SPACECRAFT TRAJECTORY OPTIMIZATION SUITE: FLY-BYS WITH IMPULSIVE THRUST ENGINES (STOPS-FLITE)

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Aaron Hogan Li

June 2022

COMMITTEE MEMBERSHIP

TITLE:	Spacecraft Trajectory Optimization Suite: Fly-bys with Impulsive Thrust Engines (STOpS-FLITE)
AUTHOR:	Aaron Hogan Li
DATE SUBMITTED:	June 2022
COMMITTEE CHAIR:	Kira Abercromby, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Eric Mehiel, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Paul Choboter, Ph.D. Professor of Mathematics
COMMITTEE MEMBER:	Timothy Fitzgerald, M.S. Aerospace Engineer

ABSTRACT

Spacecraft Trajectory Optimization Suite: Fly-bys with Impulsive Thrust Engines (STOpS-FLITE)

Aaron Hogan Li

Spacecraft trajectory optimization is a near-infinite problem space with a wide variety of models and optimizers. As trajectory complexity increases, so too must the capabilities of modern optimizers. Common objective cost functions for these optimizers include the propellant utilized by the spacecraft and the time the spacecraft spends in flight. One effective method of minimizing these costs is the utilization of one or multiple gravity assists. Due to the phenomenon known as the Oberth effect, fuel burned at a high velocity results in a larger change in orbital energy than fuel burned at a low velocity. Since a spacecraft is flying fastest at the periapsis of its orbit, application of impulsive thrust at this closest approach is demonstrably capable of generating a greater change in orbital energy than at any other location in a trajectory. Harnessing this extra energy in order to lower relevant cost functions requires the modeling of these "powered flybys" or "powered gravity assists" (PGAs) within an interplanetary trajectory optimizer. This paper will discuss the use and modification of the Spacecraft Trajectory Optimization Suite, an optimizer built on evolutionary algorithms and the island model paradigm from the Parallel Global Multi-Objective Optimizer (PaGMO). This variant of STOpS enhances the STOpS library of tools with the capability of modeling and optimizing single and multiple powered gravity assist trajectories. Due to its functionality as a tool to optimize powered flybys, this variant of STOpS is named the Spacecraft Trajectory Optimization Suite - Flybys with Impulsive Thrust Engines (STOpS-FLITE).

In three test scenarios, the PGA algorithm was able to converge to comparable or superior solutions to the unpowered gravity assist (uPGA) modeling used in previous

STOpS versions, while providing extra options of trades between time of flight and propellant burned. Further, the PGA algorithm was able to find trajectories utilizing a PGA where uPGA trajectories were impossible due to limitations on time of flight and flyby altitude. Finally, STOpS-FLITE was able to converge to a uPGA trajectory when it was the most optimal solution, suggesting the algorithm does include and properly considers the uPGA case within its search space.

ACKNOWLEDGMENTS

I could write a hundred pages about the people who've helped me to this point, but I only have this one. Special thanks to:

- Dr. Kira Abercromby, for being my advisor, in undergraduate and graduate school, but also in life. I could never have made it through this major these last 5 years without her support and advice.
- Sam Westrick and Natalia Cieply, for being my greatest supporters in my aerospace endeavors. They will be graduating alongside me with Master's degrees of their own and will always be a lovely reminder of how lucky I am be a Cal Poly student, because they were, too.
- Lauren Dennen, for being there for me at my absolute best and my absolute worst. Her love and honesty brightens and guides my life just as certainly as the Sun and stars.
- Rachel Neil, for believing in me in all things, especially when I don't believe in myself. Her kindness and patience is as boundless as the Universe itself and will surely support me for a long time to come, in success and in failure.
- My mom and dad, for teaching me the value of education, for passing down a lifelong love of learning, and for sacrificing their dreams and homeland so that I could have a chance at achieving my dreams and finding my home.

For all of us who call ourselves the children of planet Earth, may our empathy one day match our curiosity, our tenacity, our hope, and the vastness of the Universe, that our children may truly be deserving of living amongst the stars our ancestors worshipped as the realm of the gods.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Problem Statement

Powered gravity assists represent a small part of the interplanetary trajectory optimization toolbox. Advances in modeling and simulation as well as technological developments like electric propulsion have vastly improved mission design capabilities. While electric propulsion and similar advancements in spacecraft engine systems have resulted in significant gains in the ability of spacecraft to produce changes in velocity (δv) themselves, modeling and simulation improvements have allowed the design of missions that take advantage of δv gains from gravity assists. Most missions utilize a ballistic gravity assist, one in which no thrust is applied by the spacecraft engine during the gravity assist. Ballistic gravity assists are the simplest modeling of gravity assists, and many different expansions of this case exist such as aerogravity assists, tethered gravity assists, and powered gravity assists, the topic of this paper. Powered gravity assists (PGAs) take advantage of the Oberth effect to increase the δv effect beyond that of the ballistic gravity assist. The Oberth effect is the physical phenomenon in which fuel utilized inside a gravitational well is more effective than the same fuel utilized outside the gravitational well [1]. This paper will limit its scope to covering specifically impulsive thrust powered gravity assists. Though some literature discusses continuous/low thrust powered gravity assist analysis, the Oberth effect is more noticeable with impulsive/high thrust systems. Thus, this paper will analyze the effectiveness of gravity assists amplified by impulsive, high thrust engines. While studies of PGAs typically consider their effect on maximum energy transfer,

few discussions of their applicability to interplanetary trajectory optimizations have been published, leading to this study on their functionality as a tool to decrease relevant mission parameters like time of flight or δv. The Spacecraft Trajectory Optimization Suite (STOpS), first developed at California Polytechnic State University, San Luis Obispo in 2015 with multiple expansions in the years since, serves as the backbone of this study. STOpS utilizes metaheuristic evolutionary algorithms and shares solutions between them to find the global optimum of an interplanetary trajectory with regards to multiple parameters. By modifying the gravity assist modeling of STOpS, a study of PGA applicability to interplanetary trajectory optimization can be conducted. Because of its PGA capabilities, this variant of STOpS is named the Spacecraft Trajectory Optimization Suite - Flybys with Impulsive Thrust Engines (STOpS-FLITE). STOpS-FLITE optimizes trajectories using the island model paradigm and the algorithms from the original version, with the added capability of performing PGAs within the sphere of influence of planetary bodies.

1.2 Orbital Mechanics & Interplanetary Trajectories

The design of spacecraft trajectories is an important factor in the design of spacecraft themselves [2, 3]. Better optimization of an orbital path reduces required fuel (δv) or decreases time of flight (TOF), among other improvements to a spacecraft's mission [4]. These trajectories are limited by the laws of physics, specifically those of orbital mechanics.

Within the world of orbital mechanics, all bodies travel in paths defined by conic sections with one of the foci at the celestial body being orbited, be they circles, ellipses, parabolas, or hyperbolas. For example, the planets of the solar system are in orbits that are nearly circular around the Sun. These are often simplified to being

truly circular for the sake of mathematical simplicity. For a spacecraft to travel between planets, it travels on an elliptical path, with one of the foci of that ellipse being centered on the Sun. A common way to model this maneuver is by solving Lambert's problem.

1.2.1 Lambert's Problem

Solutions to Lambert's problem are a popular and well understood way of numerically calculating required δv's for orbital maneuvers [2]. While many formulations of solutions to Lambert's problem exist, all at least require the following inputs: the position vector of the spacecraft before departure, the position vector of the spacecraft at arrival, a time of flight, and the gravitational parameter (μ) of the body around which the spacecraft is orbiting. With these inputs, a Lambert's problem solver can obtain the velocity vector at departure to put the spacecraft on a trajectory to reach the arrival position given that specific time of flight. Lambert's solvers also provide the velocity vector of the spacecraft on arrival to the second position. By comparing these two velocity vectors to the velocity vectors of the spacecraft in their original orbits, the δv of the maneuver can be calculated. The velocity of the spacecraft in their original orbits are typically considered to be identical to that of the planet from which they depart or at which they arrive. An example solution of Lambert's problem is presented below in Figure 1.1.

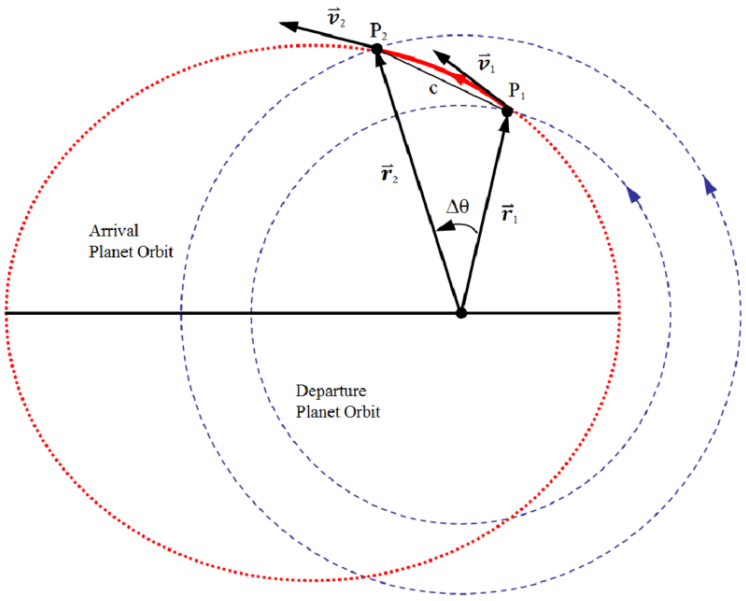

Figure 1.1: Example Solution to Lambert's Problem [5]

Robust versions of the Lambert's solver can account for a spacecraft going the "long way" (i.e. the direction in which the traversed angle is greater than 180 degrees) or performing multiple revolutions to arrive at its end position. One of these solvers is the Izzo-Gooding formulation, which is the one primarily utilized by STOpS. Other formulations are also included, but need to be called specifically by the user to be utilized. Izzo-Gooding is a compromise between computational speed and numerical accuracy, and its robustness led to its widespread usage.

1.2.2 Gravity Assists

Many missions have used gravity assist maneuvers around a planet or other celestial body by utilizing the gravity of said celestial body to achieve some change in velocity [2, 6]. A gravity assist is a maneuver that uses the gravitational pull of a celestial body to change the velocity vector of a spacecraft on an interplanetary trajectory. With a ballistic gravity assist, only the direction of this velocity vector

changes. The other expansions of the ballistic gravity assist mentioned previously may also change the magnitude of the velocity vector. In literature, gravity assists may also be found referred to as a "swing-by" or "fly-by". Within this paper, these terms will be considered interchangeable.

Gravity assists are most often considered to be ballistic, where any thrust maneuvers of the interplanetary trajectory is considered to be performed outside the sphere of influence (SOI). The SOI is a region around a celestial body in which the gravitational force of the planet overcomes that of the Sun and becomes the dominant force on an object, such as a spacecraft. Once a spacecraft arrives at the SOI of a planet, the gravity of the planet begins to be the dominant force on the spacecraft, overpowering that of the Sun. Within the SOI, the only force acting upon the spacecraft is considered to be the gravitational pull of the planet (unless orbital perturbations like N-body effects, drag, and solar radiation pressure are also considered). The gravitational pull of the Sun is considered to be acting upon both planet and spacecraft equally, and thus, can be excluded mathematically when considering the ballistic gravity assist. Before entering the SOI of a celestial body, the spacecraft is on a ballistic path described by a conic section relative to the heliocentric frame [2]. The difference between the spacecraft's velocity from the solution to Lambert's problem and the planet's velocity is the spacecraft's v_∞. If this velocity is smaller than the escape velocity (the velocity required for an object to escape the gravitational pull of the flyby body), the spacecraft will be captured into an elliptical orbit about the planet and will not leave the SOI. However, capture into an orbit around a planet will typically require a thrust maneuver to lower the velocity of the spacecraft, since most spacecraft arrive at the SOI of a planet with a v_∞ greater than the v_{esc} (the velocity required to escape the gravitational pull of the planet). If the v_∞ of a spacecraft is greater than the v_{esc}, then the spacecraft will be in a parabolic or hyperbolic path around the planet. Once the spacecraft passes through the edge of the SOI, a

switch can be made made, from the heliocentric frame to a planetary frame. This is a common change of frame to make as the planet is now the attracting body at the center of the system. Within this planet-centered frame, a spacecraft can either fly in front of a planet (relative to its direction of travel) or fly behind a planet. These are referred to as leading and trailing side flybys, respectively, and are shown in Fig. 1.2. Gravity assists are modeled with a parabolic or hyperbolic arc within the celestial body reference frame. These arcs are then linked to elliptical arcs in the heliocentric frame to model the full interplanetary trajectory, in what is known as the "patched conic" approximation. These elliptical and hyperbolic arcs link up at the edge of the SOI. The patched conic approximation sacrifices some accuracy in exchange for much greater computational speed.

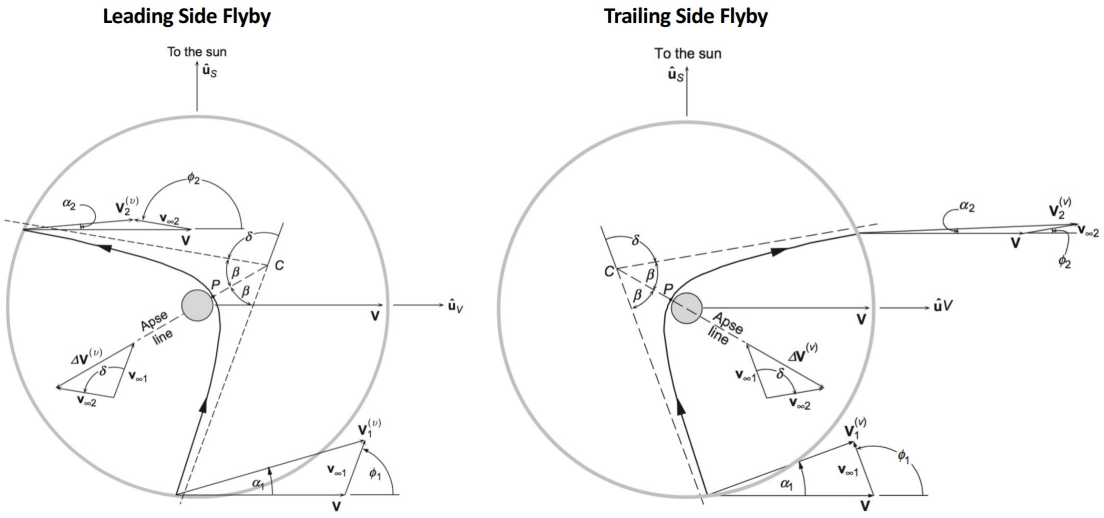

Figure 1.2: Leading Side vs. Trailing Side Gravity Assist [2]

As seen from Figure 1.2, the benefit of a ballistic flyby is in changing the direction of the spacecraft relative to the heliocentric frame, resulting in a "free" change in velocity due to a change in direction, rather than in magnitude [2]. It is important to note that conservation of angular momentum of the spacecraft-planet system is still satisfied since the spacecraft also pulls on the planet, though the much larger

mass of the planet means its change in velocity is negligible [7]. The exact change in velocity of the spacecraft (i.e. δv) can be calculated directly from the hyperbolic excess velocity when the spacecraft crosses into the SOI of the planet, the planetary heliocentric velocity (V_{planet}) and gravitational parameter (μ), and the angle (δ) and radius (r_p) of closest approach in a planetary frame [2]. Depending on which side of the planet the spacecraft passes, the spacecraft will either lose heliocentric energy (leading side flyby) or gain heliocentric energy (trailing side flyby).

With this basic breakdown of classical orbital mechanics, the next chapter will focus more specifically on the Oberth effect and its usefulness in performing powered gravity assists. For further information on basic orbital mechanics, the author refers the interested reader to Orbital Mechanics for Engineering Students by Howard D. Curtis, Fundamentals of Astrodynamics and Applications by David A. Vallado, and Orbital Mechanics by Vladimir A. Chobotov [2, 3, 8].

Chapter 2

POWERED GRAVITY ASSISTS

2.1 Oberth Effect

The Oberth effect was first described and applied in theory to spacecraft in 1927 by its namesake, Hermann Oberth. Oberth postulated that at high speeds, a spacecraft's fuel can be utilized more effectively since it will have both the chemical energy stored within it and the kinetic energy of the system as a whole. The chemical energy of the fuel remains the same and therefore the force it can apply is the same. At high speeds, the spacecraft gains an increased amount of kinetic energy for the same amount of fuel. Energy is still conserved for the whole system, despite the apparent "free" gain in kinetic energy for the spacecraft itself. This is because the fuel from a spacecraft traveling at high speeds has less energy after being burned and exhausted compared to the same fuel exhausted from the same spacecraft traveling at a lower speed.

While the Oberth effect can be studied with low-thrust systems, it is typically taken advantage of with high-thrust, impulsive maneuvers at closest approach [9]. Due to the relationship between velocity and position in orbital mechanics, the spacecraft spends very little time close to the planet with a high velocity, thus systems that can apply a large amount of energy in a short period of time (i.e. impulsive thrust) are more effective at taking advantage of the Oberth effect. Oberth and Edelbaum proved that exiting a circular orbit was most optimal with regards to fuel usage (and even TOF, at times) with a two- or three-impulse maneuver that brings the spacecraft closer to the body to take advantage of the Oberth effect [10]. This is shown in

Figure 2.1 with: a single-impulse direct escape maneuver (left), a two-impulse Oberth maneuver (center), and a three-impulse Edelbaum maneuver (right) [10].

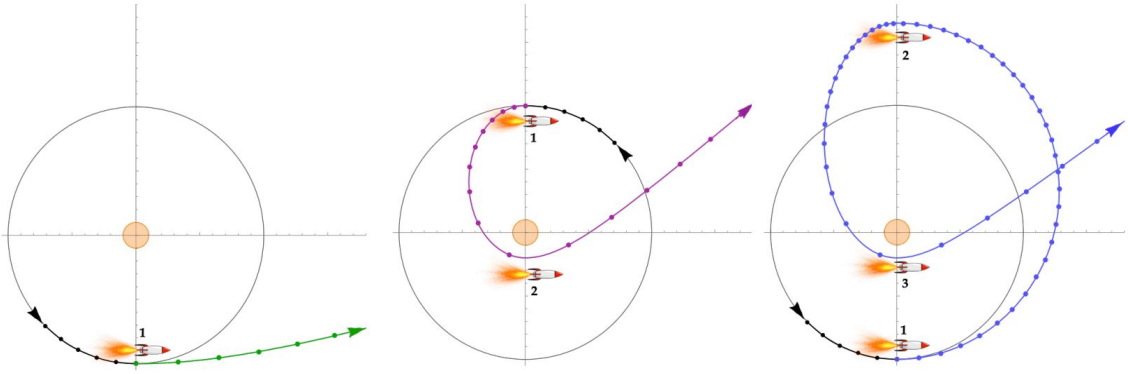

Figure 2.1: Circular Orbit Impulsive Escape Maneuvers [10]

Compared to the circular escape trajectory, hyperbolic and parabolic orbits have greater velocities with the same periapsis, suggesting an even more potent potential to use the Oberth effect to change orbital energy. Thus, a powered gravity assist that is modeled as an impulsive thrust applied at periapsis during a hyperbolic flyby could net significant gains in orbital energy compared to performing the same thrust before or after entering the sphere of influence. Since orbital energy is rarely a mission parameter to be maximized or minimized in and of itself, the modeling of these powered gravity assists with regards to a interplanetary trajectory optimization is the focus of this thesis.

A burn at periapsis is typically the most optimal in terms of δv, as well as potentially netting significant gains in TOF [9]. Impulsive maneuvers at periapsis of an ellipse are thus shown to be the most efficient in terms of transferring energy with the same amount of rocket fuel; in addition, it is seen that the higher the apoapsis altitude, the higher the periapsis speed, thus the greater the efficiency of the Oberth effect [10]. The natural extension of this idea into parabolic and hyperbolic orbits therefore suggests that a higher v_∞ on arrival into a celestial body's SOI leads to a

higher velocity at periapsis (v_p) and thus a greater Oberth effect, though the increase in efficiency may be counteracted by the δv required to reach this higher v_∞ in the first place [11]. The position of greatest efficiency for impulsive maneuvers being at periapsis is backed up by Silva et al. in their discussion of Oberth effects around the Moon, as maximum variation of energy (ΔE_{max}) is found when the angle of the position vector of the impulsive maneuver relative to the periapsis vector is equal to zero [11].

2.2 Powered Gravity Assist Literature

One of the earliest papers on powered gravity assists was written by Antonio Fernando Bertachini de Almeida Prado in 1996. "Powered Swingby" compared three types of gravity assists:

- standard (i.e. unpowered) gravity assist using the two-body patched conic approximation
- powered gravity assist using the two-body patched conic approximation
- powered gravity assist using the restricted three-body problem

To begin, Prado simplifies the system to a 2D case. Prado then defines two celestial bodies to be the primaries. The first is the celestial body about which the second orbits. For example, in the Sun-Earth system, the first primary (M_1) is the Sun and the second primary (M_2) is the Earth. Finally, Prado sets the following assumptions for each of the three cases:

1. impulse is applied at periapsis

2. impulse changes velocity of the spacecraft instantaneously

3. motion is planar everywhere

The standard gravity assist was defined to provide a comparison to the powered gravity assists. Prado defined this maneuver with three independent parameters: 1) $v_{\infty-}$, the scalar magnitude of the velocity approaching the celestial body or v_p, the scalar magnitude of the spacecraft at periapsis (either can be calculated from the other, if necessary); 2) r_p, the distance between the spacecraft and the center of the celestial body during the closest approach (the periapsis radius of the trajectory); and 3) ψ, the angle of approach (the angle between the periapse line and the line that connects the two primaries). These variable definitions are shown in Fig.2.2:

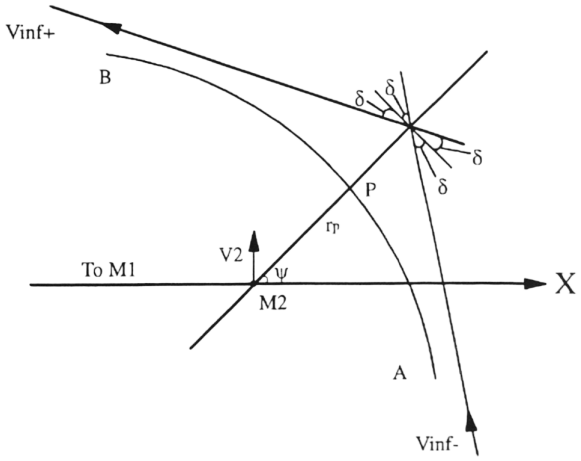

Figure 2.2: Standard Gravity Assist Diagram [12]

The A and B points are chosen to indicate the location at which the gravity of the M_2 overtakes that of M_1 (the edge of the SOI). Note that δ here is the angle between the hyperbolic asymptote and a line that is 90 degrees from the radius of periapse in the orbital plane. Some literature will instead define the turn angle as the angle between the hyperbolic asymptote and the radius of periapse itself; however, for the duration of this paper, any turn angle labeled δ will be defined the way it

was described in Prado (i.e. as shown in Fig. 2.2). This gravity assist maneuver then had a thrust applied as the spacecraft exits the SOI, which is then compared to a PGA with the same thrust applied at periapsis. For the PGA formulation, two additional variables are set: 1) δv, the scalar magnitude of the thrust applied at the periapsis and 2) α, the angle between the velocity of the spacecraft at periapsis and the direction of the thrust applied, with positive being defined away from the planet. With all of these vectors set, the exit v_∞ ($v_{\infty+}$) can be calculated through relatively simple orbital mechanical and geometric relationships. These variables are all shown in Figure 2.3[1]:

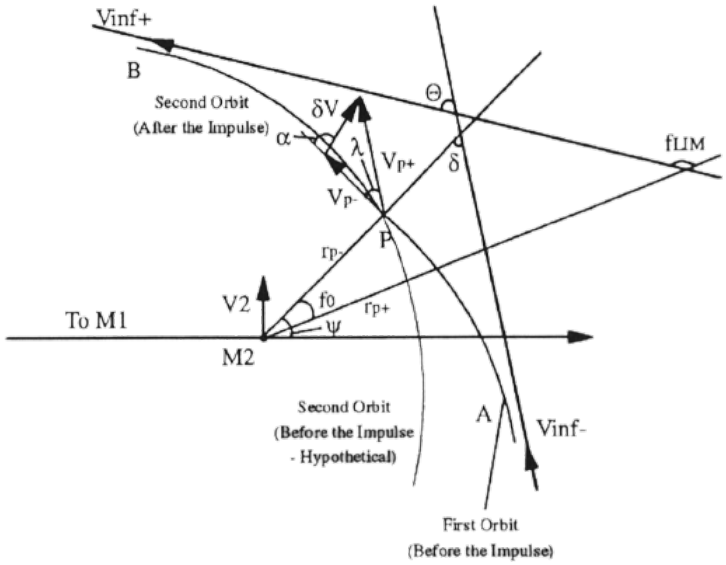

Figure 2.3: Powered Gravity Assist Diagram [12]

The main result of Prado's paper is the discovery that maximum transfer of velocity and energy occur when $\alpha \approxeq -20°$. When $\alpha = 0°$, there is the greatest direct transfer of velocity to the spacecraft, but Prado conjectured that when $\alpha \approxeq -20°$ the loss of velocity transfer due to the non-tangential component of the δv is overcome

[1]This figure is reproduced directly from Prado. Analysis of the formulas and geometry shows that the δ angle is incorrect and should be the δ angle as defined the standard gravity assist diagram rather than the angle labeled here.

by the gain of velocity due to a closer approach to M_2. This maximum change in energy or velocity could be up to 5% larger than the $\alpha = 0°$ case. As such, a PGA optimizer must be able to model a non-tangential velocity change to account for this to achieve ΔE_{max} [12]. The exact value for an optimal α depends on the exact case. Further, Prado discovered that in the majority of cases, the PGA was superior to an unpowered gravity assist with an impulse applied after, at the edge of the SOI, though this is not always true. Another interesting result is that at certain combinations of α and δv, the spacecraft is either captured into an elliptical orbit or collides with M_2. This is true when α is close to 180° or −180°. The α at which capture or collision occurs also getting smaller in size as δv increases. These trends are shown in Fig. 2.4, which is a contour plot of the ΔE from a specific powered flyby about the Moon:

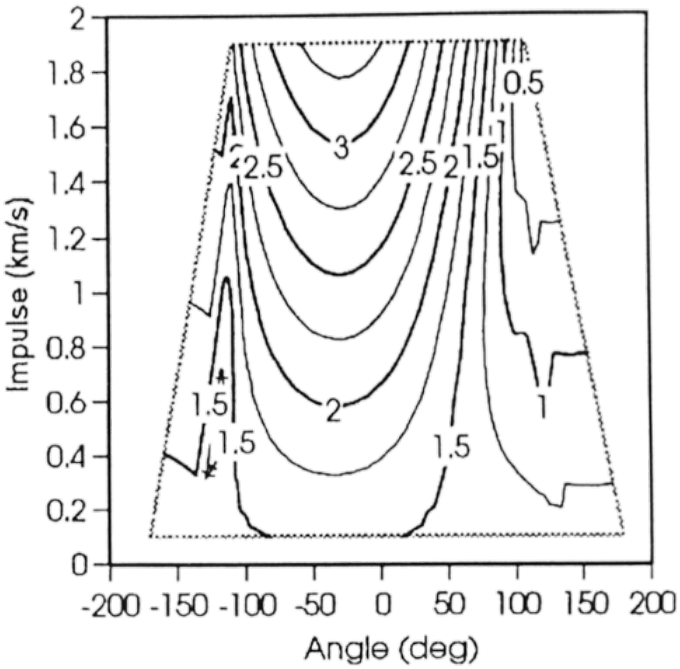

Figure 2.4: Example Capture/Collision Region [12]

The analysis of the same problem using the restricted three-body problem is another important aspect of Prado's paper. As with all restricted three-body problems, there is no analytical solution, so numerical integration solved this formulation of the

PGA maneuver. The change in energy and angular momentum were then compared to those from the two-body approximation. Prado concluded that with thrust angles between $-90° < \alpha < 90°$, the difference in Δv between the two-body and restricted three-body problem are less than 0.1 km/s, with the error growing larger as α strays further from this range. The maximum errors remain below 10% regardless of the α and δv values. Errors between the two formulations also decrease as δv increases. This analysis suggests that the two-body approximation tends to be acceptable, which fueled the decision to use that approximation in the algorithm used in STOpS.

This paper served as the basis of the PGA algorithm for STOpS. The variables defining $\vec{r_p}$ and $\vec{\delta v}$ were retained, along with the definition of the x-axis of the reference frame. Other than the use of these definitions, the PGA algorithm for STOpS was modified heavily to expand to 3D, in order to model changes to target celestial bodies with different inclinations. Specifically, two new angles were added to define $\vec{r_p}$ and $\vec{\delta v}$ directions both in plane and out of plane. However, by setting these two new angles to 0, validation could be performed by setting the remaining variables equal to those used in the test cases presented in this paper. Those validation test cases and their resulting ΔE and ΔV graphs are shown in the Validation chapter.

With co-authors Silva and Winter, Prado expanded on the work of "Powered Swingby [12]" in the 2013 paper "Powered Swing-by Maneuvers around the Moon". The variable definitions of Prado (1996) (specifically the restricted three-body problem formulation) were used and solved numerically. Silva et al. came to the same conclusions as Prado did alone when it came to the optimal angle to perform thrust if ΔE_{max} is desired. A selection of the numeric results for a lunar flyby with a r_p of 1.1 times the Moon's radius, a δv of 2 km/s, and thrust applied at periapsis is shown in Table 2.1:

Table 2.1: Maximum ΔE_{max} [11]

ψ	ΔE_{max}	α
90°	5.6036	0.0°
180°	3.8303	-6.5°
225°	6.0703	-27.3°
270°	8.6913	-22.0°
315°	10.2319	-10.2°

As shown, the actual α angle to achieve ΔE_{max} is dependent on the approach angle. Ultimately, however, ΔE is a means to an end and is typically a measure of the capability of the maneuver to change other parameters such as δv or TOF that are more relevant to the mission. In fact, Silva et al. noted that when targeting another body, it was often necessary to sacrifice ΔE_{max} for a smaller ΔE in exchange for the correct exit v_∞ direction [11]. If performing multiple powered gravity assist (MPGA) optimization, non-tangential impulsive thrust maneuvers can also help select the correct trajectory for the next segment of the flight. Finally, the decision to perform the thrust at periapsis for all the cases shown in Table 2.1 backed the same decision in the algorithm later developed for STOpS-FLITE.

While Silva et al. used the 2D model from Prado (1996), Prado and co-author de Felipe were able to expand into three-dimensions in "An analytical study of the powered swing-by to perform orbital maneuvers," [13]. The addition of another angle (i.e. β, defined as the smallest angle from the radius of perigee vector to the X-Y plane, as shown in Figure 2.5) increases the complexity of the problem, but also allows for some interesting optimization options, especially if an inclination change is desired. Note that the α angle utilized here is the same as the ψ angle from the previous two papers discussed.

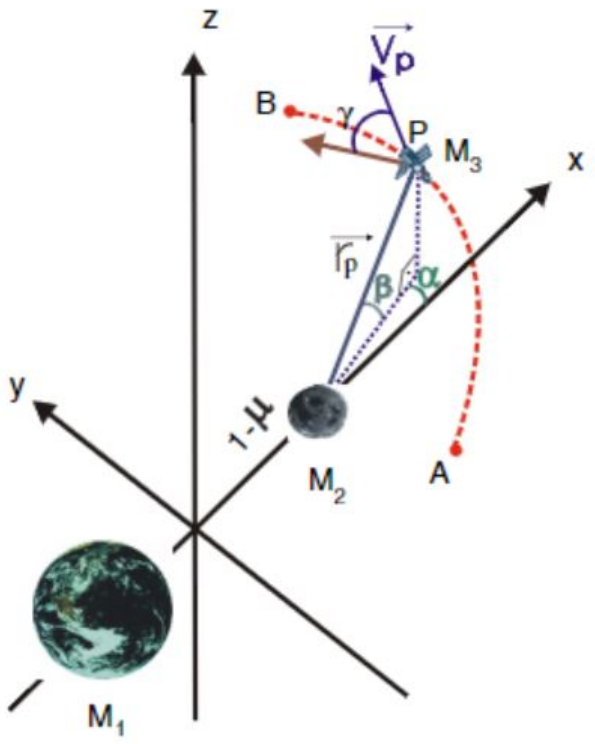

Figure 2.5: 3D Powered Gravity Assist Diagram [13]

The analytic equations were developed with only one angle defining the direction of thrust (γ). This angle was defined from the vector of the velocity at periapsis and out of the orbital plane. With this definition, the in plane angle cannot be set. This was considered acceptable since the paper was focused on using PGAs to change heliocentric energy or heliocentric velocity with inclination changes. This formulation would not work if utilized to target a specific exit v_∞, as desired by this paper.

Analytic solutions were generated for specifically the non-powered flyby from the patched conics approximation to calculate variations in velocity, angular momentum, and inclination after the swing-by, before a numerical verification via the restricted three-body problem was done, confirming errors of less than 1% for each of the parameters. The error between analytic and numerical solutions were below 1% for the powered fly-by modeling. With a small impulse, the variations in velocity, angular

momentum, and inclination are close to linear with respect to the variables δv_x and δv_y, which suggested to Prado et al. that the optimal efficiency maneuver is obtained when impulses are applied along the X or Y axis [13].

Qi and de Ruiter's paper "Powered Swing-By with Continuous Thrust" primarily focused on the implementation of impulsive thrust maneuvers in order to set a preliminary guess for their implementation of continuous thrust within the SOI [14]. Rather than the circular restricted three-body problem, they used the planar elliptic restricted Rather than using the SOI as the region in which the planet's gravity overtakes that of the Sun, they use the "circular neighborhood". This is a region defined by a radius R in Eq. 2.1:

$$R = \frac{R_{SOI}}{2a} \tag{2.1}$$

where R_{SOI} is the radius of the SOI and a is the semi-major axis of the planet's elliptical orbit about the Sun. Given that a is always greater than 1, R is always smaller than R_{SOI}. Qi and de Ruiter considered any continuous thrust to be a series of discrete optimal one-impulse powered swing-by segments. To initialize a guess for the optimal trajectory, Qi and de Ruiter begin with a single impulsive thrust applied at the edge of this circular neighborhood (i.e. when the spacecraft's position vector relative to the planet has a magnitude less than R). Note that this is well within the SOI, but also not performed at the periapsis. This was then expanded to a two impulse PGA in which the first impulse occurs at the edge of the circular neighborhood as before and the second impulse occurs somewhere before the spacecraft exits the circular neighborhood. Finally, the two impulse PGA was expanded to a continuous thrust maneuver. These are shown in Fig. 2.6:

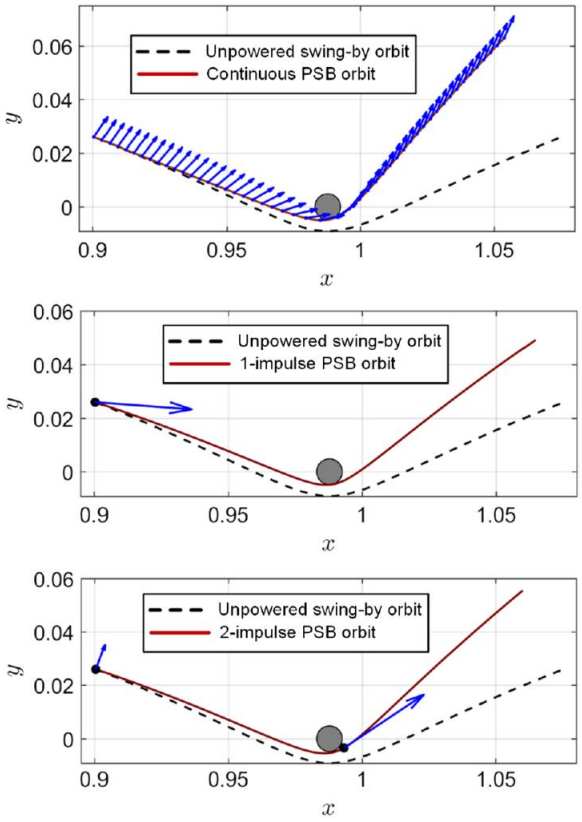

Figure 2.6: Multi-Impulse PSB Comparison [14]

The most notable discovery from Qi and de Ruiter's work that is relevant to this thesis is the fact that when the maximum thrust applied is greater than 0.003 m/s^2, a two-impulse maneuver can generate much greater energy gains than the corresponding one-impulse maneuver. This is likely because the first impulse is usually towards the planet (as shown in Fig. 2.6), resulting in a smaller radius of periapsis, thus an increase in velocity at periapsis and Oberth effect [14]. The second thrust is typically adding velocity (and thus energy) near or at periapsis. According to Qi and de Ruiter, these two impulse PGA can also be more efficient in changing orbital energy than an equivalent continuous PGA. Note that from an orbital mechanics standpoint, 0.003 m/s^2 is an incredibly small thrust, especially if considering an impulsive thrust. It was partly due to this paper that the PGA algorithm in STOpS allows for thrust to

be applied at the edge of the SOI (either just before entering or just after exiting the SOI). By doing so, it may be able to reduce the total δv of a maneuver.

2.3 Applied Powered Gravity Assists

The papers discussed above represent a small subset of the literature on the topic of PGAs. Specifically, those papers focused on the effectiveness of a PGA on either ΔV and/or ΔE. This means that they were not specific to use for interplanetary trajectory optimization. While considering these changes is important, maximizing them is not typically of benefit to the mission in and of themselves. The following papers instead focus on the use of PGAs to target specific trajectories, rather than simply maximizing heliocentric velocity or heliocentric energy.

Piñeros et al. combined PGA modeling with aerogravity assist (AGA) modeling to consider powered aerogravity assists (PAGAs) [15]. The work presented by Piñeros et al. largely focused on the modeling of AGAs, utilizing PGA modeling to account for drag losses to maintain, rather than change, total energy. Piñeros et al. were able to create generalized graphs based on a wide variety of magnitudes and directions of impulsive thrust. At certain higher magnitudes of δv, the impulsive thrust maneuver overshadowed the AGAs while lower magnitudes resulted in the opposite relationship, thus potentially allowing their PAGA optimizer to do AGAs or PGAs separately by setting lift-to-drag ratios (L/D) to 0 or δv of the impulsive thrust to 0. Piñeros et al. specifically pointed out that the impulsive thrust maneuver could be used to force ΔE to 0 in scenarios where atmospheric study by science instruments onboard the spacecraft were desired but changes in orbital energy were not. Because of the great variance between planetary atmospheres, the optimal L/D is very different between planets. As such, there are a large number of papers considering AGAs around

different planets, each with the same use of PGAs to maintain orbital energy through the atmosphere. While technically applying PGAs, these papers make the assumption that the thrust applied only maintains the same orbital energy. The thrust maneuver itself is not modeled beyond this assumption, but these papers deserve mention as interplanetary trajectory optimizers that utilize PGAs.

Ceriotti's PhD dissertation on multiple gravity assist (MGA) optimization touches briefly on powered fly-bys [16]. Comparisons were made to the Satellite Tour Design Program (STOUR) variants developed by Sims et al. and the mission-direct trajectory optimization program (MDTOP), both of which are capable of modeling PGAs. Ceriotti pointed out that changes in orbital parameters from PGAs are sensitive to the application of δv, since the real maneuver is not truly instantaneous, which can cause significant deviation from expected results. To match with the desired exit v_∞, a tangential maneuver is explored first, as it is often easiest to perform and typically most effective. However, Ceriotti suggests it is not always possible due to periapsis altitude restrictions of actual missions, thus a modeling of non-tangential maneuvers is also included to match the incoming and exiting v_∞, which are calculated separately in Ceriotti's work. In their discussion of the benefits and drawbacks of powered fly-bys, it is noted by Ceriotti that while there are many orbital advantages in terms of δv, there are also significant challenges and constraints in performing these maneuvers in the operations phase, citing the Cassini mission as an example. Thus, Ceriotti treats powered fly-bys as a necessary, but undesirable, tool to correct inequalities in the incoming and exiting v_∞. Nevertheless, Ceriotti was able to successfully model these and validate them against expected results from STOUR and MDTOP.

Finally, STOpS itself does not model PGAs, but applies a penalty δv for several conditions [17, 18]. Specifically, it applies a thrust at the edge of the SOI if the v_∞ into and out of the SOI generated by the Lambert's problem and planet velocities do not

match. This is similar to the comparison case used by Prado (1996) in that it is a two-step maneuver (i.e. the gravity assist is performed and then a thrust is applied). Once that penalty δv has been applied, STOpS checks if the spacecraft would collide with the planet or fly through its atmosphere on this trajectory. If it would have, it applies another penalty δv dependent on how low the flyby altitude was. This was meant to simulate the cost of a PGA to avoid these conditions without actually calculating the required PGA trajectory. While an acceptable compromise for previous versions of STOpS, STOpS-FLITE would show that this penalty δv tends to be smaller than the actual δv of the required PGA when fully modeled and calculated.

One mission that has already flown with a powered gravity assist is the Cassini mission [19]. However, it utilized a continuous burn of 96 minutes to slow down enough to be captured by Saturn's gravity, thus it is not well modeled with an impulsive thrust maneuver. Most other missions use their engines outside the SOI in order to target a specific ballistic flyby.

Chapter 3

SPACECRAFT TRAJECTORY OPTIMIZERS

Spacecraft trajectory optimization is a problem space with a near-infinite number of solutions which can often be incredibly difficult or mathematically impossible to solve for analytically [4]. All optimization tools require a way to analyze if a solution is the optimal one, namely a cost function. For spacecraft trajectory optimizers, the most common cost functions are to reduce δv or time of flight. These two are almost always included, although cost functions are incredibly mission dependent. Other relevant cost functions may account for the mission designers desiring as close a flyby as possible for scientific reasons or maximizing heliocentric energy or velocity to leave the solar system.

Even from one celestial body to another, a multitude of paths can be generated depending on a number of characteristics, including but not limited to: spacecraft capabilities, date of departure and arrival, and timing of thrust maneuvers [4]. The wide variety of variables to optimize has led to the development of a number of numerical methods, ranging from systems based on Pontryagin's principle developed in 1956 to more modern techniques such as biogeography-based optimization, which was first applied to spacecraft trajectory optimization in 2017 [20, 21]. Performing gravity assists increases the number of parameters which must be optimized, thus increasing the size of the problem space and the computational cost of optimizing the trajectory with numerical methods [17].

Traditional interplanetary trajectory optimization programs typically use classical approaches to the trajectory optimization problem, of which there are a wide variety of

categories and subcategories. Direct and indirect methods, single/multiple shooting and collocation methods, and linearized and nonlinear programming methods are just some of the techniques and systems used. Modern interplanetary trajectory optimization programs often rely on the usage of metaheuristic algorithms, such as evolutionary or swarm-intelligence algorithms. Hybridization (the use of multiple different algorithms which are allowed to share their solutions with each other in order to converge to the optimal solution) can minimize the weaknesses of any of these algorithms and is the basis for many modern spacecraft trajectory optimizers [17, 22].

3.1 Commonly Used Spacecraft Trajectory Optimizers

A variety of orbital trajectory optimizers have been developed for government and corporate use. NASA's Jet Propulsion Laboratory (JPL) has created a number of these which were condensed into the following table (Table 3.1) by Sheehan. Two additions briefly mentioned in the rest of Sheehan's work were also added to this table [23, 24, 25].

Table 3.1: JPL Trajectory Tools [23]

Name	Description
MALTO	Mission Analysis Low-Thrust Optimization
Mystic	Optimization of trajectory/entire missions
Copernicus	Generalized spacecraft trajectory design and optimization scheme
OTIS	Optimal Trajectory by Implicit Simulation
SNAP	Spacecraft N-body Analysis Program
CHEBYTOP	Chebyshev (Polynomial) Trajectory Optimization Program
VARITOP	Variational Calculus Trajectory Optimization Program
SEPTOP	Solar Electric Propulsion Trajectory Optimization Program (VARITOP-based)
NEWSEP	Newer version of SEPTOP
Sail	Solar sail optimization (VARITOP-based)
STOUR	Satellite Tour Design Program
GALLOP	Gravity-Assist Low Thrust Local Optimization Program

While these tools are effective at their strong suits, they are not all publicly available [24, 25]. MALTO, Copernicus, VARITOP and its variants, STOUR-LTGA, and GALLOP are all limited to NASA employees and universities with contractual affiliations to NASA, while Mystic is further limited to only NASA employees [23, 26, 27]. OTIS and SNAP are nominally available to anybody in government, academia, and industry, but are subject to export control regulations [23]. Finally, CHEBYTOP, the only truly publicly available software from Table 3.1, was last updated in the mid-1970s, is a combination of Excel and FORTRAN, and is the weakest of the tools in terms of accuracy [23].

Some tools developed outside of JPL are free, including General Mission Analysis Tool (GMAT), Java Astrodynamics Toolkit (JAT) [28], Skipping Stone [29], Parallel Global Multiobjective Optimizer (PaGMO) [30], and Python Global Multiobjective Optimizer (PyGMO) [30]. PaGMO and PyGMO are designed to be general application tools and require major modification to be applied to a orbital trajectory scenario [30]. Skipping Stone has important limitations to TOF, spacecraft mass, and number of gravity assists (which is capped at four) [29]. JAT is a combination of multiple individual functions that can be used to analyze missions [28]. It requires significant manipulation of the code base to be applicable to a specific trajectory problem [17, 28]. GMAT is similarly built for general mission design and includes an incredibly wide variety of capabilities, requiring significant understanding of the software to modify it to a specific trajectory optimization scenario. While a large number of other optimization tools exist, most are prohibitively costly, limited by government, academic, and/or corporate affiliation, or not directly applied to spacecraft trajectory optimization [24, 25].

3.2 Spacecraft Trajectory Optimization Suite (STOpS)

The Spacecraft Trajectory Optimization Suite (STOpS) was originally developed at California Polytechnic State University, San Luis Obispo to optimize orbital paths for spacecraft [17]. Specifically, it was designed to model and optimize multiple gravity assist trajectories, given a set of planets to use for gravity assists. Different variants of STOpS utilize different input variables to optimize, depending on their focus, but all require at minimum the ranges for departure time from the first planet and the time of flights for each leg [17, 18, 23, 22, 31, 32]. Multiple cost functions are built in to STOpS, with the most commonly used being time of flight, departure δv, and flyby penalty δv, though many more are included or can be added. STOpS was built on the island model paradigm adapted from the PaGMO and PyGMO with multiple metaheuristic algorithms forming the islands [17, 22]. STOpS uses this system to find the optimal value for each of the variables from the input to minimize the cost functions used [17, 22]. The island model paradigm and the algorithms used to accomplish this will be discussed in the following sections.

3.2.1 STOpS Optimization Scheme

STOpS is a collection of four optimization algorithms, combined within an island model paradigm [17, 22, 18]. These optimization algorithms are stochastic metaheuristic methods (genetic algorithm, differential evolution, particle swarm optimization, and ant colony algorithm) used to converge to a final optimized solution [17, 22, 18]. These algorithms are not sensitive to initial guesses as they start with a random input and evolve from there, making them well-suited to problem spaces like spacecraft trajectory optimization which may not have an obvious place to initialize searching.

3.2.2 Island Model Paradigm

STOpS utilizes an island model paradigm originally based on the one found in PaGMO [17]. The island model paradigm in STOpS allows multiple algorithms (to be discussed in the following section) to run simultaneously, then share and compare their solutions for the next iteration. This allows the different algorithms to converge more quickly, as each one can share its strengths while getting its own weaknesses mitigated by the others' strengths. Thus, the different algorithms feed off each others' strengths and overcome each others' weaknesses.

Each of the algorithms is an "island" within the island model paradigm. The way in which these islands are connected and which islands share information with which is called a "topology" or an "archipelago". Different topologies may be more effective in solving different problems. The sharing of solutions themselves is called "migration" and how often migrations occur is called the "migration policy". Each island can select certain solutions to be allowed to migrate to other island(s) through its own selection policy. Once these have been selected to migrate, the island(s) with which these solutions have been shared can decide whether to accept or deny these solutions as part of its replacement policy. Effective migration policies allow for improved convergence rates and decreased computational time. Policies that do not allow for enough sharing means that a well-performing algorithm can not properly share its strength, while too much sharing slows down the whole program, defeating the purpose of the island model paradigm. Good topology selection is difficult to achieve without some knowledge of the problem space. It may be most effective to run each algorithm alone first. A more informed decision can then be made on which algorithms to include in the final topology and how to connect them. Some examples of topologies utilized in literature can be found below in Figure 3.1:

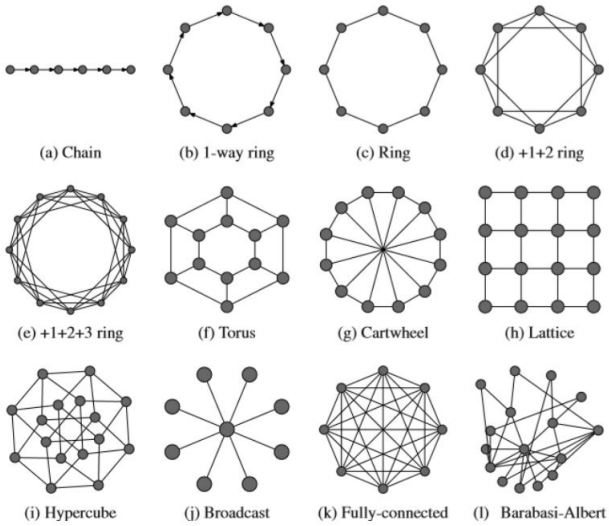

Figure 3.1: Example Island Model Topologies [22]

Migration policy is often divided into two options: synchronous and asynchronous. As its name suggests, synchronous migration policy means that all migrations occur at the same time (for this reason, this policy may also be referred to as a "simultaneous migration policy"). For this to occur, every algorithm must have run to completion, thus, an island model paradigm utilizing synchronous migration policy will run as slowly as the slowest algorithm. Once the slowest algorithm has obtained its solutions, connected islands will share, compare, and accept or reject solutions based on their selection and replacement policies [17, 22]. On the other hand, asynchronous migration policy allows each island to run as quickly as it can, without the need to wait for other islands. Asynchronous migration is further divided into sharer-driven and receiver-driven asynchronous migration. In the sharer-driven policy, as soon as an island finishes running, it sends its solutions to the islands connected to it. The receiving island then takes the solutions based on its replacement policy. If two islands are connected and one is much slower than the other, the slower one could be rendered useless as it will never finish running and thus never be able to share its solutions. Receiver-driven policy allows the algorithm to complete running before it takes in solutions, but depending on topology, certain islands could still be rendered

useless. Thus, despite being slower, synchronous migration policy is better, assuming that each algorithm included in the topology adds something of value.

Selection policy dictates which and how many solutions get shared. This policy can be random selection, natural selection, or weighted probabilities. Replacement policy dictates which and how many solutions replace solutions at the receiving island. Similar to selection policy, this can include random replacement, best solution replacement, or weighted probability/threshold replacement. Another possibility is for an island to only keep solutions that are better than those generated by itself.

3.2.3 Genetic Algorithm

The first evolutionary algorithm to be discussed herein is the genetic algorithm (GA). GA is a form of biomimicry which applies Darwin's theory of natural selection to converge to an optimal solution. An initial set of possible solutions (a "population") is generated randomly. These solutions are represented as a vector of variables. If a solution's cost is too great, it is not allowed to generate "offspring" for the next round. Those solutions with acceptable costs are selected to "mate". The selection process is what decides if a solution has an acceptable or unacceptable cost.

A wide variety of selection processes exist including, but not limited to: tournament method, natural selection, rank weight random, cost weight random, and thresholding. Tournament method divides the generation method into groups of a size specified by the user. Within each group, the lowest cost solution is allowed to mate. Natural selection ranks all the solutions from best to worst, with a user specified number of solutions from the top being allowed to mate until the population is back to the original size. Rank weight random and cost weight random are considered roulette selection methods. Solutions are given selection probabilities pro-

portional either to their rank or cost, respectively, with better solutions having higher probabilities. These are then selected from based on their selection probability for the mating process. Finally, thresholding allows all solutions above a certain threshold to progress to the mating process, but requires some user knowledge as to what the cost should be to be used properly.

The mating process mixes two good solutions from the current generation into one solution to move on to the next generation. During mating, mutations may occur, which creates solutions which were neither present in the previous generation nor offspring of two solutions in the previous generation. Mutation ensures that the algorithm does not converge too quickly to a local optimum. The mating process continues until a new generation of potential solutions of the same size as the previous generation is created. This process could continue forever, but it is limited either by a user-specified tolerance or a user-specified number of generations. Through the selection and mating processes, the GA converges towards an optimal solution.

From STOpS-PY forward, the GA also implements elitism, automatically allowing the best solution(s) to automatically survive to the next generation. These members are still part of the mating pool, but a user-specified number of solutions pass through to the next generation unchanged.

For further information and a more detailed description of GA and how it is implemented in STOpS, refer to the works by Fitzgerald and Graef [17, 22].

3.2.4 Differential Evolution

Like GA, differential evolution (DE) is an evolutionary algorithm, utilizing the current population to create the next generation, with each solution being represented

as a vector of variables. The main differences are in how solutions are selected to progress and how the next generation is created.

DE creates new generations through processes called mutation (not to be confused with mutation in GA) and recombination. The solutions of the next generation are created by combining multiple different parent solutions from the current generation, as shown in Equation 3.1. The DE algorithm creates a difference vector by subtracting two vectors ($\vec{x}_{r1}$ and $\vec{x}_{r2}$) in the current generation. This difference vector is multiplied by a scaling factor (F) before being added to another vector from the current generation called the root vector ($\vec{x}_{r0}$). The resulting vector ($\vec{V}_i$) is called the mutant vector.

$$\vec{V}_i = \vec{x}_{r0} + F(\vec{x}_{r1} - \vec{x}_{r2}) \tag{3.1}$$

After each mutant vector is formed, a trial vector is generated. The i^{th} trial vector takes its trait from either the i^{th} mutant vector or the i^{th} current generation member in a process called recombination. After recombination, the DE algorithm will have created twice as many trial solutions as there were parent solutions. This new population is then pared down through a selection process. The same selection processes used in GA are used for DE, with the goal of reducing this new population down to its original size. This process continues until a user-specified tolerance is met or a user-specified number of generations is reached. Like with GA, from STOpS-PY forward, the DE implements elitism, automatically allowing the best solution(s) to automatically survive to the next generation. These elite solutions are still included in the pool for mutation, but also move on to the next generation unchanged.

For further information and a more detailed description of DE and how it is implemented in STOpS, refer to the works by Fitzgerald and Graef [17, 22].

3.2.5 Particle Swarm Optimization

Particle swarm optimization (PSO) is a swarm-intelligence algorithm. Unlike evolutionary algorithms, swarm-intelligence algorithms do not create new generations, but rather the population members change parameter values over time to explore the search space. PSOs are another form of biomimicry that search for solutions in the same way that bees search for flowers and pollen. As before, solutions are represented by a vector of length n, with n being the number of variables being optimized. For the purposes of STOpS, the idea of a "hive" for each bee to start from and return to is unnecessary, thus each bee starts at a random location. Each bee (or particle, hence PSO) has a position represented in n-dimensional space, with some velocity also of length n. Each particle starts with a random initial velocity and as time progresses, these particles move about the search space to find solutions. Particles will communicate with each other to utilize swarm-intelligence to converge to an optimal solution. When considering the i^{th} particle, the particles which are providing information to it are called informants. Each particle is given a confidence in its own velocity (c_1), confidence in the best location it has discovered itself (c_2), and the best location received from an informant (c_3). The i^{th} particle has position x_i and velocity v_i, while the best location it has discovered is designated p_i and the best location discovered by an informant is designated g_i. The relationship between these is demonstrated by Equation 3.2 and the position of the i^{th} particle in the next time step is provided by Equation 3.3.

$$\vec{v_i} = c_1\vec{v_i} + c_2(\vec{p_i} - \vec{x_i}) + c_3(\vec{g_i} - \vec{x_i}) \tag{3.2}$$

$$\vec{x_i}^{t+1} = \vec{x_i}^t + \vec{v_i}^t \tag{3.3}$$

The number of particles which communicate with each other is carefully selected. If too many particles communicate, the best solution found thus far will dominate, leading to premature convergence. Meanwhile, too little communication leads to a search method which is too random to lead to proper convergence. Thus, the number of informants for each particle needs to be small enough to promote a sufficient exploration of the search space, but large enough that convergence can be achieved.

For further information and a more detailed description of PSO and how it is implemented in STOpS, refer to the works by Fitzgerald and Graef [17, 22].

3.2.6 Ant Colony Optimization

The final optimization algorithm used by STOpS is ant colony optimization (ACO). It is occasionally abbreviated ACA for ant colony algorithm. The two terms are used interchangeably within this paper. Another example of biomimetic swarm-based intelligence, ACO simulates how ants optimize their paths between their nest and food. Ants communicate through the deposition of a chemical called pheromone, which evaporates over time. On a path which is determined to have a good outcome, an ant will lay down more pheromone. On a path which is determined to have a bad outcome, an ant will lay down less pheromone. A path with higher pheromone levels leads to a later ant having a higher probability of following it. Thus, the best trails will eventually build up a large number of pheromone while the worse trails will have their pheromone dissolve away over time. When the ACO reaches an end condition (typically, a number of iterations), one path will have the most pheromone, making it the most optimal solution to the given problem.

ACO is most often considered as a solver for the traveling salesperson problem (TSP), where each destination the ant travels to is called a city or node. In a tradi-

tional TSP, ants randomly travel to an initial node before then visiting every other node and returning to the node at which they started. Based on the efficiency of the tour, ants change the amount of pheromone at each node. In ACO, simulated ants know how far away each node is and can only visit each node once, unlike real ants.

A traditional TSP is not the best representation of spacecraft trajectories, as they are typically one-way rather than round-trip. Each ant travels from the first planet to the last, altering their pheromone levels based on the costs associated with each of the paths. As with the other optimization algorithms, there are settings which need be carefully balanced. Here, it is the evaporation rate and pheromone matrices which can greatly affect premature convergence or failure to converge. If the pheromone evaporates too quickly the ants will explore the search too randomly, while too much pheromone being deposited along a path will result in premature convergence to that path.

For further information and a more detailed description of ACO and how it is implemented in STOpS, refer to the works by Fitzgerald and Rockett [17, 18].

3.2.7 History of the Spacecraft Trajectory Optimization Suite

The original version of STOpS (STOpS-1) did not model the SOI of each of the planets; rather, it bent the incoming approach vector by the angle expected from a gravity assist of a specific altitude [17]. STOpS-1 was effective at finding comparable solutions to problems solved in historic missions [17]. Despite not modeling the SOI, STOpS-1 still converged to the same optimal trajectory for missions which employed multiple gravity assists such as Voyager 1 [17]. Other versions of STOpS expanded to model these SOIs, but continued to perform thrust maneuvers solely outside of them [23, 22]. While the optimal point to perform these maneuvers can be outside

the SOI, past missions have shown this is not always true [19]. Sheehan's work (STOpS-LT) added low thrust modeling capabilities but was limited to applying the thrust during the interplanetary segments of a planetary transfer, thus avoiding the need to model thrust maneuvers within the SOI [23]. Malloy's work (STOpS-MGALT) implemented the multiple gravity assist capability to STOpS-LT, while adding a monotonic basin hopping algorithm as an optimization island [31]. STOpS-MGALT again used a point-mass assumption for the planets, removing the need to model the SOIs [31]. The last of the STOpS variants in MATLAB was Woods's work which added environmental perturbation modeling to STOpS-1 [32]. Graef's work (STOpS-PY) converted STOpS from MATLAB into Python, in keeping with the ultimate goal of providing STOpS to the public, as Python is freely available and generally runs quicker than MATLAB [22]. STOpS-PY removed the gravity assist capability to implement a B-plane targeting and thrust correction maneuver (TCM) capability [22]. STOpS-PY and STOpS-MGALT both modeled the SOIs of the planets, but neither had an option to conduct thrust maneuvers within these SOIs [22, 31]. Finally, Rockett's work (STOpS-DSM) focused on modeling deep space maneuvers (DSMs), which by definition can not occur within an SOI [18]. The two newest variants of STOpS were created in 2022, including the one described by this paper (STOpS-FLITE) and a variant which looks to implement pseudostate theory (STOpS-PSI). All of these variants are tabulated below (in chronological order) in Table 3.2:

Table 3.2: Current Variants of STOpS

Name	Author	Language	Purpose
STOpS-1 [17]	Tim Fitzgerald	MATLAB	Original
STOpS-LT [23]	Shane Sheehan	MATLAB	Low Thrust
STOpS-EP [32]	Eric Woods	MATLAB	Environmental Perturbation
STOpS-MGALT [31]	Michael Malloy	MATLAB	Multiple gravity assist + low thrust
STOpS-PY [22]	Jared Graef	Python	B-plane + thrust correction maneuvers
STOpS-DSM [18]	Elliott Rockett	Python	Deep space maneuvers
STOpS-PSI [33]	Dominick Bologna	Python	Pseudostate theory
STOpS-FLITE	Aaron Li	Python	Impulsive thrust fly-bys

The best version to use is dependent on the mission. For example, a mission using low thrust systems would be best optimized with STOpS-LT or STOpS-MGALT. Note that while each version is built from the previous, not every one has the capabilities of the previous versions. Since STOpS-FLITE is designed to study PGAs only, it does not include STOpS-PY or STOpS-DSM's capabilities to model thrust correction maneuvers or deep space maneuvers without significant modification to the code base.

It is also important to mention Doughty's work on automated flyby sequences using STOpS. Though not a variant of STOpS on its own, the findings presented within that work provide context for the continued use of fixed orbit flyby sequences. STOpS requires a defined planetary body sequence [17, 22]. This leads to a constant length set of design variables. Automating this process means exploring a much larger search space, since an optimal trajectory could require any number of flybys. Early in the design of spacecraft trajectories, this system could find an optimal solution not considered by mission designers or guide mission designers when the optimal flyby sequence is not easy to find without *a priori* knowledge. The Hybrid Optimal Control Problem (HOCP) was utilized to compare performance within a variable size design space (VSDS). Unfortunately, the HOCP was incredibly computationally expensive, even with only a single evolutionary algorithm, which would become an even greater problem with the multiple optimization algorithms utilized in almost every variant of STOpS. A Hidden Gene Algorithm was implemented which improved the computation time. Ultimately, the conclusion drawn was that even with tight variable bounds, the system requires the user to either have some knowledge of the desired solution (defeating the main purpose of automating the flyby sequence) or commit to extensive and time-consuming testing and manipulation of the input variables. For more information on the methodology and other findings, the interested reader is referred to Doughty's thesis "Interplanetary Trajectory Optimization with Automated Fly-by Sequences" [34].

Chapter 4

POWERED GRAVITY ASSIST MODEL AND ALGORITHM

The powered gravity assist model generated for STOpS takes large portions of Prado's work in "Powered Swingby" and adds elements of the other works discussed in the Powered Gravity Assists chapter. Primarily, the additions help expand Prado's formulation into three dimensions and adjust the algorithm to be more robust in searching for optimal trajectories.

4.1 Required Inputs

The algorithm developed for STOpS-FLITE utilizes the same variables defined in "Powered Swingby" (r_p, $v_{\infty A}$, ψ, α, and δv), while adding two new variables to represent the out of plane angles (ϕ and β). These variables are listed below:

- ${v_{\infty A}}^1$ = scalar magnitude of velocity of spacecraft relative to planet when entering the sphere of influence
- r_{pA} = scalar magnitude of radius of periapsis
- ψ = angle between x-axis of PCSF frame and projection of $\vec{r}_p$ on to x-y plane of PCSF frame
- ϕ = angle between projection of $\vec{r}_p$ on to x-y plane of PCSF frame and $\vec{r}_p$
- α = angle between y-axis of OP_A frame and projection of $\vec{\delta \text{v}}$ on to x'-y' plane of OP_A frame

[1]Subscripts A and B will be used to differentiate parameters relating to the spacecraft trajectory before the impulsive thrust is applied (A) and after the impulsive thrust is applied (B).

- β = angle between projection of $\vec{\delta \text{v}}$ on to x'-y' plane of OP_A frame and $\vec{\delta \text{v}}$
- δv = scalar magnitude of thrust applied at periapse

The frames and axes mentioned in these parameter definitions will be explained and defined in the following paragraphs. $v_{\infty A}$ can be calculated from a specific $\vec{v}_{\infty A}$ from the heliocentric inertial frame, which is given by the Lambert's problem utilized by STOpS. r_p, ψ, ϕ α, β, and δv are randomly selected by STOpS as part of the population being optimized.

4.2 Definition of Incoming Hyperbolic Orbit

STOpS propagates interplanetary trajectories in the ICRS frame. This is an excellent frame for the heliocentric elliptic segments of the patched-conics approximation, but not a helpful frame for PGAs. Thus, a rotation to a frame named the sun centered planet fixed (SCPF) is performed. The SCPF frame is a reference frame centered on the Sun (interchangeable with M_1 within this paper), with the x-axis of said frame fixed to the center of mass of the planet that is being utilized for the gravity assist, the y-axis directed along the velocity of the planet (assuming the planet is traveling in a circular orbit), and the z-axis completing the 3D reference frame through the right hand rule. Note that this is nearly identical to the frame described in "Powered Swingby". However, rather than setting the center of the frame to be the barycenter of the Sun-planet system, the Sun's center is set as the origin of this reference frame. Setting the barycenter as the origin of the reference frame is more common with three-body assumptions. Since this algorithm will utilize the two-body assumption, this is not necessary. Ultimately, the definition of the x-axis here is the main similarity to Prado's work. This definition is the same, regardless of if the frame originates on the Sun's center or the system barycenter. This frame is shown in Fig. 4.1:

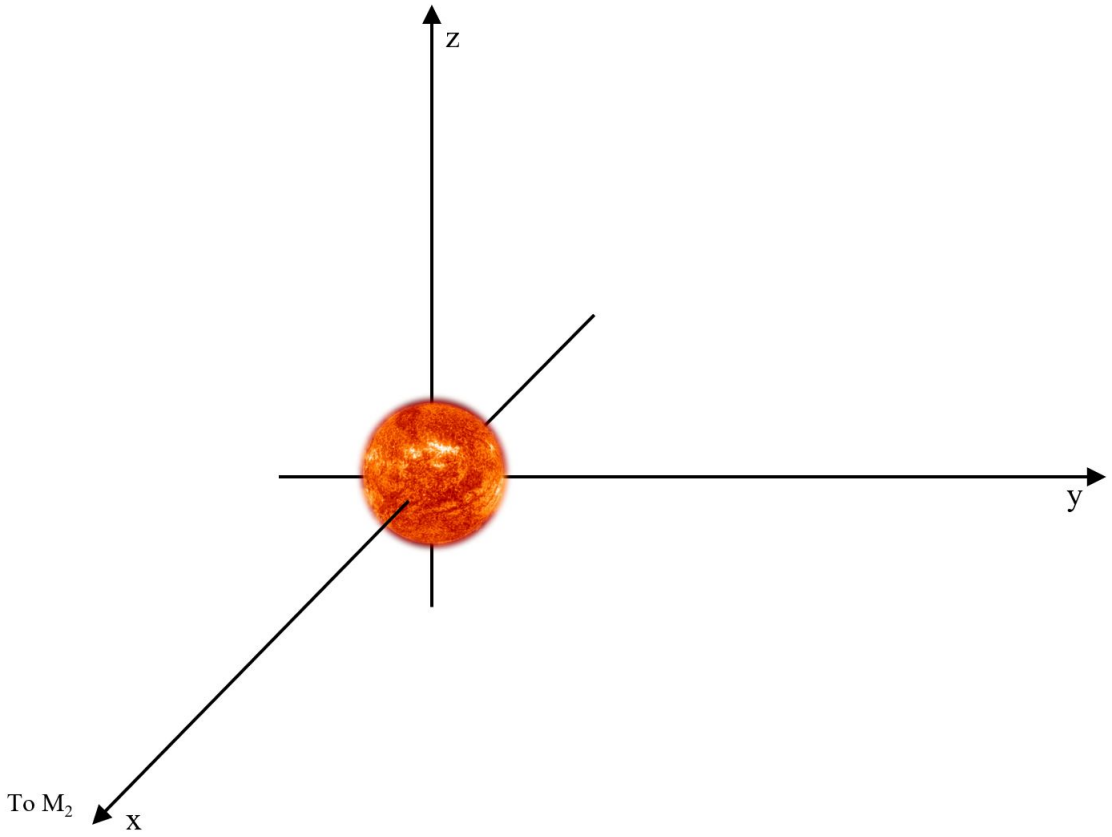

Figure 4.1: Sun Centered Planet Fixed Frame

The planet centered sun fixed frame (PCSF frame) is a direct shift of origin from the sun centered planet fixed frame (SCPF frame). The PCSF frame simply moves the center of this frame to the planet in question, with the x-axis defined as from M_2 directly away from M_1, the y-axis directed along the velocity of the planet, and the z-axis completing the 3D reference frame through the right hand rule once again. The PCSF frame is also utilized in Prado's work as the frame in which gravity assists are performed when a two-body assumption is utilized. This frame is shown in Fig. 4.2:

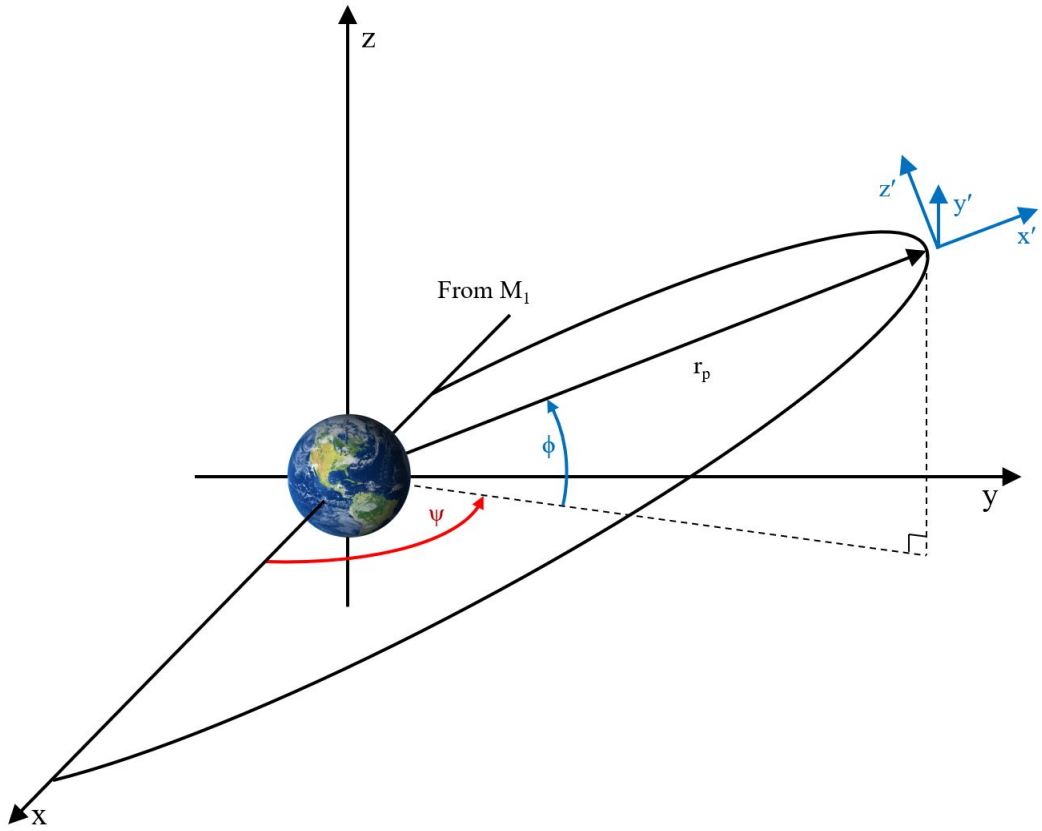

Figure 4.2: Planet Centered Sun Fixed Frame

A parameter of this orbit needs to be calculated from the inputs using Eq. 4.1: the velocity at periapsis (v_{pA})

$$v_{pA} = \sqrt{v_{\infty A}^2 + \frac{2\mu_2}{r_{pA}}} \tag{4.1}$$

where $v_{\infty A}$ is the magnitude of the velocity vector as the spacecraft enters the sphere of influence and μ_2 is the gravitational parameter of M_2.

For comparison later, $\vec{v}_{\infty A}$ will also need to be calculated. This can be done by finding the turn angle δ_A with Eq. 4.2, and combining with the ψ and ϕ angles to find the vector, given the magnitude from the inputs. Recall the definition of the

turn angle δ (i.e. δ_A and δ_B, in this algorithm) from the Powered Gravity Assists chapter. Simply by rotating the x-axis of the PCSF frame by these angles, the unit vector $\frac{\vec{v}_{\infty A}}{||\vec{v}_{\infty A}||}$ can be found, which can then be multiplied by the magnitude of $\vec{v}_{\infty A}$ to give the final $\vec{v}_{\infty A}$. Thus, all that is needed to be calculated is δ_B before the rotation matrices are applied to the x-axis. A new variable, τ is also defined by Eq. 4.3 to simplify the rotation. τ represents the angle between the radius of periapsis vector and the hyperbolic asymptote of the spacecraft approach vector, represented by $\vec{v}_{\infty A}$.

$$\delta_A = \sin^{-1}\left(\frac{1}{1+\frac{r_{pA}V_{\infty A}^2}{\mu_2}}\right) \tag{4.2}$$

$$\tau = \frac{\pi}{2} - \delta_A \tag{4.3}$$

Then, the rotation matrices can be applied, resulting in Eq. 4.4:

$$\vec{v}_{\infty A} = R_z(\tau)R_y(-\phi)R_z(\psi)\begin{bmatrix}1\\0\\0\end{bmatrix} v_{\infty A} = \tag{4.4}$$

Multiplying out Eq. 4.4 results in Eq. 4.5:

$$\vec{v}_{\infty A} = \begin{bmatrix}-\sin(\psi)\sin(\tau)+\cos(\phi)\cos(\psi)\cos(\tau)\\ \sin(\psi)\cos(\tau)+\cos(\phi)\cos(\psi)\sin(\tau)\\ \cos(\psi)\sin(\phi)\end{bmatrix} v_{\infty A} \tag{4.5}$$

4.3 Application of Thrust

In the PCSF frame, the radius of periapsis of the incoming hyperbolic orbit can be defined by the magnitude of periapsis radius and the angles ψ and ϕ. These values

can also be obtained from a specific incoming $\vec{V}_\infty$, if necessary. A third reference frame is defined, which will be called the OP_A frame (short for orbital plane A). It is centered on the spacecraft, with the x'-axis pointed directly away from M_2, the y'-axis directed along the spacecraft velocity vector, and the z'-axis completing the right hand rule (by definition, this is the direction of the angular momentum vector $\vec{h}_A$). This frame is shown in blue on Fig. 4.2 as well as in greater detail in Fig. 4.3

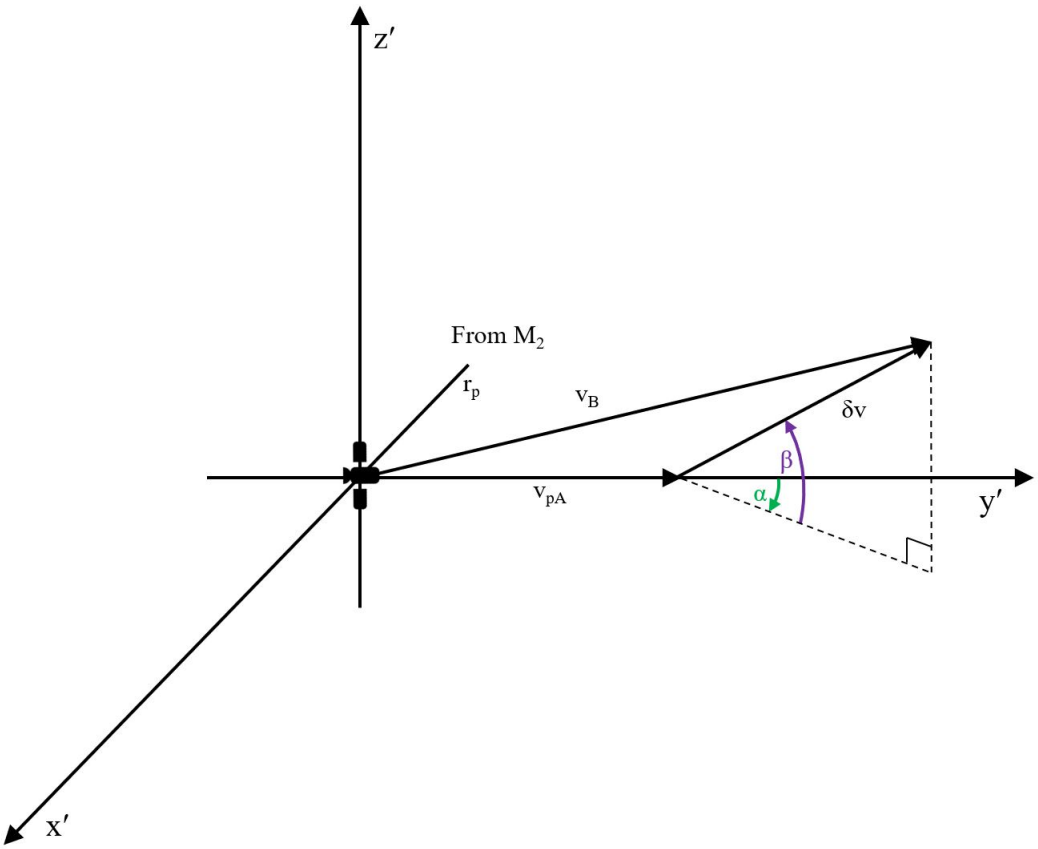

Figure 4.3: Orbital Plane A Frame

Here, the velocity before the thrust is applied is $\vec{v}_{pA}$. The thrust δv is applied at the radius of periapsis of the orbit from before the thrust is applied, though it is shown attached to the end of $\vec{v}_{pA}$ for the sake of showing the relationship between $\vec{v}_{pA}$, $\vec{v}_{pB}$, and $\vec{\delta v}$. $\vec{\delta v}$ is defined in this frame by δv, α, and β. In this frame v_{pA} is

directly along the y-axis, resulting in the following definition of $\vec{v}_{pA}$ in Eq. 4.6:

$$\vec{v}_{pA} = \begin{bmatrix} 0 \\ v_{pA} \\ 0 \end{bmatrix} \tag{4.6}$$

Analyzing the geometry of Figure 4.3 gives Eq. 4.7:

$$\vec{v}_{pB} = \begin{bmatrix} 0 \\ v_{pA} \\ 0 \end{bmatrix} + \begin{bmatrix} \cos(\beta)\sin(\alpha) \\ \cos(\beta)\cos(\alpha) \\ \sin(\beta) \end{bmatrix} \delta v = \begin{bmatrix} \delta v(\cos(\beta)\sin(\alpha)) \\ v_{pA} + \delta v(\cos(\beta)\cos(\alpha)) \\ \delta v(\sin(\beta)) \end{bmatrix} \tag{4.7}$$

4.4 Calculation of Outgoing Exit Velocity

The magnitude of $\vec{v}_{pB}$ is required for the calculation of $\vec{v}_{\infty B}$ and is simply the norm of $\vec{v}_{pB}$. The magnitude of the velocity of the spacecraft as it exits the sphere of influence can then be calculated, though the vector direction will take further calculation. This calculation is accomplished through Eq. 4.8:

$$v_{\infty B} = \sqrt{v_{pB}^2 - \frac{2\mu_2}{r_{pA}}} \tag{4.8}$$

With the thrust applied, the spacecraft is in a new orbit, one in which the radius of periapsis from before the thrust applied is no longer the radius of periapsis of the new orbit (except in the case of α and β equaling 0 or 180). A new frame to define this new orbital plane is used to make calculations easier to conceptualize. This frame is designated OP_B, in keeping with the subscript nomenclature defined before. The OP_B frame is defined with the x"-axis aligned with the x'-axis from the OP_A frame, the z"-axis along the direction of the angular momentum vector $\vec{h}_B$ of the new

orbit, and the y"-axis completing the right hand rule. $\vec{h}_B$ is calculated from the cross product of $\vec{r}_{pA}$ and $\vec{v}_{pA}$, as shown in Eq 4.9. Because of this, the z"-axis will always be in the x'-y' plane from OP_A, as $\vec{h}_B$ must be orthogonal to $\vec{r}_{pA}$.

$$\vec{h}_B = \vec{r}_{pA} \times \vec{v}_{pB} \tag{4.9}$$

Plugging in what is known about these values, Eq. 4.9 becomes Eq. 4.10

$$\vec{h}_B = \begin{bmatrix} r_{pA} \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} \delta v(\cos(\beta)\sin(\alpha)) \\ v_{pA} + \delta v(\cos(\beta)\cos(\alpha)) \\ \delta v(\sin(\beta)) \end{bmatrix} = \begin{bmatrix} 0 \\ -\delta v(r_{pA})(\sin(\beta)) \\ (r_{pA})(v_{pA} + \delta v(\cos(\alpha)\cos(\beta))) \end{bmatrix} \tag{4.10}$$

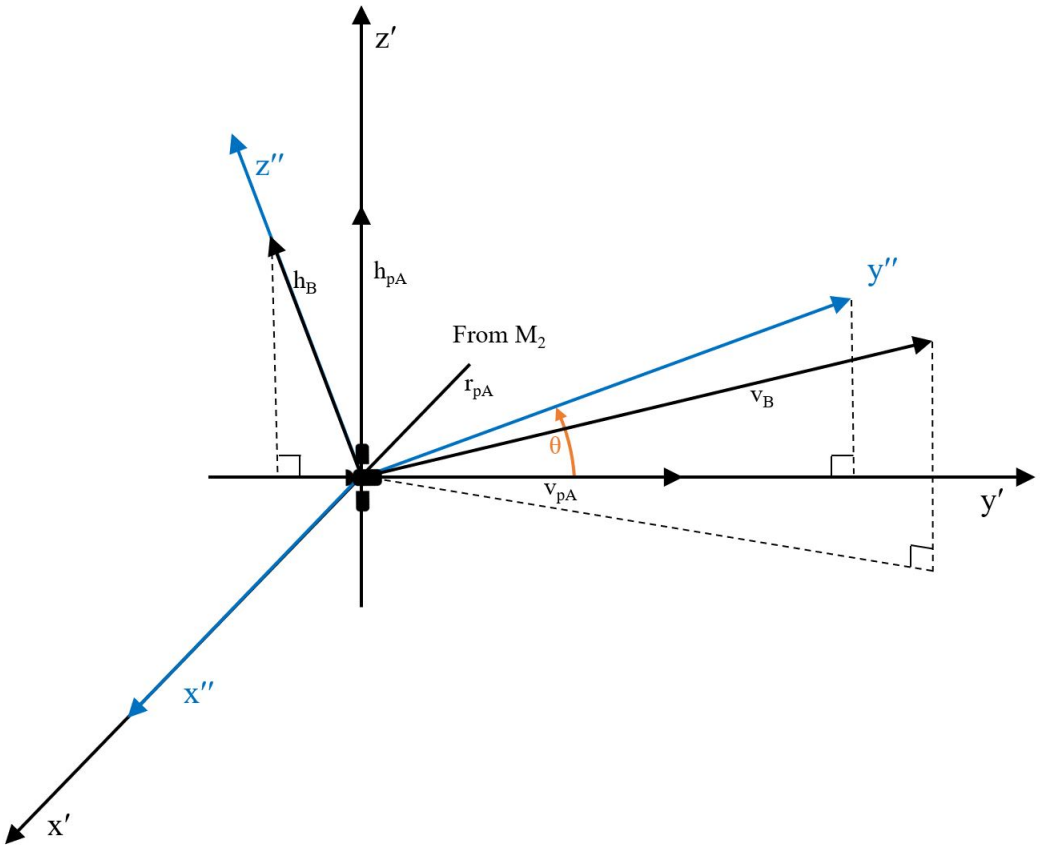

Figure 4.4: Orbital Plane B Frame

Several parameters can be found in the orbital frame to assist in the calculation of the direction of $\vec{v}_{\infty B}$. These parameters are labeled in Figure 4.5. Specifically, the angle between the current position of the spacecraft and the periapsis position (f_0) needs to be calculated. In the pursuit of that goal, several parameters must be calculated first, using Eqs. 4.11, 4.12, and 4.13:

$$P_B = \frac{h_B^2}{\mu_2} \tag{4.11}$$

$$a_B = \frac{\mu_2}{v_{\infty B}^2} \tag{4.12}$$

$$e_B = \sqrt{1 + \frac{P_B}{a_B}} \tag{4.13}$$

where P_B is the semi-latus rectum, a_B is the semi-major axis, and e_B is the eccentricity of the post-thrust hyperbola.

With these parameters calculated, f_0 is calculated using Eq. 4.14:

$$f_0 = \cos^{-1}\left(\frac{1}{e_B}\frac{P}{r_{pA}} - 1\right) \tag{4.14}$$

The OP_B frame, by definition, has its x"-y" plane in the orbital plane of the post-PGA orbit. Thus, the post-thrust orbit can be considered to be solely in the x"-y" plane in this frame, as shown in Figure 4.5:

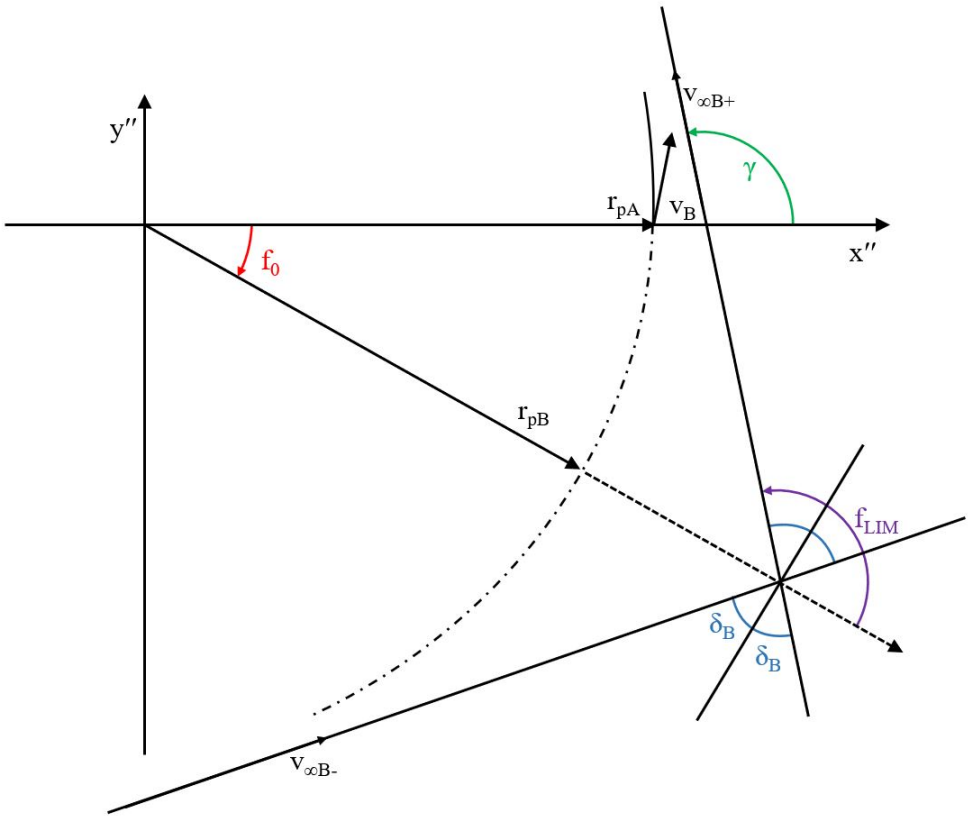

Figure 4.5: Post-PGA Orbit in OP$_B$

The goal is to obtain $\vec{v}_{\infty B}$ from given parameters, so the x"-axis undergoes a rotation of γ about the z"-axis to give the direction of $\vec{v}_{\infty B}$ in OP$_B$. After calculating f_{LIM} (the angle between the radius of periapsis vector and the direction of $\vec{v}_{\infty B}$ in OP$_B$) using Eq. 4.15, γ can be calculated through Eq. 4.16. Note that in a ballistic gravity assist that f_{LIM} is the same as the angle between

$$f_{LIM} = \cos^{-1}\left(-\frac{1}{e_B}\right) \tag{4.15}$$

$$\gamma = -f_0 + f_{LIM} \tag{4.16}$$

Once again using the x-axis as a starting point, the unit vector $\frac{\vec{v}_{\infty B}}{||\vec{v}_{\infty B}||}$ can be attained through several rotation matrices, similar to the process used for $\frac{\vec{v}_{\infty A}}{||\vec{v}_{\infty A}||}$. As with that unit vector, it can be multiplied by the magnitude to obtain the final vector $\vec{v}_{\infty B}$. This set of rotation matrices is shown in Eq. 4.17:

$$\vec{v}_{\infty B} = R_z(\gamma)R_x(-\theta)R_y(-\phi)R_z(\psi) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} v_{\infty B} \tag{4.17}$$

Multiplying out Eq. 4.17 results in Eq. 4.18:

$$\vec{v}_{\infty B} = \begin{bmatrix} (\cos(\psi)(\cos(\gamma)\cos(\phi) - \sin(\gamma)\sin(\phi)\sin(\theta)) - \cos(\theta)\sin(\gamma)\sin(\psi)) \\ (\cos(\psi)(\sin(\gamma)\cos(\phi) + \cos(\gamma)\sin(\phi)\sin(\theta)) + \cos(\theta)\cos(\gamma)\sin(\psi)) \\ \cos(\psi)\cos(\theta)\sin(\phi) - \sin(\psi)\sin(\theta) \end{bmatrix} v_{\infty B} \tag{4.18}$$

Thus, the algorithm provides the single output $\vec{v}_{\infty B}$:

- $\vec{v}_{\infty B}$ = vector of the velocity of the spacecraft relative to planet when exiting the sphere of influence, in the PCSF frame.

Chapter 5

VALIDATION

The PGA algorithm built for STOpS was based primarily on the 2D model developed by Prado in 1996, with significant changes to accommodate the third dimension [12]. Thus, the results presented in Prado were used as the validation when out of plane angles of the 3D algorithm were set to zero.

Prado displayed two test cases in their paper, both with the spacecraft performing a flyby around the Moon as M_2 with the Earth as the M_1 [12]. One of those test cases will be displayed below, utilizing the following parameters in Table 5.1:

Table 5.1: Prado Model Example Parameters [12]

Parameter	Value	Units
μ_1	4900	km^3/s^2
μ_2	398600	km^3/s^2
V_2	1.02	km/s
$V_{\infty-}$	1	km/s
δv	0 to 4	km/s
ψ	270	deg
α	-150 to 150	deg

From that paper, the following graphs (Fig. 5.1 and Fig. 5.2) were pulled for ΔV and ΔE, using the parameters tabulated above. Prado noted and showed in their paper that specific combinations of extreme angles of alpha (i.e. $\alpha \gtrapprox 150$ and $\alpha \lessapprox 150$) and δv resulted in capture or collision with the flyby body. Those regions are represented by the blank region outside of the trapezoid formed by the dotted line.

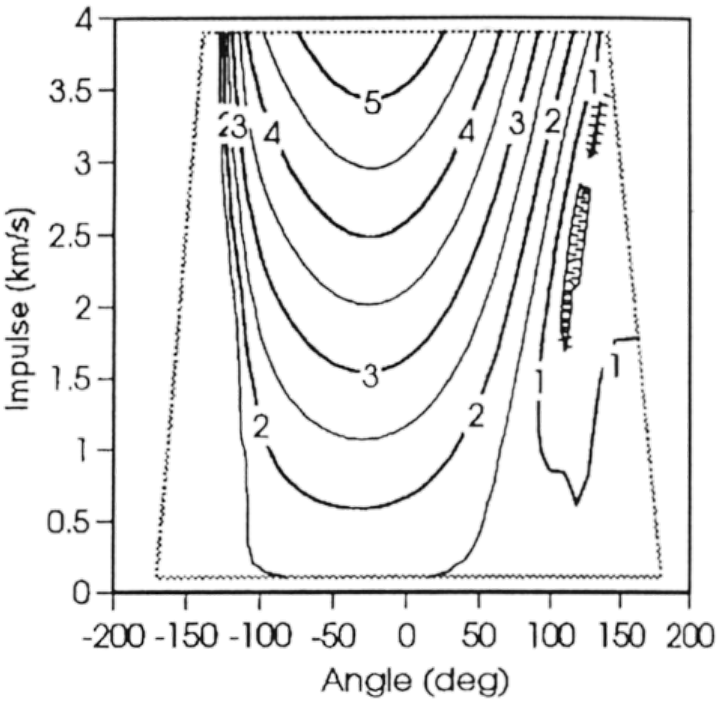

Figure 5.1: Change in Heliocentric Velocity [12]

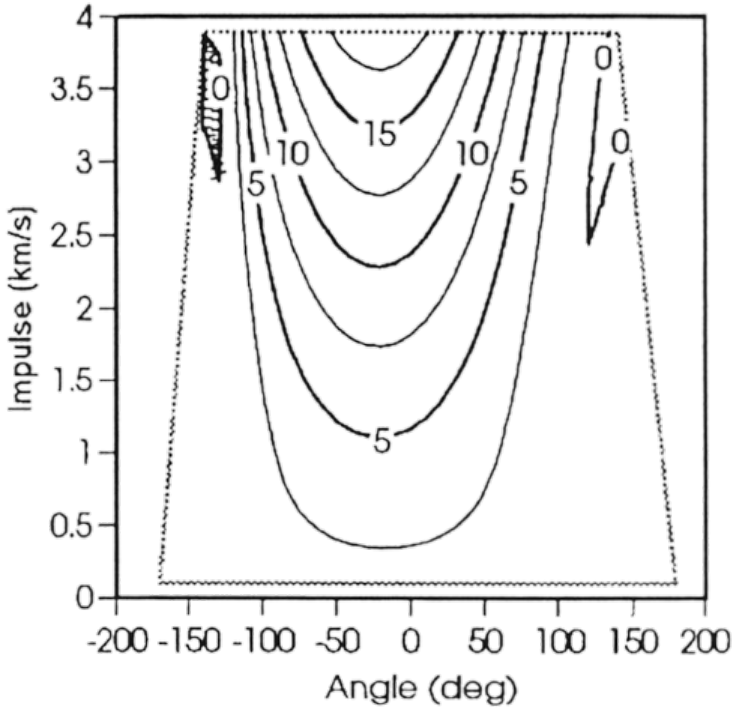

Figure 5.2: Change in Orbital Energy [12]

The algorithm generated by Prado was converted over to a Python model in order to verify future 3D variations of the formulation. Using this model with the same parameters set as before, the following graphs were generated for Δv (Fig. 5.3) and ΔE (Fig. 5.4). The capture and collision regions were not marked off since the region plotted was made smaller to focus on the region of maximum ΔV and ΔE.

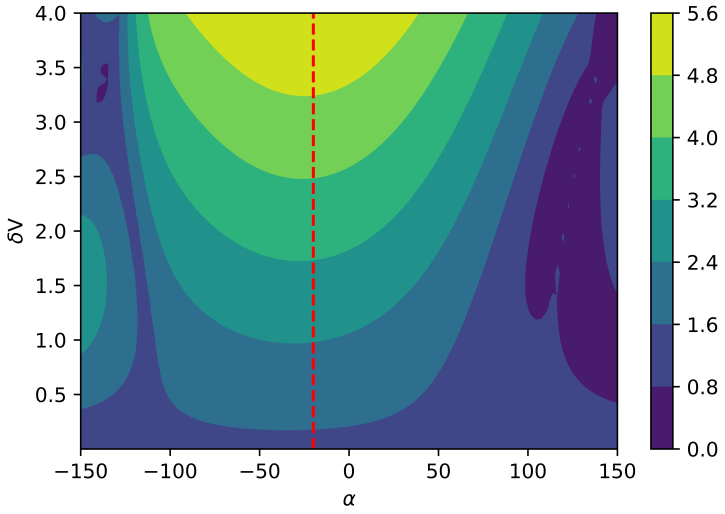

Figure 5.3: Change in Heliocentric Velocity (Prado)

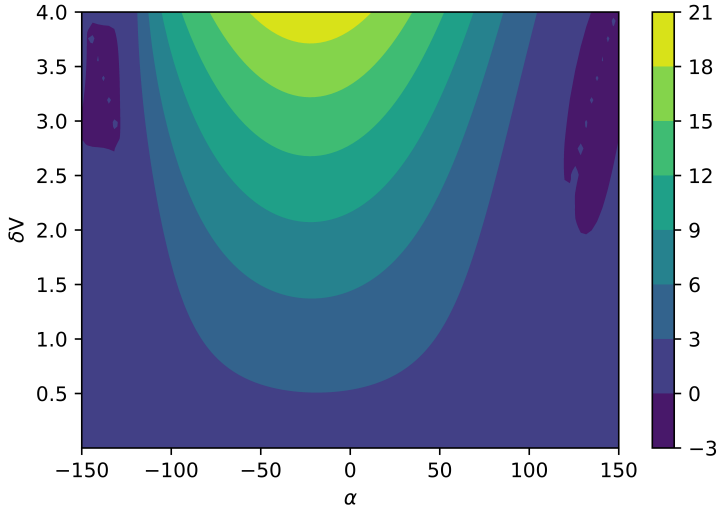

Figure 5.4: Change in Orbital Energy (Prado)

A few important behaviors can be seen as a result of Fig. 5.1 and Fig. 5.2. First, the in-plane thrust angle does not equal zero when maximum Δv for a given δv is desired. In this particular scenario, Prado noted that the angle for maximum Δv was approximately -20 degrees, denoted with the red dotted line. For this scenario, the ΔE followed a similar pattern of behavior. Finally, the spikes in Δv in the 100 degree to 150 degree range of Fig. 5.3 are a behavior of the model and orbital mechanics, not breakdowns of the numerical solution to the problem, and thus should exist in the 3D model as well.

Note that the Prado model does not use the out of plane angles as it is a 2D model. For comparison, the 3D model had those out of plane angles (i.e. β and ϕ) defined as 0. As a result, the parameters being fed into the 3D model are listed in Table 5.2:

Table 5.2: 3D Model Example Parameters

Parameter	Value	Units
μ_1	4900	km^3/s^2
μ_2	398600	km^3/s^2
V_2	1.02	km/s
$V_{\infty-}$	1	km/s
δv	0 to 4	km/s
ψ	270	deg
ϕ	0	deg
α	-150 to 150	deg
β	0	deg

With these variables set and identical to the Prado model (other than the inclusion of the out of plane angles), Fig. 5.5 and Fig. 5.6 (to be compared to Fig. 5.3 and Fig.5.4) were generated. Once again, the capture and collision regions were not marked off since the region plotted was made smaller to focus on the region of maximum ΔV and ΔE.

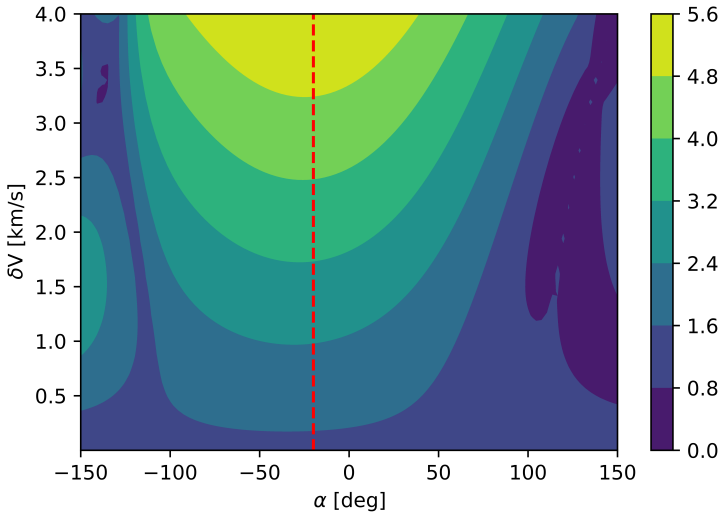

Figure 5.5: Change in Heliocentric Velocity (3D)

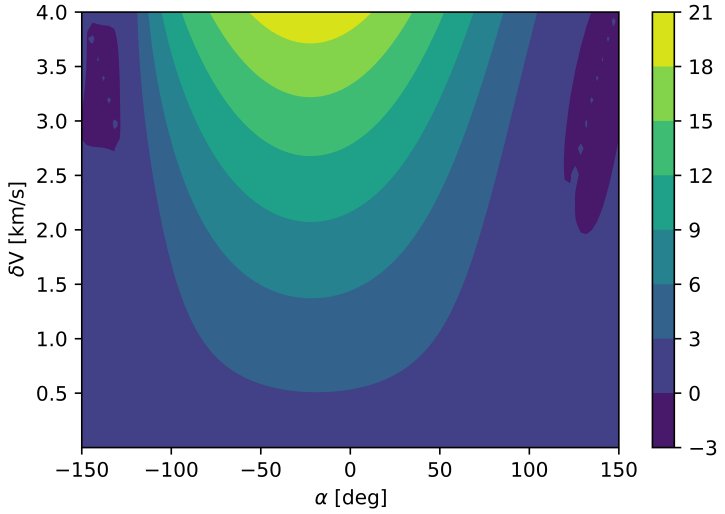

Figure 5.6: Change in Orbital Energy (3D)

The generated models were also validated against the other test case presented in the paper, in which ψ was set to 90°. The parameters tabulated above were also varied (except for the out of plane angles) to perform validation between the Prado model and the 3D model. In every case tested, the ΔV and ΔE values matched exactly.

While Prado's algorithm was entirely done with scalars, the transition to three dimensions required the utilization of multiple 3D rotation matrices and the representation of several values as vectors. The behaviors listed previously all show up in the newly generated figures. Multiple different sets of parameters were run, with the general behaviors and the values of ΔV and ΔE at each point always matching between the Prado model and the 3D model. The 3D model also was judged to have correctly implemented the rotations for the out of plane angles as modifications to these rotation matrices resulted in significant deviations/errors in comparison to the corresponding figure generated by the Prado model. Thus, the author concludes that the 3D model was properly generated and verified against the Prado model.

Chapter 6

IMPLEMENTATION

6.1 STOpS Pre-Existing Architecture

STOpS optimizes interplanetary trajectories with a Lambert's solver. For a mission that departs from a planet, performs one flyby, and arrives at another planet, STOpS will optimize three variables for an unpowered flyby. These are departure time, the time of flight to the flyby, and the time of flight to the final arrival. Increasing the number of flybys increases the number of variables that STOpS needs to optimize by one for each flyby, as it adds another flight leg. Thus, the number of variables required for STOpS to optimize unpowered flybys is equal to the number of flybys plus two, for the departure and arrival planets.

STOpS treats every flyby as a potential penalty. This penalty is calculated in a function within STOpS called *flyby_penalty*. The formulation utilized in previous STOpS variants assumes that some specific $\vec{r}_p$ will result in the outgoing v_∞ vector being matched perfectly in direction (if not magnitude), without specifically calculating this $\vec{r}_p$. This assumption is valid if the the planet is treated as a point mass and thus any radius of periapsis can be targeted to result in any turn angle. The flyby penalty then checks if the magnitudes of the incoming and outgoing v_∞ are equal. If they are not, it subtracts the two magnitudes, since the targeted radius of periapsis and turn angle are assumed to have matched up the directions perfectly. This is considered to be the δv added as the spacecraft exits the SOI required to achieve this particular flyby. Over time, STOpS will typically manipulate the variables it can adjust (i.e. the TOFs) so that this flyby penalty will be as low as possible, typ-

ically optimizing to a flyby in which the required δv is smaller than a magnitude of 10^{-5}, which is essentially a ballistic gravity assist. Due to the point mass assumption, further checks were implemented to ensure that the spacecraft does not fly through the atmosphere of the planet or collide with the planet itself. These solutions are penalized with increases in the δv related to how low the periapsis altitude is. If the flyby is within the atmosphere of a planet, it is penalized according to Eq. 6.1:

$$\delta v_{PEN} = \frac{R_{atm}}{r_p} - 1 \tag{6.1}$$

where R_{atm} is the radius of the planet's atmosphere and r_p is the spacecraft's radius of periapsis. If the flyby would collide with the planet, it is penalized according to Eq. 6.2

$$\delta v_{PEN} = \frac{3R_{body}}{r_p} \tag{6.2}$$

where R_{body} is the planetary radius. These flyby penalties were adapted by Fitzgerald from the flyby penalties described by Curtis in Orbital Mechanics for Engineering Students [2]. Within the architecture, these are meant to simulate the δv cost of doing a powered gravity assist to avoid this low flyby. Testing and comparison against the PGA algorithm has found that the magnitude of these penalties are often smaller than the required δv to perform a PGA to avoid colliding with the planet or entering the planet's atmosphere. If these penalties for radii of periapse that are too small do not occur, then the flyby penalty function could be considered to be a uPGA algorithm and the architecture could be displayed in block diagram form as in Figure 6.1. Note that, as mentioned in the Spacecraft Trajectory Optimizers chapter, each of the evolutionary algorithms runs to completion before sharing its best solution to the island model paradigm, which then distributes the best solution per its migration policies.

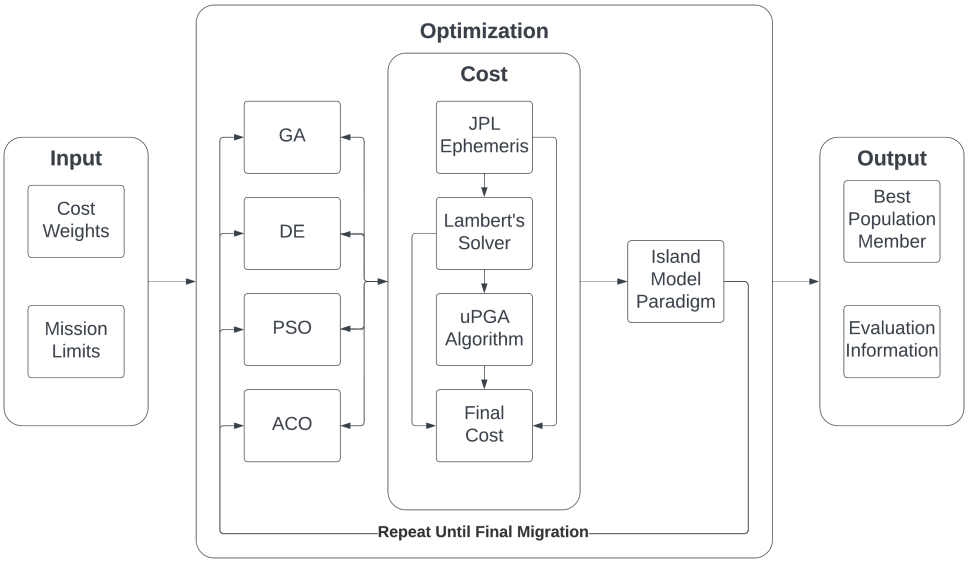

Figure 6.1: STOpS Block Diagram

6.2 Implementation of the Powered Gravity Assist Algorithm

With the addition of the PGA algorithm previously discussed, the number of variables that STOpS needs to optimize increases by six per flyby. These are, in order of storage within the STOpS-FLITE architecture: radius of periapsis of flyby (r_p), in plane r_p angle (ψ), out of plane r_p angle (ϕ), thrust applied at periapsis (δv), in plane thrust angle (α), and out of plane thrust angle (β). Thus, for the same mission with one flyby mentioned previously, the number of required variables goes from three for an unpowered flyby optimization to nine for a powered flyby.

STOpS-FLITE first optimizes the v_∞ into and out of the sphere of influence from the TOF optimization via Lambert's discussed above. The PGA algorithm will then define a trajectory using the variables discussed previously and compare the v_∞ vectors into and out of the SOI. STOpS-FLITE subtracts these vectors from each other to obtain a thrust maneuver required at the edge of the SOI when entering and when

exiting the SOI to properly fly this trajectory. In order to properly explore the search space of the potential PGA trajectories, limits are imposed, as shown in Table 6.1:

Table 6.1: PGA Algorithm Limits

Variable	Lower Limit	Upper Limit	Units
r_p	Atmosphere Radius	SOI Radius	km
ψ	-180	180	deg
ϕ	0	180	deg
δv	0	4	km/s
α	-180	180	deg
β	-90	90	deg

The angles are limited in order to properly explore any periapsis or thrust direction. The δv can be adjusted depending on spacecraft capabilities and were initialized to these values due to the magnitude of δv's from converged solutions in previous versions of STOpS. Finally, the radius of periapse has a lower limit of the atmospheric radius in order to avoid flybys through the atmosphere. The upper limit ensures the spacecraft actually enters the sphere of influence.

From this, it can be seen that the population of variables to optimize gets drastically larger as the number of flybys increase. With no flybys, STOpS only needs to optimize two variables, time of departure (TOD) and TOF from the departure planet to the arrival planet. The time of departure is stored as a Julian date, while the time of flight is stored in days. STOpS-FLITE stores these as a population vector as shown:

$$\begin{bmatrix} TOD & TOF_1 \end{bmatrix}$$

Adding even a single flyby into STOpS-FLITE adds another seven variables as there are now the six variables to define the powered gravity assist as well as a new TOF for the new leg. STOpS-FLITE stores the population vector for a single flyby trajectory

as:

$$\begin{bmatrix} TOD & TOF_1 & TOF_2 & r_{p1} & \psi_1 & \phi_1 & \delta v_1 & \alpha_1 & \beta_1 \end{bmatrix}$$

This storage system expands naturally to any number of flybys. Assuming there are n flybys, there will be $7n + 2$ variables, with the first $n + 2$ variables being TOD and TOF(s) and the remaining variables defining the PGA variables for each flyby.

With a traditional δv cost function, STOpS-FLITE will attempt to minimize the differences in incoming and outgoing v_∞ as well as the δv applied at the periapsis. Once the optimization is complete, this will result in δv's at the edge of the SOI (typically on the order of 10^{-2} km/s on entry to the SOI and as high as ≈ 1 km/s on exit from the SOI). The δv applied at periapsis can vary greatly, from 0 km/s when the PGA algorithm converges to an unpowered gravity assist to the upper limit set within the STOpS-FLITE architecture. This is heavily dependent on the mission, and these dependencies will be discussed in the following Results section. To let the mission designer minimize the δv at the edge of the SOI (as they are a result of mismatched v_∞s from the Lambert's and PGA algorithms), separate weights can be provided to STOpS for the flyby penalty δv and the δv from the applied thrust of the PGA. Both are included within the cost function for flyby penalties, but it takes in two weights: one for the flyby penalty as a whole and one specific to the PGA δv. The second is a multiplier of the first and serves as a cost modifier specifically for the thrust applied at periapsis. Thus, STOpS can penalize mismatched v_∞s more than PGA thrust applied. For example, if the departure δv has a weight of 2, the weights for the total flyby penalty and PGA δv might be 4 and 0.5. This penalizes the edge of SOI thrust maneuvers with a weight multiplier of 4, but considers the effective weight of the PGA δv to be 2, equal to that of the departure δv. In this way, the mission designer can choose whether or not only applying thrust at periapsis is an important condition to require.

One significant change from STOpS-DSM that is important to note is the exclusion of the Ant Colony Algorithm (ACA). It was previously implemented by Rockett for use with DSMs [18]. Testing with the PGA-adapted version of the ACA found that it did not improve convergence to an optimized solution for PGA trajectories but still increased the computational cost of STOpS as a whole. ACA is typically used when a problem can be considered in terms of "nodes" and "paths". ACA has great effectiveness in optimizing problems that can be considered as the best paths to reach all nodes, such as the traveling salesperson problem and the formulation of the DSM optimization problem used in STOpS-DSM. Within the STOpS-DSM formulation, the planets utilized for gravity assists are considered the nodes. Since the PGA algorithm affects only the modeling of the gravity assist at each planet, it changes the cost function at each node rather than along each path. Thus, the ACA is considered to be non-optimal for this formulation of the PGA trajectory optimization. As such, the other three optimization algorithms were considered to be capable of converging to an optimized solution without the inclusion of the ACA.

With all of these changes, the new block diagram is shown in Figure 6.2. Within this diagram, the blocks which were removed are colored in red (i.e. the ACO block), those which were modified are colored in yellow (i.e. the cost weights, mission limits, final cost, and evaluation information blocks), and the blocks which were added are colored in green (i.e. the PGA algorithm block). The cost weights, mission limits, and final cost only have small adjustments, to account for the new variables and cost multipliers with the PGA algorithm. The evaluation information was expanded in order to provide the user with more information related to the PGA thrust application and trajectory, as well as the corresponding uPGA, at the end of the optimization process. It is for this reason that the uPGA algorithm remains, in order to generate the evaluation information to output, but does not connect to or affect the final cost.

The overall structure largely remains the same, as the FLITE addition is exactly that, simply an extension of the original STOpS architecture.

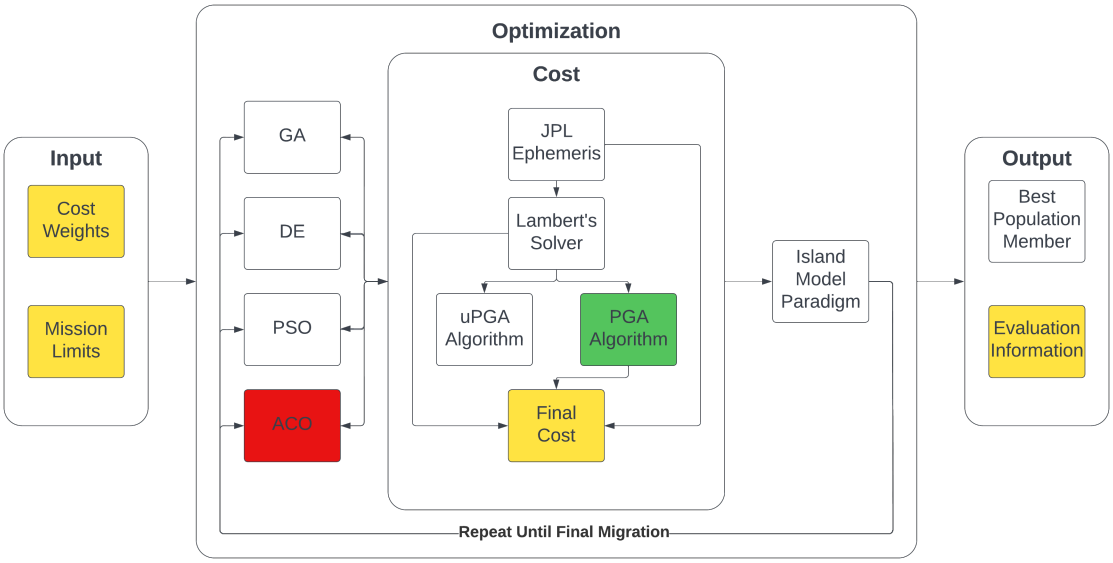

Figure 6.2: STOpS-FLITE Block Diagram

Chapter 7

RESULTS

As mentioned previously, this paper analyzes the effectiveness of gravity assists amplified by impulsive thrust engines, specifically to their functionality as a tool to decrease relevant mission parameters like time of flight or δv. In order to make this comparison, each of the following test cases will first discuss the same trajectory optimized by STOpS[1] without PGAs and then compare this control trajectory to that of STOpS-FLITE. This trajectory will be called the unpowered gravity assist (uPGA) trajectory. Each of the solutions presented is the average of at minimum five solutions to which STOpS or STOpS-FLITE converged. Due to the stochastic nature of the evolutionary algorithms used by STOpS, optimal trajectories can be slightly different from each other. For the most part, these differences are negligible. By requiring at least five solutions to average, non-optimal solutions from premature convergence are excluded. This scenario was rare, but is still possible. Typically, they can be avoided with large generation sizes (for GA/DE) or large swarm sizes (PSO), as well as increasing the number of migrations. The generation sizes were left at the standards selected by Fitzgerald and Graef, and the interested reader is referred to those works to see the reasoning behind those selections [17, 22]. The migration numbers utilized depended on the complexity of the problem. For single flyby missions, there are nine variables to optimize. Typically, after ten migrations, the cost function changes by less than 1%. Increasing the number of migrations does not significantly change the converged solution or the cost function value. Adding another flyby adds another

[1]Throughout this segment, STOpS will refer to a variant of STOpS without PGAs included, to differentiate from STOpS-FLITE. Unless otherwise noted, this will be STOpS-DSM with DSMs turned off, as that was the version on which STOpS-FLITE was built.

seven variables to optimize, thus requiring more migrations. Testing until the cost function changes by less than 1% provides a reasonable number of migrations for any specific trajectory. Finally, after a rough estimate of the most optimal time to perform the flybys are known, the window narrowing mentioned in the Implementation chapter can be performed. Through window narrowing, sufficient migrations, and the averaging of multiple solutions, converged solutions can be reasonably considered to be the optimal solution.

7.1 Test Scenario 1: Earth-Jupiter-Saturn (EJS) Trajectory

7.1.1 Description

The first test case to be considered was a single flyby mission departing Earth, performing a powered gravity assist at Jupiter, and arriving at Saturn. This mission is meant to be similar to the Voyager missions in purpose, which is to say that capture at Saturn is not a targeted condition. The Voyager missions themselves were not utilized due to their complexity and their use of DSMs, which would necessitate DSM modeling to be accurately modeled. This scenario is simply meant to show the effectiveness of the PGA at either reducing TOF or δv. With these objectives stated, the cost functions considered for this scenario are simply departure δv, flyby penalty, flyby δv, and TOF. These cost functions are shown with their weights in Table 7.1:

Table 7.1: EJS Trajectory Cost Functions

Cost Function	Weight	Units
Departure δv	1	km/s
Flyby Penalty (PGA δv)	2 (0.5)	km/s
TOF	0.01	days

Note that the weight in the parentheses of the flyby penalty cost function is the cost modifier described in the Implementation chapter. These weights resulted in thrust maneuvers at the edge of the SOI being relatively small (i.e.$<$ 0.2 km/s, in this case) compared to the departure and PGA δv's. The TOF weighting is low in order to scale the TOF cost function so as to not dominate the cost function. In this scenario, the TOFs of each leg are on the order of magnitude of hundreds of days. By multiplying this TOF value by 0.01, it is on the same order of magnitude as the δv cost functions ($\approx 10^0$ to 10^1). The chosen limits for each member of the population are shown in Table 7.2. Note that when STOpS is used as opposed to STOpS-FLITE, the population limits are solely the first three rows. STOpS was adjusted to have the flyby penalty cost set to infinity if the spacecraft had to fly through the atmosphere or flew into the planet. This was because in previous versions of STOpS, flyby penalties for these scenarios were meant to simulate the cost of a PGA to avoid these scenarios. If a spacecraft were to fly through the atmosphere or into the planet, the resulting damage would be considered to cause mission failure, thus the decision to set flyby penalties to infinity in these cases to force STOpS to only consider uPGA trajectories. While all STOpS variants store the starting date limits as Julian dates and angles as radians, they have been converted to calendar dates and degrees, respectively, in the following tables for readability.

Table 7.2: EJS Trajectory Limits

Parameter	Lower Limit	Upper Limit	Units
TOD	09/01/1977	9/10/1977	Date
TOF_1	450	650	days
TOF_2	700	750	days
r_p	69911	48200000	km
ψ	-180	180	deg
ϕ	-90	90	deg
δv	0	5	km/s
α	-180	180	deg
β	-90	90	*deg*

7.1.2 Control Case

First, STOpS is used to model this trajectory with PGAs turned off. This result is the best case scenario converged to by STOpS using only ballistic gravity assists and thrust maneuvers at the edge of the SOI. The results of five of these solutions is tabulated in Table 7.3, along with the δv cost function values and total cost, with the trajectory itself plotted in Fig. 7.1.

Table 7.3: EJS uPGA Results

Parameter	Lowest	Highest	Average	Units
TOD	09/02/1977	09/03/1977	09/03/1977	Date
TOF_1	648.123	648.366	648.220	days
TOF_2	734.185	734.244	734.207	days
δv_{Launch}	9.664	9.673	9.665	km/s
$\delta v_{FlybyPenalty}$	0	0	0	km/s
$TotalCost$	22.917	23.225	23.469	N/A

Note that these values are expressed to the third digit after the decimal point so that comparisons between small differences between solutions which converge to a small window can be made. In reality, some of these values may be more accurate a spacecraft is capable of targeting or accomplishing, especially when it comes to pointing the spacecraft at the correct angles for thrust at periapse, the burning the correct amount of fuel to achieve the δv, or targeting the exact position of the radius of periapse.

The plots of the interplanetary trajectory generated by STOpS are in the ICRS frame. Figures are generated looking down on the X-Y plane of that frame, which leads to the circular orbits of the planets occasionally appearing elliptical and/or off center from the origin.

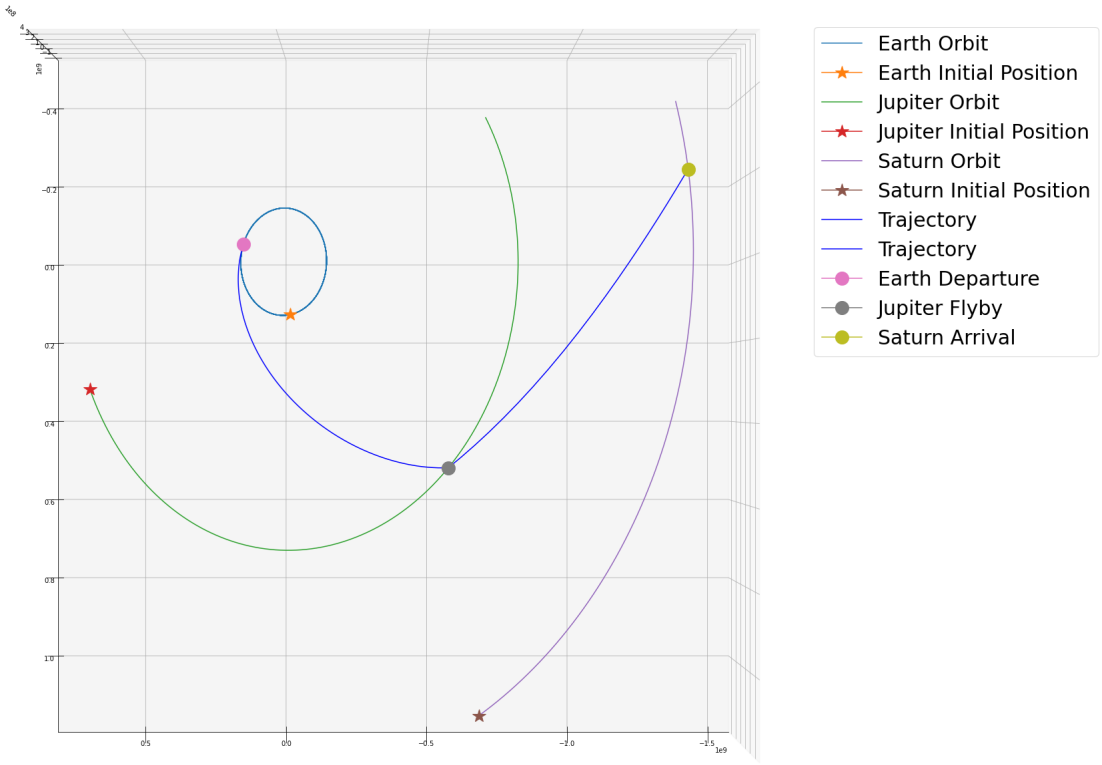

Figure 7.1: Earth-Jupiter-Saturn uPGA Trajectory

This trajectory takes a total of 1382.4 days, departing September 2nd, 1977 and arriving almost exactly between June 15th and June 16th, 1981 and using a total of 9.665 km/s of δv. Note that the final cost is a combination of both δv cost and time of flight cost, thus it has no units. This sets the baseline for comparison to the powered gravity assists results.

7.1.3 PGA Case

With this baseline uPGA trajectory set, STOpS-FLITE is now run with the parameters listed above. In order to keep the tables comparable to the uPGA table, the flyby penalties listed for the PGA cases are the δv's applied at the edge of the SOI. The δv applied at periapsis is still included in the total cost function. The parameters

defining this solution set are defined in Table 7.4, along with relevant cost functions and trajectory details:

Table 7.4: EJS PGA Results

Parameter	Lowest	Highest	Average	Units
TOD	09/06/1977	09/07/1977	09/07/1977	Date
TOF_1	495.268	498.409	496.567	days
TOF_2	699.297	713.632	704.407	days
r_p	600151	607430	602126	km
ψ	167.361	167.590	167.476	deg
ϕ	7.974	8.160	8.058	deg
δv	1.450	1.593	1.515	km/s
α	179.832	180.000	179.914	deg
β	0.211	4.810	1.875	deg
$\delta \mathrm{v}_{Launch}$	10.836	10.873	10.857	km/s
$\delta \mathrm{v}_{FlybyPenalty}$	0.053	0.259	0.159	km/s
Cost	24.662	24.781	24.700	N/A

The PGA trajectory utilizes a total of 1.674 km/s of δv in flight after launch in exchange for a TOF decrease of nearly 6 months compared to the uPGA trajectory. Whether or not this is an acceptable trade is up to the mission designer to choose. Good cost weights can help the mission designer let STOpS converge to the optimal solution for their particular considerations, but without prior knowledge, tests can be run using default weights and adjusted accordingly. The PGA trajectory is shown in Fig. 7.2, while the flyby itself is plotted in Fig. 7.3.

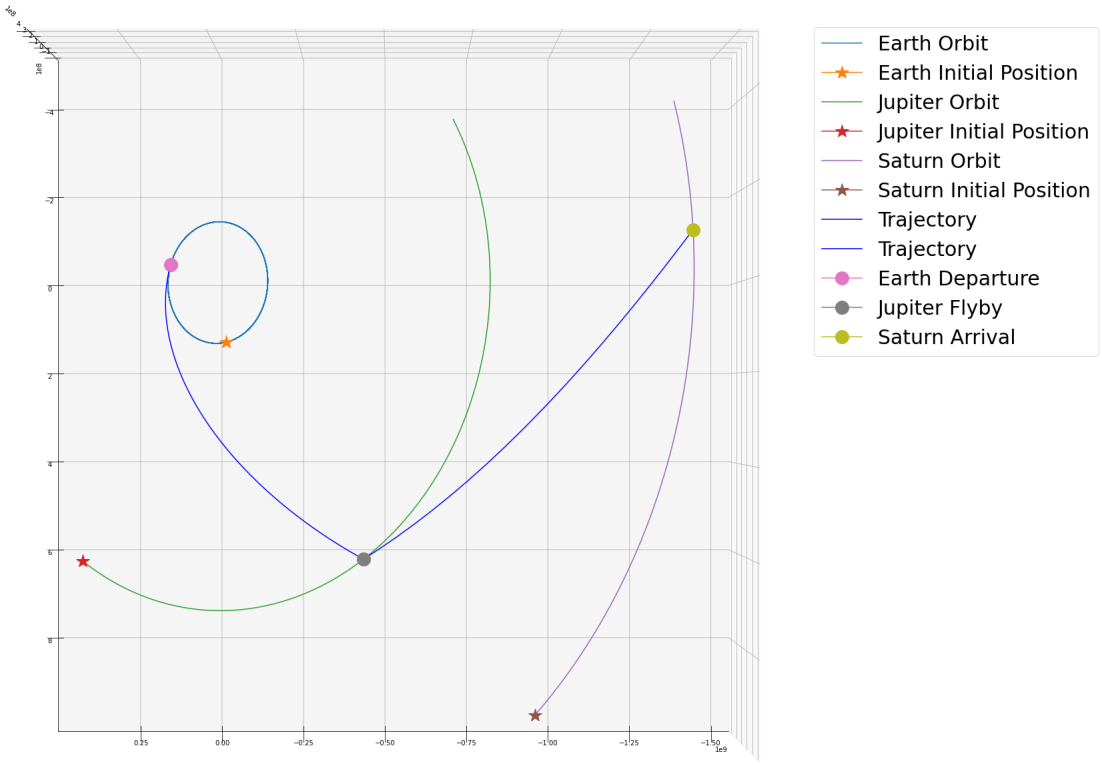

Figure 7.2: Earth-Jupiter-Saturn PGA Trajectory

The low β angle means that there is very little inclination change due to the δv applied at periapse. The thrust is mostly pointed in the direction opposite the velocity at periapse, effectively slowing the spacecraft at periapse. In doing so, the spacecraft changes its post-thrust turn angle and its speed. Since the trajectory is propagated the same amount of time backwards from periapse for the pre-thrust segment and forwards from periapse for the post-thrust segment (for the purposes of plotting), this slower speed is evidenced by the shorter post-thrust segment.

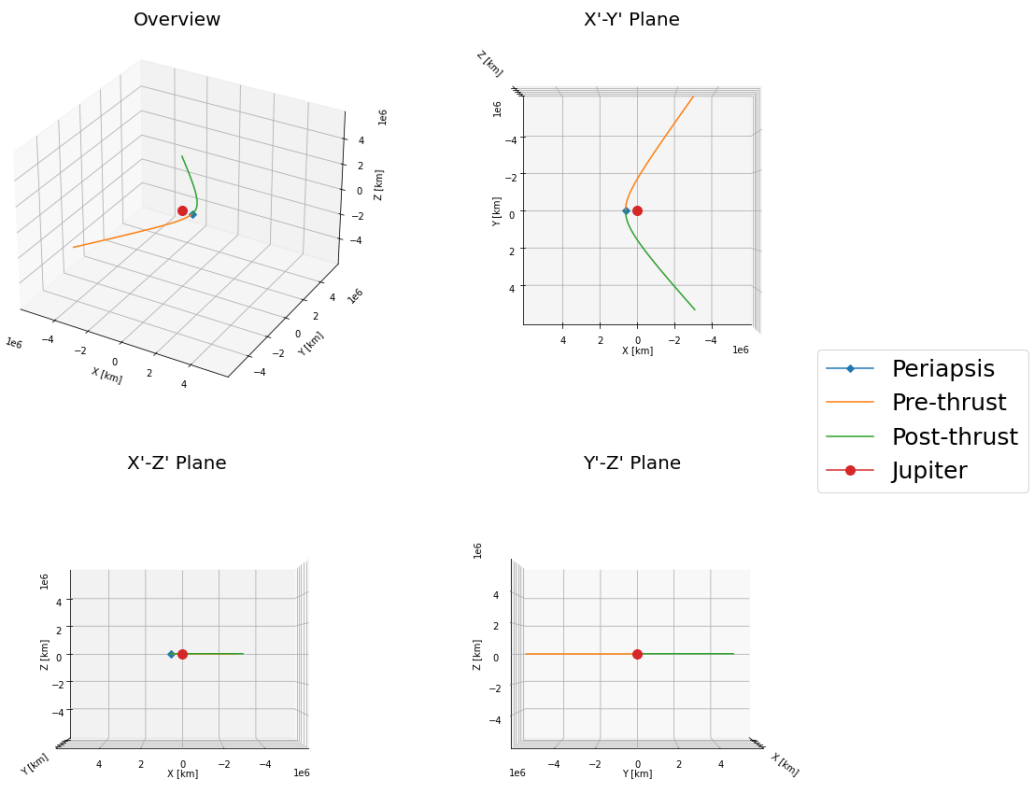

Figure 7.3: EJS Jovian Flyby (OP_A Frame)

Note that this plot (and all of the flyby plots within this paper) is meant to show the flyby maneuver near the planet, so it is zoomed in on the area around the planet, rather than plotting to the edge of the SOI. If the trajectory was propagated to the edge of the SOI, the plot would simply appear as two line segments, and the hyperbolic nature of the flyby would be difficult to see as the curved segment would be much smaller than the much larger, nearly linear segments of the hyperbola as it approaches the asymptote. The marker for the planet is meant to show the location of the center of the reference frame, and the size of the marker is not scaled to the size of the planet. Furthermore, the marker is placed on top of the plotted path in each subplot regardless of if the spacecraft travels behind or in front of the planet.

The time of departure shifts forward four days, and due to the significantly decreased time of flight to Jupiter, the arrival date at Jupiter is earlier than in the uPGA case. This is the majority of the decrease in TOF and is accounted for by the higher departure δv. If the launch vehicle is still capable of applying this δv, this increase is not particularly important. The second leg accounts for about 30 days of the total decrease in TOF. Nevertheless, the PGA allows the spacecraft to target this trajectory when previously it was incapable of doing so. Thus, even though there is only a small decrease in TOF after the PGA, the total TOF decrease is still partially attributable to the execution of a PGA. The PGA algorithm also converges to an α angle of almost exactly 180 degrees. This means the spacecraft is thrusting directly opposing the direction of the periapsis velocity. This sharpens the angle of the flyby significantly, which is visually apparent in Fig. 7.2 compared to Fig. 7.1. There are values of the out of plane angles ϕ and β close to, but not exactly equal to 0. These are likely a result of the differences in inclination between the orbits of Earth, Jupiter, and Saturn. Slightly larger thrust maneuvers for proper targeting are also required at the edge of the SOI in the PGA case. Finally, the radius of periapsis is the same before and after the thrust. This is because the thrust is directed exactly opposing the periapsis velocity, which means that the periapsis location does not shift (i.e. $f_0 \approx 0$), but the periapsis velocity does. The smaller periapsis velocity with the same periapsis radius leads to a sharper turn angle as the eccentricity of the hyperbola decreases.

Table 7.5: EJS Trajectory Detail Comparison

Parameter	uPGA	PGA	Units
Total TOF	1382.427	1200.974	days
Equivalent uPGA Periapse	600000	466901	km
Lowest Periapse	600000	602126	km
Thrust Maneuver Altitude	N/A	602126	km
In Flight δv	0	1.674	km/s
Total δv	9.665	12.531	km/s
Arrival v_∞	11.788	13.473	km/s

As seen here, there are trades between TOF and in flight δv across these three cases. The uPGA has lower in flight δv (as well as lower total δv, as it has a lower launch δv) required, but has a higher TOF. Meanwhile, the PGA uses more δv and decreases the TOF by about 6 months. Another interesting point to note is how close each of the trajectories of the Jovian atmospheric radius of 600000 km. With an uPGA, the optimal trajectory skims the Jovian atmosphere at periapsis, while the PGA trajectory could provide some tolerance for errors in the δv provided at launch or at the entrance to the SOI. the PGA trajectory provides over 2000 km of error tolerance at periapsis, and because the thrust is directly opposing the velocity at periapsis, it does not come any closer to the planet. Convergence to thrust angles that are not exactly 180 degrees also suggest some robustness against thrust pointing errors. Currently, no version of STOpS models spacecraft attitude dynamics or account for potential errors in thrust application or direction, so this was not able to be included as a relevant cost function. Finally, the arrival v_∞ at Saturn is lowest for the uPGA case. This makes sense as the PGA trajectory is using more δv overall than the uPGA case. This extra δv leads to extra heliocentric velocity, leading to a greater v_∞ at Saturn, which could be beneficial in leaving the Solar System, if a

Voyager-like mission was desired[2]. Further, the capability to perform a PGA (such as burning any remaining fuel) at Saturn to increase this heliocentric energy even further would provide an even better maximum heliocentric energy for this mission than an equivalent uPGA. Also of note is the fact that the PGA case has only a slightly higher total cost than the uPGA scenario. This suggests that with these weightings, the two missions are considered almost equally optimal. Further, the uPGA case considers a highly idealized gravity assist, while the PGA algorithm provides an actual radius of periapse to target.

7.1.4 Missed Launch Window

Assuming the maximum flight time for this spacecraft is the 1450 days that represent the sum of the maximum limit of $TOF1$ and $TOF2$, a scenario in which the early September launch window is missed is considered. In order to do this, the end date for launch consideration was initially shifted to December 31$^{\text{st}}$, 1977 to open up options to STOpS without allowing for a launch in the next full rotation of the planets. Next, the start date for launch consideration was shifted towards the end of the year until STOpS was unable to converge to a solution. STOpS was able to converge to increasingly δv heavy solutions until October 13$^{\text{th}}$, 1977, at which point all trajectories would have required the spacecraft fly through the Jovian atmosphere in order to stay under the maximum flight time limit. This was considered to be a failure state for the mission (i.e. flying through the atmosphere results in the flyby penalty being set to infinity), resulting in a failure for STOpS to converge. Meanwhile, STOpS-FLITE was able to converge to a solution defined by the variables displayed in Table 7.6:

[2]While a heliocentric energy cost function could be utilized, PGAs introduce such a wide variety of potential heliocentric energies post-PGA that they span multiple orders of magnitude, making them difficult to scale properly to not be dominated/dominate other cost functions.

Table 7.6: EJS Missed Launch Window PGA

Parameter	Lowest	Highest	Average	Units
TOD	10/14/1977	10/14/1977	10/14/1977	Date
TOF_1	471.411	473.400	472.321	days
TOF_2	709.407	715.055	712.223	days
r_p	600270	602626	601177	km
ψ	167.194	167.420	167.293	deg
ϕ	8.375	8.766	8.535	deg
δv	1.486	1.535	1.505	km/s
α	179.880	180.000	179.969	deg
β	0.211	5.446	3.143	deg
δv_{Launch}	15.660	15.677	15.670	km/s
$\delta v_{FlybyPenalty}$	0.057	0.133	0.098	km/s
Cost	29.188	29.246	29.217	N/A

In comparison to the regular launch date trajectory, the missed launch date trajectory is visually very similar. The approach angles ψ and ϕ are within 2 degrees, as are the thrust pointing angles. The biggest change is the much shorter TOF_1, which is a result of the much higher δv at launch. This results in a lower total TOF, though the 50% greater launch δv is considered to be a less optimal solution, as shown by the significantly higher cost function value. Nevertheless, this is an improvement over STOpS being completely unable to find a trajectory that does not fly through the Jovian atmosphere after October 13$^{\text{th}}$, 1977. Here, the margin between the periapse radius of the PGA is cut in half compared to the regular launch date trajectory, but it remains at a good value to provide some factor of safety against targeting and maneuver errors. The in flight δv value of 1.598 km/s is lower than that of the regular launch date trajectory.

In summary, STOpS-FLITE is capable of converging to optimal solutions within the problem space of a single PGA trajectory. Specifically, the PGA algorithm grants the mission designer an option between a trajectory with lower TOF and a higher δv requirement or vice versa. By modifying the cost weights, a mission designer can

choose how important preserving fuel is compared to reducing time of flight. In the event of a missed launch window, STOpS-FLITE is still able to converge to a solution where STOpS was unable to do so. Thus, PGAs prove effective as a tool to increase robustness and provide opportunities where none would have otherwise existed.

7.2 Test Scenario 2: Earth-Mars-Venus-Mercury (EMVM) Trajectory

7.2.1 Description

To consider the possibilities of the multiple PGA (MPGA) trajectory, a mission was considered involving an Earth departure in the mid-2040s followed by PGAs at Mars and Venus before arriving at Mercury. The scenario was first optimized using STOpS-FLITE, before being run in STOpS to consider the equivalent uPGA trajectory. The uPGA trajectory proved to be largely infeasible due to massive required δv's to avoid flybys through the Martian and Venusian atmospheres or through the planets themselves. In addition, the uPGA trajectory converged to by STOpS resulted in a significantly larger TOFs. As such, this trajectory is also considered to be proof of the hypothesis that the usage of PGAs creates possibilities for trajectories when traditional uPGA trajectories are simply impossible.

The cost functions for this trajectory were adjusted to penalize δv less, as the windows for TOF are now larger. This means that extremely high TOFs with extremely low δv were dominating the solution set when using the previously used cost weighting. As a result, the δv-related cost functions were lowered until more consistent TOFs were achieved with slightly larger δv's. These cost functions and weights are described in Table 7.7:

Table 7.7: EMVM Trajectory Cost Functions

Cost Function	Weight	Units
Departure δv	1	km/s
Flyby Penalty (PGA δv)	2 (0.5)	km/s
TOF	0.01	days

The limits for the uPGA and PGA trajectories were slightly different from each other, as the PGA trajectory converged to a relatively small TOF for the first leg of the trajectory that the uPGA trajectory was unable to match. STOpS was unable to converge to a uPGA trajectory at all within this TOF window without flying through either the atmosphere or the body of one or both flyby planets. As a result, larger windows for each of the TOF-related variables were considered by STOpS to allow convergence. This is tabulated in Tables 7.8 and 7.9. Note that the limits for the angles describing the PGA as well as the δv applied at periapsis are the same for both the Martian and Venusian PGA.

Table 7.8: EMVM PGA Trajectory Limits

Parameter	PGA Lower Limit	PGA Upper Limit	Units
TOD	01/01/2043	12/31/2043	Date
TOF_1	210	250	days
TOF_2	230	270	days
TOF_3	60	100	days
r_{p1}	3390	577000	km
r_{p2}	69911	48200000	km
ψ	-180	180	deg
ϕ	-90	90	deg
δv	0	5	km/s
α	-180	180	deg
β	-90	90	*deg*

Table 7.9: EMVM uPGA Trajectory Limits

Parameter	Lower Limit	Upper Limit	Units
TOD	01/01/2043	12/31/2043	Date
TOF_1	210	750	days
TOF_2	50	300	days
TOF_3	20	200	days

7.2.2 Control Case

For this scenario, the uPGA trajectory was considered after the PGA trajectory. This was done in order to show some of the trajectories converged to by STOpS-FLITE are impossible for STOpS to converge to without entering planetary atmospheres or colliding with planetary bodies. In these conditions, the cost function of the flyby penalty was set to infinity. In this case, this was shown to be true, with STOpS unable to converge to a uPGA trajectory in the original $TOF1$ window given. In order to have something to compare to, the $TOF1$ window was widened until STOpS was able to converge to a solution. This solution, while valid from an orbital mechanics perspective, requires incredibly high amounts of δv and higher TOFs than the equivalent PGA trajectory. The parameters for this solution are listed in Table 7.10:

Table 7.10: EMVM uPGA Results

Parameter	Lowest	Highest	Average	Units
TOD	11/21/2043	11/22/2043	11/21/2043	Date
TOF_1	749.871	750.000	749.997	days
TOF_2	86.215	87.030	86.980	days
TOF_3	47.112	47.598	47.201	days
δv_{Launch}	30.225	30.782	30.356	km/s
$\delta v_{Flyby1Penalty}$	7.008	7.231	7.200	km/s
$\delta v_{Flyby2Penalty}$	12.442	12.824	12.699	km/s
Total Cost	76.741	78.660	77.010	N/A

This set of parameters was clearly still incredibly costly, as evidenced by the high cost function. Plotting the trajectory (in Fig. 7.4) also shows that this trajectory sends the spacecraft far outside of Earth's orbit due to the incredibly high $TOF1$.

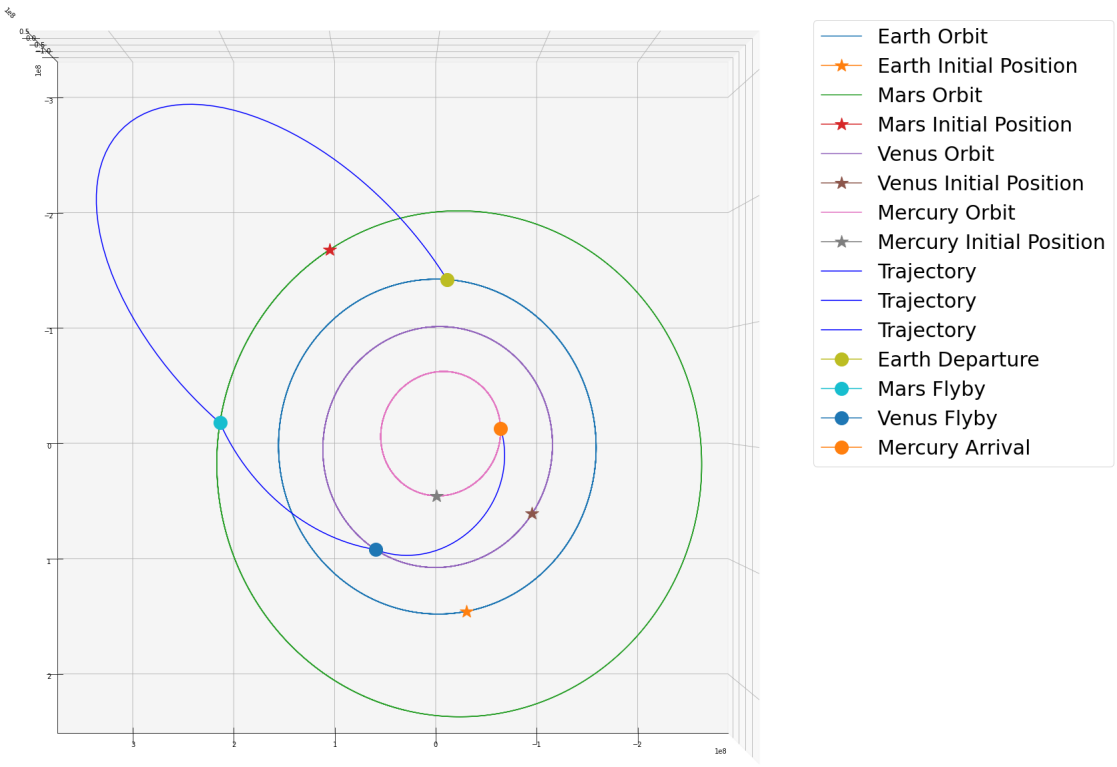

Figure 7.4: Earth-Mars-Venus-Mercury uPGA Trajectory

In fact, this $TOF1$ was the edge of the limits for $TOF1$, but increasing it further would simply allow it to deviate from the comparison PGA trajectory even further, while smaller values of $TOF1$ result in flybys through the Martian atmosphere. Even with the relatively small atmospheric radii of the inner planets (compared with the Jovian atmosphere used in the previous test scenario), STOpS was unable to converge to a trajectory that had a lower cost function than the PGA case. Specifically, the values for each of the δv cost functions were abnormally high, requiring high launch δv and high corrective δv's to match up v_∞'s properly. Even if this trajectory is valid from an orbital mechanics standpoint, it is likely impossible to perform given

current propulsion system technology. Without a PGA, this mission either need to shift its launch window to another year or have the capability to perform these large δv maneuvers.

7.2.3 PGA Case

With PGAs, the mission trajectory converged to a solution with much lower δv's and TOF. First, consider the same variables as displayed previously in the uPGA results (namely all of those that are not directly related to the PGA application) as shown in Table 7.11:

Table 7.11: EMVM PGA Results

Parameter	Lowest	Highest	Average	Units
TOD	12/09/2043	12/24/2043	12/18/2043	Date
TOF_1	230.598	240.543	237.696	days
TOF_2	247.876	254.464	251.488	days
TOF_3	67.958	72.139	69.881	days
δv_{Launch}	3.388	4.332	3.874	km/s
$\delta v_{Flyby1Penalty}$	0.991	1.716	1.309	km/s
$\delta v_{Flyby2Penalty}$	1.825	2.548	2.047	km/s
Total Cost	20.483	21.357	20.968	N/A

Note the much lower total cost, a result of the significantly lower δv penalties and launch δv. In particular, the launch δv is an order of magnitude smaller, and the total flyby penalties are a third of the those of the uPGA case. Though $TOF2$ and $TOF3$ are larger than in the uPGA case, the much smaller $TOF1$ makes up for them, leading to a net decrease in TOF relative to the uPGA case. Even with the PGA δv's, the total δv of this trajectory is lower than the total δv of the uPGA trajectory. Next, consider the variables relating to the actual application of the PGAs in the Martian and Venusian SOIs in Table 7.12:

Table 7.12: EMVM PGA Application

Parameter	Lowest	Highest	Average	Units
r_{p1}	413314	552461	490335	km
ψ_1	-27.101	-20.627	-24.465	deg
ϕ_1	0.859	2.922	2.234	deg
δv_1	4.233	4.805	4.564	km/s
α_1	-5.157	5.672	-0.516	deg
β_1	13.694	35.982	21.715	deg
r_{p2}	6390	6463	6433	km
ψ_2	-129.603	-117.514	-124.217	deg
ϕ_2	80.100	89.381	84.855	deg
δv_2	0.087	0.527	0.227	km/s
α_2	13.579	180.000	143.526	deg
β_2	43.316	87.090	64.802	deg

Several things are important to note from Table 7.12. Despite the wide range of periapse radii for the Martian flyby, the ψ and ϕ angles are very similar, suggesting that the optimal approach to this flyby is more dependent on approach angle than flyby radius. This is likely because the very large δv applied during this maneuver, combined with the relatively small gravitational parameter of Mars, means that most of the work of this PGA is done by the applied δv, rather than the gravity assist itself. This δv value of 4.735 km/s is close to the δv limiter, but STOpS-FLITE did not converge to a value of exactly 5 km/s, thus this limiter value is not considered to be hindering the convergence to an even more optimal (i.e. lower cost) solution. Meanwhile, in the Venusian flyby, the range of the radii of periapsis is very small. This flyby similarly has consistent approach angles, but a widely ranging δv_2, α_2, and β_2. This suggests that this flyby has the opposite relationship from the Martian flyby, in that most of the work is done by the gravity assist rather than the δv applied. In fact, when the pre-thrust and post-thrust periapses during the Venusian flyby are considered, they never change by more than 0.01 km, with the periapsis position shift always less than 0.1 degrees. This is likely because the pre-thrust periapsis velocity is

already incredibly high due to the close flyby that the small thrust applied in plane is indistinguishable from no thrust at all. This also explains the wide range of α_2, as the applied δv affects the periapsis shift or post-thrust periapsis very little regardless of which direction the thrust is applied. However, the more consistent β_2 angle suggests that the δv applied is being used to assist in the inclination change to target Venus, but even then the range is much wider than it was for β_1. Ultimately, since the final cost functions are in a very small window, the wider range of converged values for the second PGA suggest that it is heavily affected by even small changes to TOF and approach angle from the previous PGA, but ultimately result in a roughly equally optimal solution. Finally, there are flyby penalties greater than 1 km/s at both flybys. This suggests that there is some amount of the change in v_∞ at both flybys that is more efficient to perform at the edge of the SOI than at periapse, due to the nature of this specific trajectory. This trajectory is plotted in Fig. 7.5:

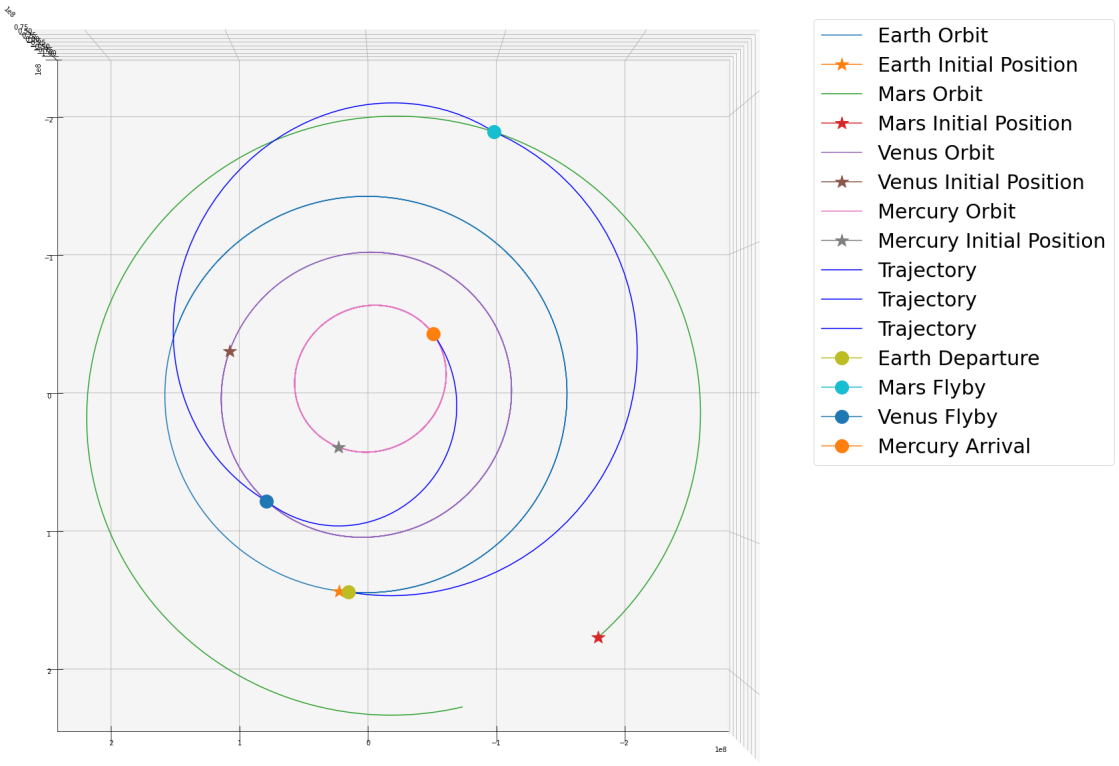

Figure 7.5: Earth-Mars-Venus-Mercury PGA Trajectory

Though difficult to see from Fig. 7.5, there is an inclination difference between the orbits of Mercury and Venus of 3.6 degrees, which is the reason for the high ϕ value at the Venusian flyby. The β angle at the Venusian flyby is also large, but as mentioned previously, the spacecraft is flying so fast at periapse in this flyby that the thrust applied is negligible in comparison. Compare this to the Martian flyby, which needs to account for the inclination difference between Venus and Mars (1.5 degrees). Here, the ϕ angle is under 5 degrees, but the majority of the work of this PGA is done by the δv applied. This explains the high β angle, as the inclination change is more a result of the δv of the PGA rather than the gravity assist itself. These inclination changes are more visible in the plots of the flybys themselves, as shown in Figs. 7.6 and 7.7. Specifically, the Y'-Z' plane subplots show how much greater the inclination change from the thrust applied is at the Martian flyby than at the Venusian flyby.

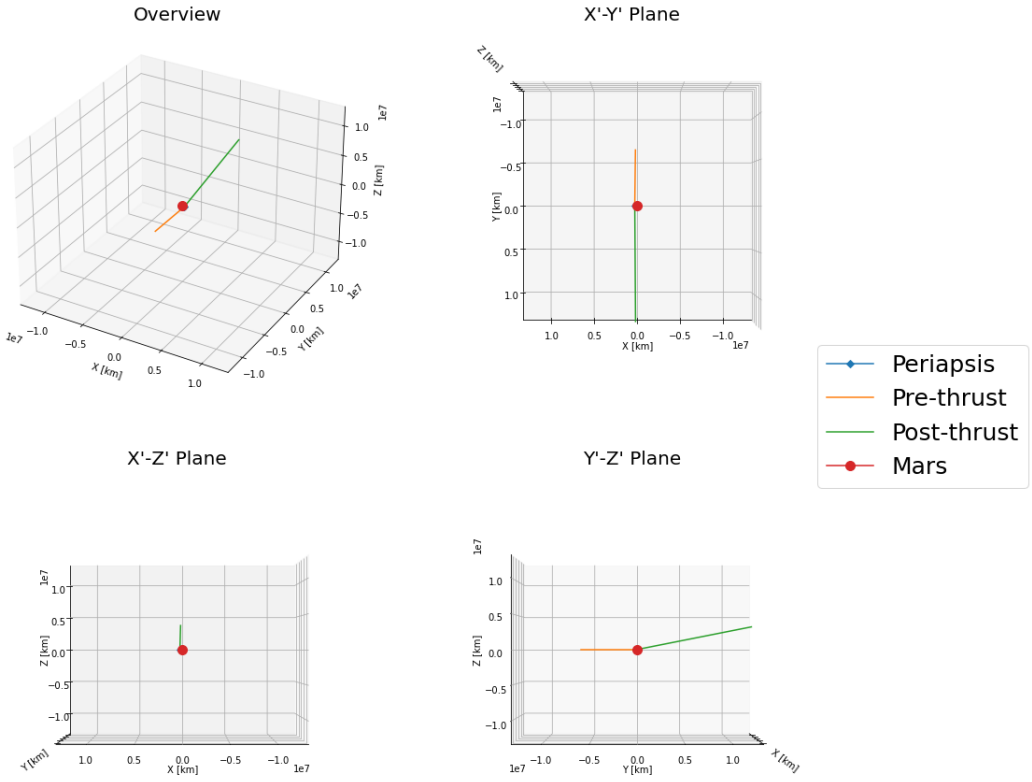

Figure 7.6: EMVM Martian Flyby (OP_A Frame)

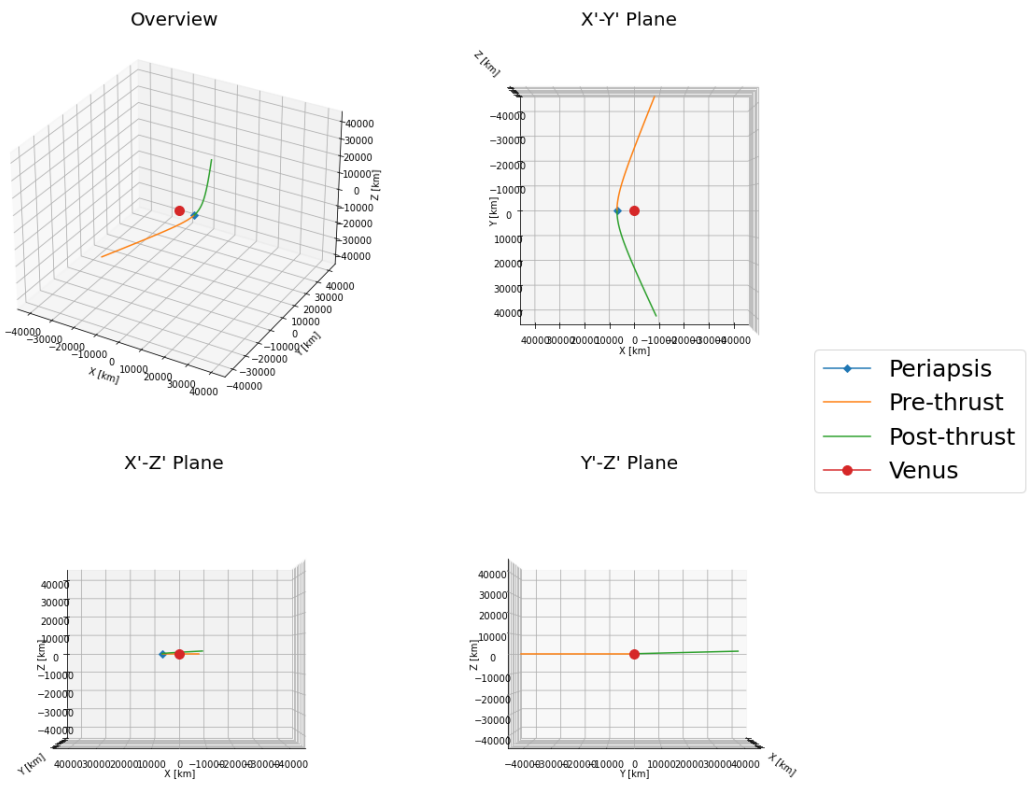

Figure 7.7: EMVM Venusian Flyby (OP_A Frame)

Note the much sharper angle of the Martian flyby due to the much greater δv applied at periapsis, though a small inclination change is visible in the Venusian flyby. Because these plots are in the OP_A frame, the inclination change shown in the Y'-Z' plane subplot is only the inclination change from δv applied at periapsis. In generating graphs, the pre-thrust and post-thrust trajectories were propagated backwards and forwards, respectively, for the same duration. The Martian flyby adds a significant δv at periapse, which is evident in the much larger post-thrust travel distance with the same propagation time. The high δv applied at periapse and the high flyby radius (which means the periapse velocity is relatively low) lead to a larger relative change in δv from this maneuver than in the Venusian flyby. The high flyby radius also results in a very small turn angle, which explains how straight the

trajectory appears in most of the subplots. At the Venusian flyby, the spacecraft is traveling at a much higher velocity, thus even a thrust maneuver of equal magnitude would result in a smaller percentage change in the δv at periapsis, and thus a smaller change visually in the trajectory. Specifically, the Martian flyby has a pre-thrust periapse velocity of approximately 3.76 km/s, meaning the δv applied at periapse of 4.56 km/s is larger than the pre-thrust periapse velocity. Meanwhile, the Venusian flyby has a pre-thrust periapse velocity of approximately 13.09 km/s, meaning the δv applied at periapse of 0.23 km/s is two orders of magnitudes smaller than the pre-thrust periapse velocity. The relatively small δv at periapse in the Venusian flyby results in a largely symmetric in plane trajectory, with a small inclination change accomplished by the δv.

In summary, this scenario is evidence that the PGA algorithm implemented in STOpS has the ability to converge to solutions where either the majority of the δv of the PGA is performed at periapsis or the majority of the δv is performed outside the SOI. This is confirmation of the hypothesis drawn from Qi and de Ruiter's work which suggested that, at times, a combination of thrust maneuvers inside and outside of the SOI would be superior to maneuver a single thrust maneuver. This scenario also demonstrates that STOpS-FLITE is capable of converging to low cost MPGA trajectories where STOpS either finds a uPGA trajectory with higher cost or is simply unable to converge to a solution at all. Thus, PGA modeling is demonstrably superior to uPGA modeling in some cases, especially when specific start dates or TOFs are desired.

7.3 Test Scenario 3: Mariner 10 (Earth-Venus-Mercury)

Lastly, a scenario in which STOpS-FLITE should converge to either a uPGA or a very small PGA is considered. For this, Mariner 10's mission from Earth to Mercury via a Venusian flyby was selected. While this mission did utilize 3 TCMs, it did not have a PGA. No control case was run with STOpS as ultimately, it should provide the same answer. The expected result for this trajectory is for STOpS-FLITE to converge to a δv at periapsis of close to 0 with minimal flyby penalties. The final trajectory should be visually similar to the trajectory actually flown by Mariner 10, shown in Fig. 7.8:

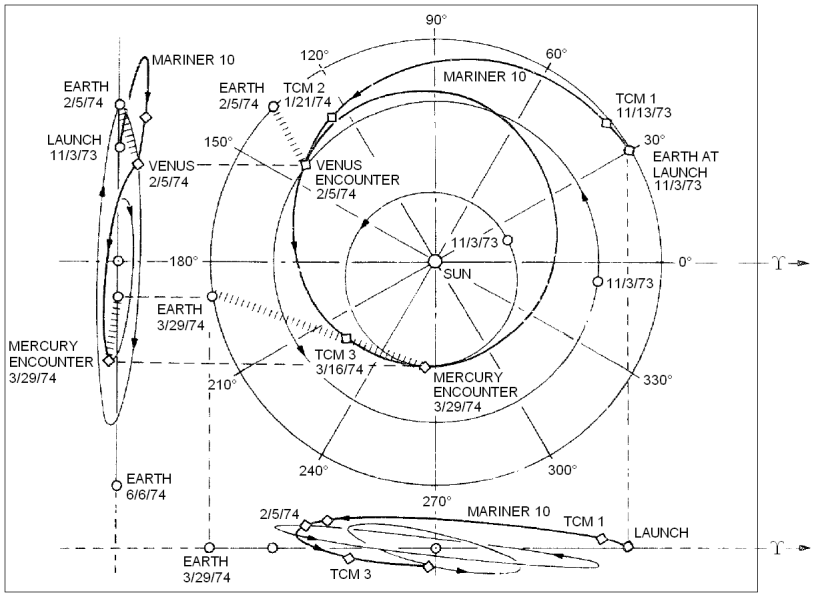

Figure 7.8: Mariner 10 Actual Trajectory [35]

Since the actual mission dates are known, the TOF windows are kept small in order to let STOpS-FLITE search the problem space fully with fewer migrations. Thus, the mission parameters provided to STOpS-FLITE are presented in Table 7.13. The limits are a five day window on either side of the actual launch date and a five day window in which the actual flyby and arrival dates sit. In order to discourage STOpS-

FLITE from using PGAs to recreate the effect of the TCMs, the flyby penalty cost function was given a higher weight than in previous scenarios, as shown in Table 7.14.

Table 7.13: Mariner 10 PGA Trajectory Limits

Parameter	PGA Lower Limit	PGA Upper Limit	Units
TOD	11/01/1973	11/10/1973	Date
TOF_1	90	95	days
TOF_2	47	52	days
r_{p1}	6052	616000	km
ψ	-180	180	deg
ϕ	-90	90	deg
δv	0	5	km/s
α	-180	180	deg
β	-90	90	*deg*

Table 7.14: Mariner 10 Trajectory Cost Functions

Cost Function	Weight	Units
Departure δv	1	km/s
Flyby Penalty (PGA δv)	4 (0.5)	km/s
TOF	0.01	days

Using these parameters, an accurate model of the Mariner 10 mission was defined. The parameters defining this solution set in STOpS-FLITE are shown in Table 7.15:

Table 7.15: Mariner 10 STOpS-FLITE Solution

Parameter	Lowest	Highest	Average	Units
TOD	11/03/1973	11/06/1973	11/04/1973	Date
TOF_1	91.103	93.262	92.446	days
TOF_2	48.117	50.010	48.969	days
r_p	9706	10390	10123	km
ψ	26.353	29.021	28.034	deg
ϕ	35.640	37.635	36.666	deg
δv	0.000	0.000	0.000	km/s
α	-118.647	79.392	-61.224	deg
β	0.000	28.647	7.997	deg
δv_{Launch}	4.302	4.375	4.323	km/s
$\delta v_{FlybyPenalty}$	0.154	0.357	0.271	km/s
Cost	10.650	11.565	11.142	N/A

From Table 7.15, it is clear that STOpS-FLITE was able to converge to a uPGA when it is the most optimal choice. While the α and β angles have extremely wide ranges, it is important to note that the δv of 0 means these angles have no effect on the trajectory. Thus, STOpS-FLITE will simply output whichever angles happened to be correlated with the solution set that was most optimal in the other variables, as the angles no longer affect the final cost. The δv ranged from values on the order of 10^{-4} to true 0 within the population vector of the solution set. Similarly, all the variables related to the flyby itself (not the thrust maneuver) converged to within a small range, other than the single outlier of the November 6^{th}, 1977 launch date which represents the largest deviation in the first seven variables. Launch δv was very consistent as well, with some greater variance in the flyby penalty δv, with total costs all within 1 of each other. This solution is plotted in Fig. :

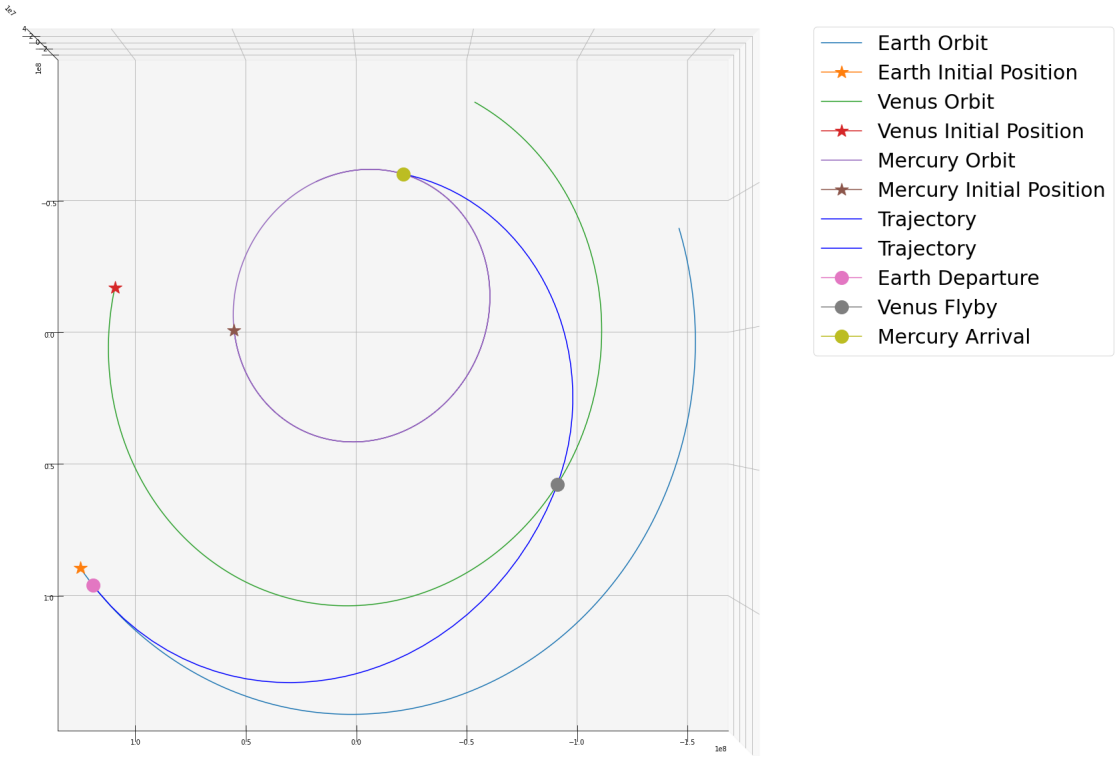

Figure 7.9: Mariner 10 STOpS-FLITE Trajectory

The PGA flyby with a δv at periapsis of 0 km/s is identical both visually and mathematically to a uPGA, thus the plot of its trajectory is symmetric about the plane defined by the radius of periapsis and angular momentum vectors (i.e. the x"-z" plane of the OP_B frame). This is visually evident in Fig. 7.10:

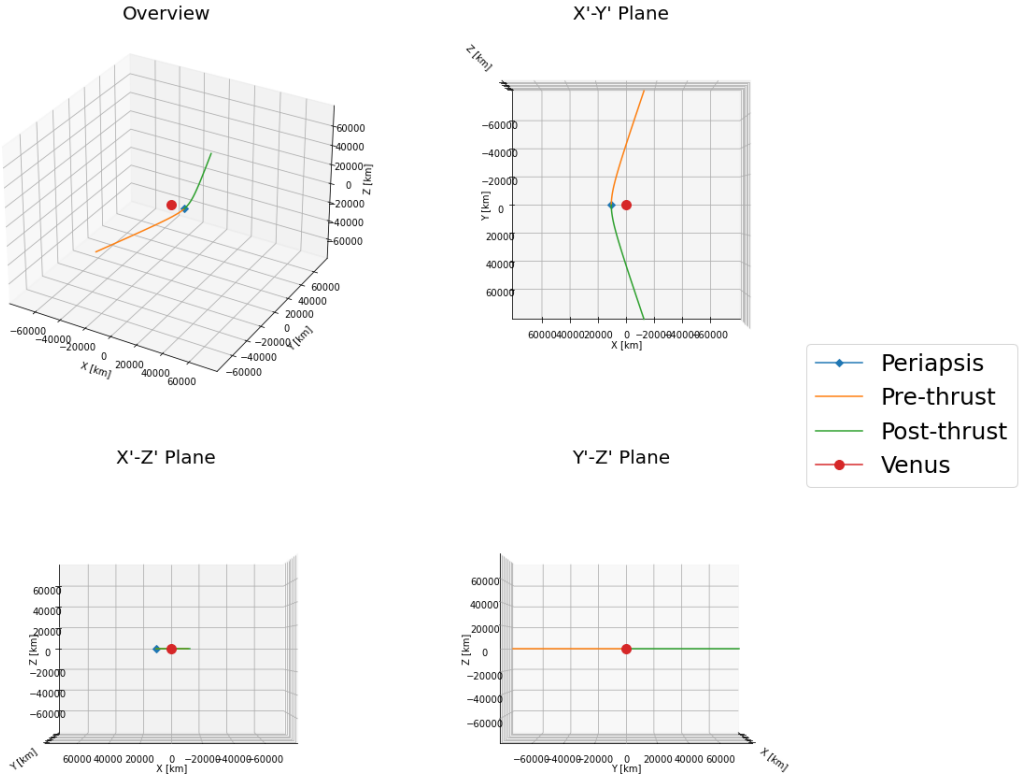

Figure 7.10: Mariner 10 STOpS-FLITE Venusian Flyby (OP_A Frame)

Compared with Mariner 10's actual trajectory, the converged solution of STOpS-FLITE is very similar visually, once the difference in reference frame between Figs. 7.8 and 7.9 is recognized. The actual mission dates are within a day of the STOpS-FLITE solution (other than the final arrival time) and the actual flyby radius is within 2000 km of that of the STOpS-FLITE solution. These are compared in Table 7.16. The TOFs were converted to dates to compare to the actual mission dates of flyby (DOFB) and date of arrival (DOA).

Table 7.16: Mariner 10 Comparison (STOpS-FLITE Average vs. Actual)

Parameter	STOpS-FLITE	Actual	Units
TOD	11/04/1973	11/03/1973	Date
$DOFB$	02/05/1974	02/05/1974	days
DOA	03/25/1974	03/29/1974	days
r_p	10123	11819	km

The difference in launch date and arrival date are likely a result of the lack of TCMs and the difference in radius of periapse. However, the flyby itself is almost exactly the correct time, coming within 2 hours of the actual mission flyby date and time. In addition, while not modeled here, Mariner 10 would go on to perform a triple flyby of Mercury, after its first arrival at Mercury. Thus, there may be some differences from the STOpS-FLITE optimal solution in the actual mission to set up for those flybys.

Lessons learned during implementation and testing include the requirement to narrow TOD and TOF windows to properly encourage convergence to a good solution. Since the PGA-related angles need to span the entire 3D space, they cannot be narrowed. Similarly, unless there is *a priori* knowledge about the optimal radius of periapse, the r_p value must span the range from the planet's atmospheric radius to the edge of the planet's SOI. The δv limiter for the thrust applied at periapse can be adjusted based on the spacecraft, so its range will be dependent on what the thrusters on board the spacecraft can provide. Thus, the biggest way a mission designer can help STOpS-FLITE to converge consistently to an optimal solution is to run a large number of tests with wide ranges for TOD and TOFs, but narrow them once a general idea of where the optimal location in the search space is. In future, a local search algorithm may assist in reducing the need for this window narrowing.

In summary, STOpS-FLITE is capable of converging to a true uPGA trajectory (or close enough that a uPGA model would result in little to no difference) when it is the most optimal solution. Proper weighting can assist in this, and as such the mission designer can adjust the weights according to their specific needs. Ultimately, the optimal solution will be highly dependent on these weighting selections, so a proper understanding of what the mission requirements and objectives are is necessary to properly select these weights, along with sufficient testing.

Chapter 8

CONCLUSION

8.1 Summary

STOpS-FLITE is shown to converge to a wide variety of trajectories using PGA modeling. It is capable of converging to single and multiple flyby missions, as well as PGA and uPGA trajectories, depending on the scenario. PGA modeling allows STOpS-FLITE to generate more options in terms of trades between δv and TOF, turn a mission that was impossible with uPGAs into a possible PGA trajectory, account for a missed launch window, and/or perform thrust maneuvers both at the edge of the SOI and at the location of closest approach. All of these present a mission designer more options and the freedom to perform trade studies on the effectiveness of a PGA for their specific mission. Proper selection of cost weights, sufficient migration numbers, and the narrowing of TOF windows in comparison to STOpS were key factors in increasing convergence to optimal solutions. In addition, the ability to perform thrusts at the periapse allows a trajectory to converge to a solution with a larger room for error in flyby periapse radius when a uPGA trajectory converges to a flyby that exactly matches the atmospheric radius of the planet. Convergence to a small window of thrust angles and thrust magnitude also suggests some robustness against thrust application errors, as well. Though these PGAs did not always maximize change in orbital energy, the small sacrifices in that parameter were used to generate significant and relevant decreases in δv, TOF or both. STOpS-FLITE was able to converge to solutions that were comparable or more optimal than the equivalent uPGA trajectory in two hypothetical scenarios, while also converging properly to a uPGA trajectory

matching the one flown by Mariner 10. In sum, PGA modeling is another tool in the STOpS toolbox as a modern interplanetary trajectory optimization tool which provides mission designers with options to consider and potentially select.

8.2 Future Work

STOpS has significant room for growth in order to consider the expansive problem that is spacecraft trajectory optimization. A few examples of potential work moving forward are presented in this section.

There are a few ways in which STOpS-FLITE and its implementation of PGAs could be expanded. First, making the algorithm more robust to allow for the thrust to be applied anywhere rather than at periapsis or at the edge of the SOI could lead to some interesting results. While periapsis may be the best location to burn in order to achieve ΔE_{max}, other locations for burns may be better for targeting a specific exit trajectory. The natural expansion of that would be performing continuous thrust maneuvers within the SOI, as algorithms like the one in Qi and Ruiter approximate continuous thrust as multiple small impulsive impulses to set an initial guess for the correct thrust direction and timings [14]. This could allow for modeling of both impulsive and continuous thrust maneuvers from chemical or electric propulsion. Along with this, a model of planetary atmospheres could allow for the implementation of aerobraking as a "continuous thrust" maneuver or follow the works of Piñeros et al. to consider aerogravity assists [15]. Expanding to implement a three-body or n-body modeling of the PGA may also net interesting results, and some literature in those assumptions within studies of PGAs for orbital energy change exist already. Finally, a future variant of STOpS could also allow the user to choose an orbit around a planet to target with thrust maneuver(s) within the SOI.

Beyond simple expansions of the PGAs, there are other areas in which STOpS could explore new regions of the spacecraft trajectory optimization problem space or improve its convergence to optimal solutions. For example, Rockett postulated that integrating a process in which the TOF windows are progressively narrowed and adding a local search system once the window is small enough could allow convergence to an even more optimal solution. An automated narrowing algorithm could also solve the issues of split convergence from the first test case in the Results chapter. By defining a "basin", a local search system such as an NLP solver could quickly converge to the best option in the basin. Other improvements suggested by multiple previous STOpS theses include the addition of celestial bodies that are not planets, such as asteroids, moons, or even interstellar objects. This would allow for modeling of missions which visited these celestial bodies and provide even more options for gravity assists. Resonant flybys and orbital synodic periods are also not currently modeled and may provide new ways for STOpS to reduce a trajectory's cost. This could help find optimal trajectories for future missions using planetary orbital position history and trends. Attitude dynamics modeling could allow for STOpS to explore potential errors in thruster pointing or thrust application, potentially leading to more robust trajectories which can achieve their objectives with some margin of safety.

More grandiose options for improving the optimization scheme in STOpS (agnostic of the application to spacecraft trajectories) is true multi-objective optimization, trajectory pruning, or the addition of another metaheuristic algorithm. Multi-objective optimization could improve the way that STOpS considers different trajectories and their benefits and flaws, especially in search spaces as large as the one presented by the PGA formulation used in STOpS-FLITE. Trajectory pruning was also suggested by previous STOpS theses to improve convergence times and avoid going down non-optimal paths. The addition of another metaheuristic algorithm may not be beneficial in and of itself, but as Rockett showed with the implementation of ACAs, it could be

useful for specific orbital trajectory optimization problems. STOpS could also benefit from improvements to code efficiency to reduce their required computation time.

Finally, there are the features of MATLAB variants of STOpS that have yet to be included in Python variants of STOpS. There is yet to be a version of STOpS in Python that can optimize low thrust trajectories or account for environmental perturbations. These are important and relevant considerations for any modern interplanetary trajectory optimizer, with both significant literature to work from and significant room to expand with future work. Lastly, the greatest change to user friendliness is the loss of the GUI from STOpS-1. The creation and implementation of one could help future users and coders of STOpS greatly in visualizing and understanding the outputs of STOpS.

BIBLIOGRAPHY

[1] H. Oberth, *Ways to Spaceflight*. Agence Tunisienne de Public-Relations, 1970.

[2] H. D. Curtis, *Orbital Mechanics for Engineering Students*. Butterworth-Heinemann, 2013.

[3] D. A. Vallado and W. D. McClain, *Fundamentals of Astrodynamics and Applications, 4th edition*. Microcosm Press/Springer, 2013.

[4] A. Shirazi, C. Josu, and J. Lozano, "Spacecraft Trajectory Optimization: A review of Models, Objectives, Approaches and Solutions," *Progress in Aerospace Sciences*, 2018. https://doi.org/10.1016/j.paerosci.2018.07.007.

[5] D. Conte and D. Spencer, "Targeting the Martian Moons via Direct Insertion into Mars' Orbit," 08 2015.

[6] R. A. Brourke, "The Celestial Mechanics of Gravity Assist," tech. rep., University of Texas, Austin, 1988. https://doi.org/10.2514/6.1988-4220.

[7] National Aeronautics and Space Administration, "Basics of Spaceflight: Section 1: Environment, Chapter 4: Trajectories." https://solarsystem.nasa.gov/basics/chapter4-1/.

[8] V. A. Chobotov, *Orbital Mechanics*. American Institute of Aeronautics and Astronautics, 2002.

[9] P. R. Blanco and C. E. Mungan, "Rocket Propulsion, Classical Relativity, and the Oberth Effect," *The Physics Teacher*, vol. 57, no. 17, 2019. https://doi.org/10.1119/1.5126818.

[10] P. R. Blanco and C. E. Mungan, "High-speed escape from a circular orbit," *American Journal of Physics*, vol. 89, no. 72, 2021. https://doi.org/10.1119/10.0001956.

[11] A. F. Silva, A. F. B. A. Prado, and O. C. Winter, "Powered Swing-By Maneuvers around the Moon," *Journal of Physics: Conference Series*, vol. 465, no. 1, 2013. http://dx.doi.org/10.1088/1742-6596/465/1/012001.

[12] A. F. B. d. A. Prado, "Powered Swingby," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 5, 1996.

[13] A. F. B. A. Prado and G. de Felipe, "An analytical study of the powered swing-by to perform orbital maneuvers," *Advances in Space Research*, vol. 40, no. 1, 2007. https://doi.org/10.1016/j.asr.2007.04.098.

[14] Y. Qi and A. de Ruiter, "Powered Swing-by with Continuous Thrust," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 1, 2020. https://arc.aiaa.org/doi/pdf/10.2514/1.G004358.

[15] J. O. M. Piñeros and A. F. B. d. A. Prado, "Powered aero-gravity-assist maneuvers considering lift and drag around the Earth," *Astroyphysics and Space Science*, vol. 362, no. 120, 2017. https://doi.org/10.1007/s10509-017-3097-9.

[16] M. Ceriotti, *Global optimisation of multiple gravity assist trajectories*. PhD thesis, University of Glasgow, 2010. https://theses.gla.ac.uk/2003/.

[17] T. J. Fitzgerald, "Spacecraft Trajectory Optimization Suite (STOpS): Optimization of Multiple Gravity Assist Spacecraft Trajectories Using Modern Optimization Techniques," Master's thesis, California Polytechnic State University, San Luis Obispo, 2015. https://doi.org/10.15368/theses.2015.154.

[18] E. Rockett, "Spacecraft Trajectory Optimization Suite (STOpS): Optimization of Spacecraft Trajectories Using Deep Space Maneuvers (DSMs) and Ant Colony Algorithms (ACO)," Master's thesis, California Polytechnic State University, San Luis Obispo, 2021.

[19] European Space Agency, "Cassini-Huygens: Approach and Arrival at Saturn." https://sci.esa.int/web/cassini-huygens/-/34955-approach-and-arrival.

[20] I. M. Ross, *A Primer on Pontryagin's Principle in Optimal Control: Second Edition*. Collegiate Publishers, 2015.

[21] J. Gao, L. Liu, and Y. Wang, "Spacecraft orbit design based on intelligent optimization," *International Conference on Advanced Robotics and Mechatronics*, 2017. https://doi.org/10.1109/ICARM.2017.8273244.

[22] J. Graef, "B-Plane Targeting with the Spacecraft Trajectory Optimization Suite," Master's thesis, California Polytechnic State University, San Luis Obispo, 2020. https://digitalcommons.calpoly.edu/theses/2251.

[23] S. Sheehan, "Spacecraft Trajectory Optimization Suite (STOpS): Optimization of Low-Thrust Interplanetary Spacecraft Trajectories Using Modern Optimization Techniques," Master's thesis, California Polytechnic State University, San Luis Obispo, 2017. https://doi.org/10.15368/theses.2017.82.

[24] L. D. Kos, T. Polsgrove, R. C. Hopkins, D. Thomas, and J. A. Sims, "Overview of the Development for a Suite of Low-Thrust Trajectory Analysis Tools," tech. rep., National Aeronautics and Space Administration, 2006. https://doi.org/10.2514/6.2006-6743.

[25] T. Polsgrove, L. Kos, and R. Hopkins, "Comparison of Performance Predictions for New Low-Thrust Trajectory Tools," tech. rep., National Aeronautics and Space Administration, 2006. https://doi.org/10.2514/6.2006-6742.

[26] *Recent Improvements to the Copernicus Trajectory Design and Optimization System*, 2012. https://ntrs.nasa.gov/citations/20120001856.

[27] D. Morante, M. S. Rivo, and M. Soler, "A Survey on Low-Thrust Trajectory Optimization Approaches," tech. rep., Multidisciplinary Digital Publishing Institute, 2021. https://doi.org/10.3390/aerospace8030088.

[28] "Java Astrodynamics Toolkit (JAT)." https://opensource.gsfc.nasa.gov/projects/JAT/.

[29] R. P. Oldenhuis, "Trajectory Optimization of a Mission to the Solar Bow Shock and Minor Planets," Master's thesis, Delft University of Technology, 2010.

[30] D. Izzo, "Parallel Global Multiobjective Optimizer (PaGMO)." https://github.com/esa/pagmo.

[31] M. G. Malloy, "Spacecraft Trajectory Optimization Suite (STOpS): Design and Optimization of Multiple Gravity-Assist Low-Thrust (MGALT) Trajectories Using Modern Optimization Techniques," Master's thesis, California Polytechnic State University, San Luis Obispo, 2020. https://digitalcommons.calpoly.edu/theses/2247.

[32] E. Woods, "Orbital Optimization of Interplanetary Trajectories with Environmental Perturbations," Master's thesis, California Polytechnic State University, San Luis Obispo, 2018. https://doi.org/10.15368/theses.2018.64.

[33] D. Bologna, "Spacecraft Trajectory Optimization Suite: Preliminary Design of Interplanetary Trajectories Using Pseudostate Theory," Master's thesis, California Polytechnic State University, San Luis Obispo, 2022.

[34] E. Doughty, "Interplanetary Trajectory Optimization with Automated Fly-by Sequences," Master's thesis, California Polytechnic State University, San Luis Obispo, 2020.

[35] Jet Propulsion Laboratory, "Mariner venus-mercury 1973 final report," Report 33-374, California Institute of Technology, 1976. http://ser.sese.asu.edu/M10/M10_PDF/TM33-7_3.PDF.

APPENDICES

Appendix A

USER GUIDE

A.1 Code Base

The STOpS-FLITE code consists of two segments: *main.py* and *STOpS_FLITEvFinal.py*. The *main* file simply has the call to run STOpS-FLITE, along with the indicator of which mission should be run, the number of migrations to utilize, and some lines of code to make Spyder (the preferred IDE for STOpS variants in Python) create a noise signal when all migrations are complete. The *STOpS_FLITEvFinal* file houses all of the functions used by STOpS-FLITE, as well as some vestigial code from STOpS-PY and STOpS-DSM. The folder including both of these files also includes function files for all of the sub-functions used within all previous Python variants of STOpS, so that they can be referenced or re-implemented if needed.

A.2 Inputs

The inputs of STOpS-FLITE can be loosely categorized into two types: cost weights and mission limits. For ease of reference, the block diagram displayed in the Implementation chapter is reprinted here in Figure A.1. Recall that most of the architecture remains from previous STOpS variants with very few changes, the most significant of which were the removal of ACO and the addition of the PGA algorithm.

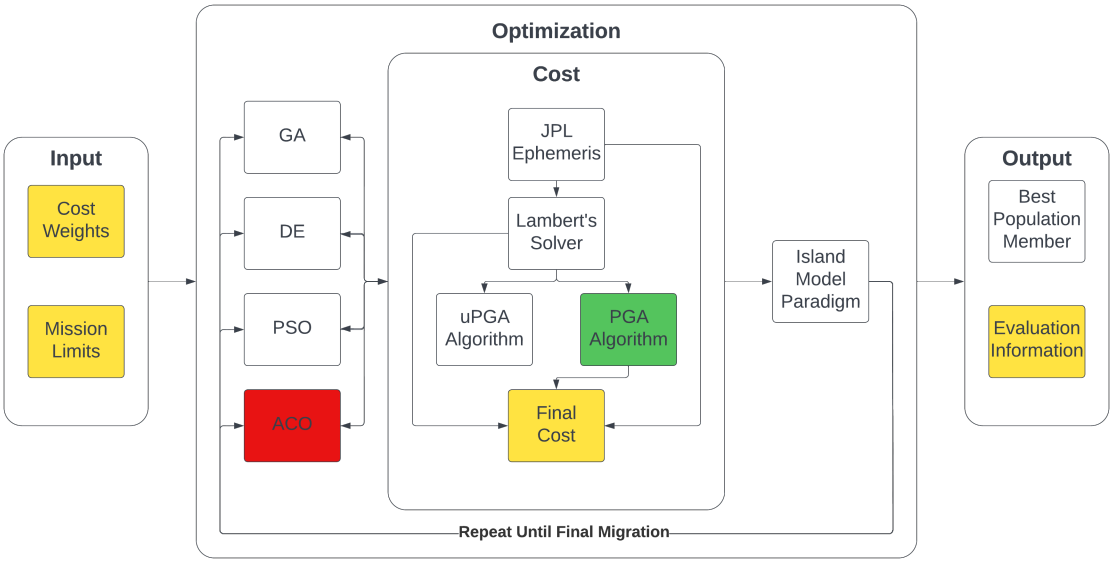

Figure A.1: STOpS-FLITE Block Diagram

The easiest way to adjust these is to either modify an existing "Preset Mission" or creating a new one. These can be found starting on Line 48 of the final STOpS-FLITE code or by searching for *class presets()*. If modifying a mission, one simply needs to adjust the cost options within one of the three defined missions and then call that mission to run from Line 6 of the *main.py* file. If a new one is created, it must be defined within *class presets()* with a new name, as well as being added to the Mission Dissection and Definition segment which begins on Line 288 of the final STOpS-FLITE code.

Cost weights can be found in each of the standard missions under the moniker "Cost Options". Cost weights are the multiplicative weight attached to each of the chosen mission objectives. For example, a cost weight of 2 applied to the launch δv means that the value for the launch δv will be multiplied by 2 in the final cost. The application of thrust within the PGA algorithm has two stacking multipliers, one from the flyby penalty as a whole and one specifically for the thrust applied at periapse. This allows the user to specify different weights for the thrusts at the edge

of the SOI and at periapse. Since the thrusts at the edge of the SOI are used to match up the v_∞'s from the Lambert's solver and the PGA algorithm, they can be considered to corrective thrusts, though they may also be part of an optimal solution per Qi and de Ruiter's work [14]. From testing, the weighting never precluded edge of SOI thrusts or periapse thrusts, but rather changed how much of the thrust was applied at each location (i.e. a higher weight on the periapse thrust would push more of the thrust to the edge of the SOI). Since the periapse thrusts were preferred when testing for convergence to a PGA, they were typically weighted at half the weight of the overall flyby penalty weight. Thus, the periapse thrusts were multiplied by a weight that was half the value of the edge of SOI thrusts. The remainder of the δv weights were selected to match the periapse thrust weights. For example, if the overall flyby penalty weight was 2, and the periapse thrust weight modifier was 0.5, then the remaining δv weights were set to 1 to match the final weight on the periapse thrust cost. TOF cost weights were selected to put the TOF costs to be on the same order as the δ costs. Thus, if the total TOF was in the order of hundreds of days, then the TOF cost multiplier is selected to be 0.01, as δv is usually between 1 and 10 km/s for a trajectory. This can be adjusted after some initial testing. Finally, the flyby altitude penalties and heliocentric velocity cost options should also be adjusted to be on the same magnitude as the other costs being used. However, the PGA algorithm can drastically change both of these within the same mission, and thus they can have values which span multiple orders of magnitude, making it difficult to choose proper weights.

Mission limits can be found in two segments of the code. Those related to the planet flyby order and the TOFs to be fed into the Lambert's solver are found at the top of each of the standard missions. Within the "Start" category, the planet to start at, as well as the window for time of departure can be selected. The "Waypoints" category stores a vector representing the planets to be used for gravity assists. It

also contains a vector representing the lower limits of the TOF windows and one representing the upper limits of the TOF windows. For example, for *standard()* (the mission corresponding to the EJS trajectory), the "Waypoints" planet vector is *[5]*, which sets Jupiter as the gravity assist planet. The "Shortest Trans-Time" array is *[450,700]* and the "Largest Trans-Time" array is *[650,750]*, representing the TOF_1 window of 450 to 650 days and the TOF_2 window of 700 to 750 days. Finally, "End" stores solely a number corresponding to the arrival planet. The other mission limits are those corresponding to the PGA algorithm, and can be found in the *read_inputs(mission)* function on Line 2375 of the final STOpS-FLITE code. The angles α, β, ϕ, and ψ are defined to allow the PGA algorithm to fully explore any direction of thrust/location of radius of periapse, and therefore should not be modified without a valid reason. Similarly, the radius of periapse window is defined between the atmospheric radius and the SOI radius so as to make sure the spacecraft does get affected by the planet's gravity while also not entering the atmosphere, and therefore should not be adjusted without a valid reason. This leaves the δv limiter, which should be modified if STOpS-FLITE converges to a solution at the upper limit of the δv window (as it likely needs more δv to reach the true optimum) or if there is sufficient knowledge of the spacecraft's propulsion system.

A.3 Outputs

The outputs fall into two segments as well. The first is the best population member. Within the Variable Explorer of Spyder (the preferred IDE for all STOpS variants in Python), this can be found under *data* $\rightarrow$ *Heliocentric* $\rightarrow$ *pop*. The other is evaluation information, which is a catch-all term for any information regarding the optimal solution that is valuable to the user. These can be found in *mission* $\rightarrow$ *Eval Info*. The evaluation information in STOpS-FLITE is listed below:

- **Arrive dV**: δv magnitude required to capture at the arrival planet [1x1, km/s]
- **Entry/Exit dV (PGA)**: v_∞ vectors ([1x3]) in the OP_A frame[1] stored together in the following order: v_∞ into the SOI from the Lambert's solver, v_∞ into the SOI from the PGA algorithm, v_∞ out of the SOI from the Lambert's solver, and v_∞ out of the SOI from the PGA algorithm [1x12n[2], km/s]
- **Entry/Exit dV (uPGA)**: v_∞ magnitudes into and out of the SOI, as well as an indicator of if the flyby enters the atmosphere/collides with the planet. This indicator has a value of 1 if the spacecraft is not in the atmosphere, 2 if the spacecraft is in the atmosphere but does not collide with the planet, and 3 if the spacecraft collides with the planet. [1x3n, km/s & count]
- **Flyby Altitude 1 (PGA)**: Magnitude of pre-thrust radius periapse [1x1, km]
- **Flyby Altitude 2 (PGA)**: Magnitude of post-thrust radius periapse [1x1, km]
- **Flyby Altitude (uPGA)**: Magnitude of the radius periapse of equivalent uPGA if using the uPGA algorithm (ignores cost) [1x1, km]
- **Flyby dV**: Total δv achieved by flyby [1xn, km/s]
- **Flyby Penalty (PGA)**: Magnitude of the edge of SOI thrusts from the PGA algorithm [1xn, km/s]
- **Flyby Penalty (uPGA)**: Magnitude of the edge of SOI thrusts from the uPGA algorithm (this will typically Inf to indicate the spacecraft flew through the atmosphere or into the planet) [1xn, km/s]
- **Helio**: Heliocentric energy at end of trajectory [1x1, km^2/s^2]

[1] All vectors are stored in the OP_A frame unless otherwise noted

[2] n = number of flybys

- **Leave dV**: δv magnitude required at launch for first leg of trajectory [1x1, km/s]
- **Periapse Shift**: Magnitude of f_0 (periapse shift as a result of thrust) [1xn, radians]
- **Periapse Velocity (Post-thrust)**: Vector of periapse velocity after thrust is applied [3xn, km/s]
- **Periapse Velocity (Pre-thrust)**: Vector of periapse velocity before thrust is applied (note that since this is in the OP_A frame, it will be purely in the y'-axis) [3xn, km/s]
- **Print**: Command for STOpS-FLITE to print evaluation information to the console [1x1, boolean]
- **TOFs**: Time of flights for each leg [n+1, days]
- **Trajectory JDs**: Julian dates of departure, flybys, and arrival [n+2, days]

Some of these are also population members and therefore are included in *Eval Info* in order to be more easily printed to the console. The remainder of these are calculated inside the *flyby_penalty()* function or within the main cost function.