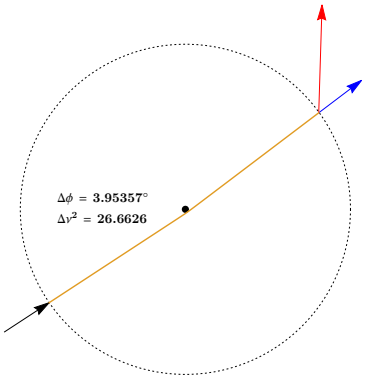

FIG. 17. Mercury assist on trajectory to 67P

To complete our analysis we must compare the thrust factors for each encounter with the optimization fitness. The result is shown in figure 18. The fitness function was designed to output higher scores for more optimal trajectories, however this is incompatible with the thrust factors as a lower value represents a more efficient maneuver. To resolve this issue, the fitness score is shifted down and negated so it can be plotted alongside the thrust factors for more accurate comparison.

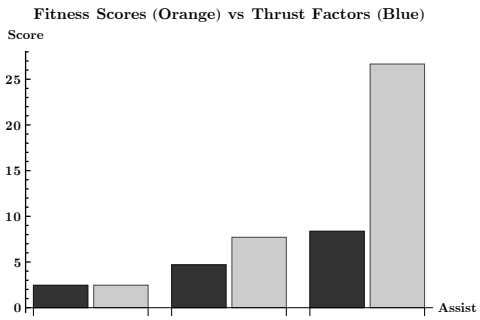

FIG. 18. Comparing fitness scores with analytic thrust factors

We see a definite correlation between thrust factors and fitness scores. Although a clear issue is shown by the non uniform scaling of the fitness with the thrust factors, the results are more then acceptable given the scope of this project.

X. CONCLUSIONS

Since Galileo first looked at the night sky through his telescope, exploring the cosmos has been one of humanity's greatest challenges and obsessions. For centuries curiosity and the desire for knowledge has driven us through ever-improving technology to reach further into space. Now we have reached the point where companies are taking commercial interest in our Solar System, and the private sector is racing to make space tourism a reality.
Whatever the motivation behind the exploration, the fact remains that space travel still faces some fundamental and challenging obstacles. One of the biggest challenges is the exorbitant cost of carrying payloads into space due to fuel and mass limitations. This is especially

pertinent as exploration moves more and more towards carrying people into space and potentially to other planets.

In this project we have proposed an optimization and physics - based algorithm to take advantage of the enormous benefits presented by gravity assists in making space travel more efficient. The algorithm quickly and accurately calculates and visualizes the optimal trajectory for a spacecraft leaving from Earth and heading to an arbitrary planet in our Solar System. The trajectories are designed iteratively using a Self-Adaptive Differential Evolutionary Genetic Algorithm with a fitness function rooted in the physics of gravity assists that minimizes fuel usage. The parallel-running optimization algorithm is paired with a novel, highly efficient Lambert solver to provide fast and reliable results.

Given more time and perhaps faster computers, this project could be expanded to operate within the almost perfectly accurate solar system model we introduced initially. Subsequently, the genomes of the individuals being evolved in the Genetic Algorithm could be made more complex by adding chromosomes describing the actual gravity assist parameters at each planet, as described in section VI. The fitness function could then be updated to calculate exactly the physics of each gravity assist in 3 dimensions and provide a realistic value for fuel consumption. With these few but very difficult and computationally intense augmentations, our algorithm would then present a nearly completely accurate optimization of spacecraft trajectories, ready to be put into action to plan actual space voyages.

However, while the resulting optimal trajectories from our simplified algorithm rely upon some assumptions and simplifications such as a planar Solar System, circular orbits, a patched conic approximation, and a fitness function that does not strictly solve the gravity assist problem, we have shown through hand calculations and validating examples from sources like JPL and the ESA that our generated trajectories are accurate and indeed present highly efficient paths to the destination planets. In addition, through our intuitive and attractive GUI we can demonstrate some of the fundamentals of gravity assisted space travel and why this sort of optimization is key to the future of space exploration.

XI. REFERENCES

- JPL Ephemeris Data: https://ssd.jpl.nasa.gov/?planet_pos
- PyGMO: http://esa.github.io/pygmo/
- Qin, A., & Suganthan, P. (2005). Self-adaptive Differential Evolution Algorithm for Numerical Optimization. IEEE Congress on Evolutionary Computation.
- Carnelli, I., Dachwald, B., & Vasile, M. (2009). Evolutionary Neurocontrol: A Novel Method for Low-Thrust Gravity-Assist Trajectory Optimization. Journal of Guidance, Control, and Dynamics, 32(2), 616-625.
- Vasile, M., Martin, J. M., Masi, L., Minisci, E., Epenoy, R., Martinot, V., & Baig, J. F. (2015). Incremental planning of multi-gravity assist trajectories. Acta Astronautica, 115, 407-421.
- Zuo, M., Dai, G., Peng, L., Chen, L., Chen, X., & Song, Z. (2016). Global optimisation of multiple gravity assist spacecraft trajectories based on search space exploring and PCA. IEEE Congress on Evolutionary Computation (CEC).