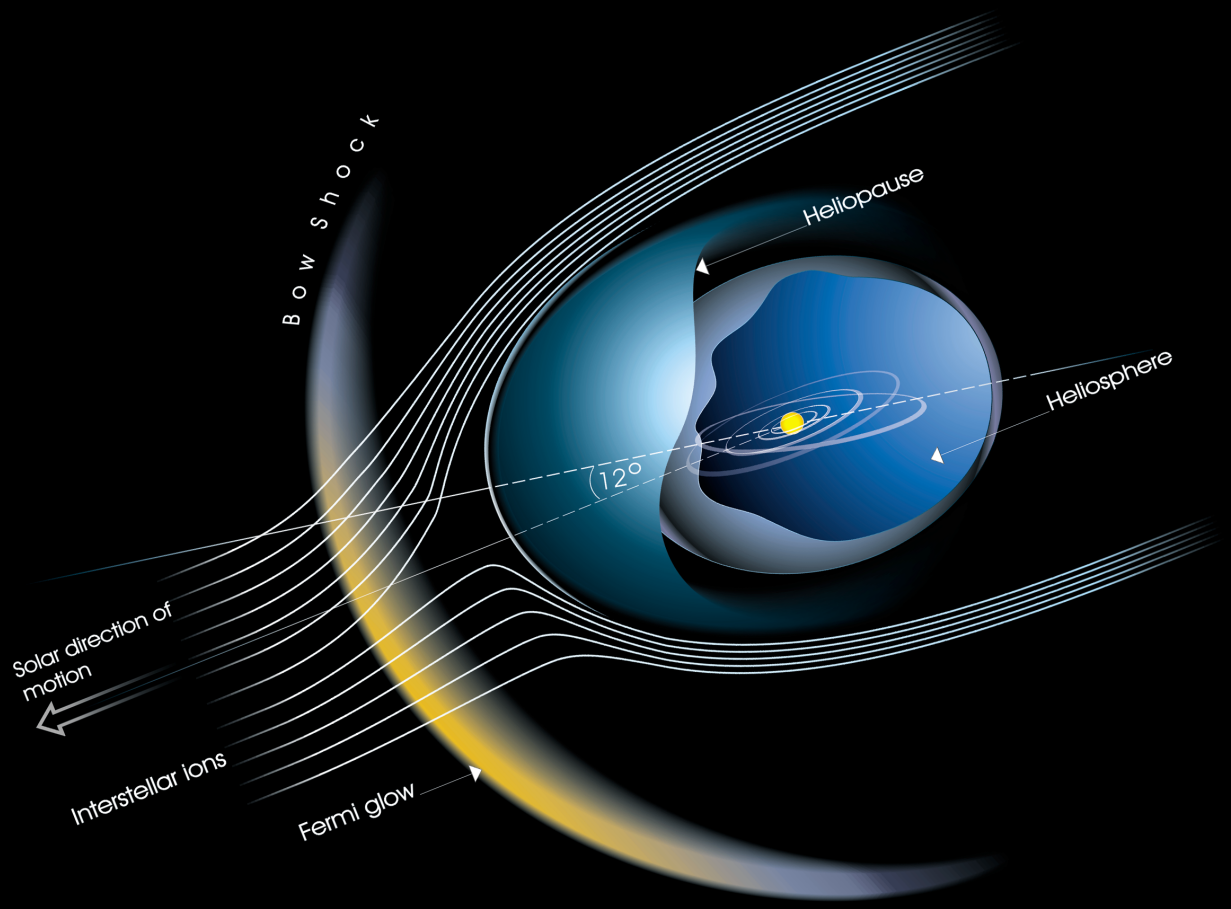
Trajectory Optimization for a Mission to the Solar Bow Shock and Minor Planets



Thesis Report

Rody P.S. Oldenhuis

January 25, 2010

JAXA TU Delft

Trajectory Optimization of a Mission to the Solar Bow Shock and Minor Planets

Thesis Report

Rody P.S. Oldenhuis

January 25, 2010

Delft University of Technology
Faculty of Aerospace Engineering
Astrodynamics & Satellite Systems
Kluyverweg 1, 2629 HS Delft
The Netherlands

TU Delft

Cover figure adopted from `http://www.spacetelescope.org/images/html/heic9911a.html`.

Document typeset with LaTeX 2_ε, using the packages `graphicx, eso-pic, subfigure, amsmath, amssymb, rotating, natbib, url, ccaption, wrapfig, makeidx, multirow, glossaries, longtable, geometry,hyperref, floatflt, fncychap, tabularx, quotchap, slashbox, fancyvrb, relsize and fancyhdr`.

Original (uncited) images created with InkScape 0.46 (with the `Tex Text` extention), Adobe® Photoshop®, Microsoft® Visio®, Microsoft® Paint or MATLAB® R2008b/R009b.

Acronyms

AD	Automatic Differentiation. Technique in computer programming that uses operator overloading to compute the derivatives of any function automatically, to within machine precision.
AOF	Aggregate Objective Function
BFGS	Broyden-Fletcher-Goldfarb-Shanno formula's. Formulas that enable approximating the Hessian of a system of equations, rather than computing it directly.
BOL	Beginning Of Life. Parameters as they are when the production of the hardware in question is just complete.
DE	Differential Evolution
DS4G	Dual-Stage 4-Grid ion engine
DSM	Deep Space Manoeuver
EKB	Edgeworth-Kuiper belt
ENA	Energetic Neutral Atom
ESA	European Space Agency
ExpoSin	Exponential Sinusoid
FE	Function Evaluation
GALOMUSIT	Genetic ALgorithm Optimization of a MUltiple Swing-by Interplanetary Trajectory
GAM	Gravity Assist Manoeuver
GA	Genetic Algorithm
GODLIKE	Global Optimum Determination by Linking and Interchanging Kindred Evaluators
GTOC	Global Trajectory Optimization Competition. Annual competition originally started by ESA, in which a large number of teams worldwide compete to come up with the best design for a space mission to (a) challenging target(s).
GTO	Geostationary Transfer Orbit
IBEX	Interstellar Boundary Explorer
JAXA	Japan Aerospace Exploration Agency
KKT	Karush-Kuhn-Tucker. First-order optimality conditions of an arbitrary constrained optimization problem.
LISM	Local Interstellar Medium
LIW	Local Interstellar Wind
MAB	Main Asteroid Belt
MGA	Multiple Gravity-Assist
MMRTG	Multi-Mission Radioisotope Thermoelectric Generator
MOO	Multi-Objective Optimization
MOPSO	Multi-objective Particle Swarm Optimizer
MPCORB	Minor Planet Center Orbital Database
MPCe	Minor Planet Center
MPC	Method of Patched Conics
MPE	Method of Patched ExpoSins

MPμC	Method of Patched Micro-Conics
MP	Minor Planet. All bodies in the Solar system that are *not* planets, natural or artificial satellites, or the Sun.
MS	Multi-Start
NASA	National Aeronautics and Space Administration
NBI	Normal Boundary Intersection
NEP	Nuclear Electric Propulsion. Electric propulsion powered by nuclear batteries.
NLP	Non-Linear Programming
NSGA-II	Non-dominated Sorting Genetic Algorithm, improved version
NSGA	Non-dominated Sorting Genetic Algorithm
ODE	Ordinary Differential Equation
OPTIDUS	OPTimization of Interplanatary trajectories by Delft University Students
PCT	Parallel Computing Toolbox. `MATLAB`-toolbox that allows executing any `M`-code on a computer cluster/multiple CPU's.
PSO	Particle Swarm Optimization.
PUI	Pick-up Ion
QP	Quadratic Program
RKN	Runge-Kutta-Nyström. ODE-integrator especially suited for second-order ODE's of the form $y(t)'' = f(t, y(t))$.
RTG	Radioisotope Thermoelectric Generator
SA	Simulated Annealing
SBS	Solar Bow Shock
SDO	Scattered Disc Object
SEP	Solar Electric Propulsion. Electric propulsion powered by Solar panels.
SF	Solar flyby
SOI	Sphere of Influence. Spherical region surrounding a celestial body in which the body's gravitational attraction dominates over that of all other bodies.
SQP	Sequential Quadratic Programming. Iterative optimization technique that bases its steps on the solutions of locally approximated QP's.
SRG	Stirling Radioisotope Generator
STM	State Transition Matrix. A quick way to calculate the Cartesian statevector any time step removed from an initial statevector, in a two-body context.
SW	Solar wind
TNO	Trans-Neptunian Object
Xe	Xenon

List of Symbols

ΔV	Change in the spacecraft's velocity.	[km/s]
I_{sp}	Specific impulse; performance parameter for spacecraft engines.	[s]

m	Number of complete revolutions (for Lambert problems).	[-]
m_{dry}	Spacecraft dry mass; the spacecraft's mass after all consumables have been exhausted.	[kg]
$\mho$	Number of swingby planets in `seq`.	[-]
μ	Standard gravitational parameter of the central body.	[$\text{km}^3\ \text{s}^{-2}$]
m_{wet}	Spacecraft wet mass; the spacecraft's mass just after leaving the Earth's sphere of influence.	[kg]
r_a	Apocenter distance; largest distance to the central body.	[km], [AU]
r_p	Pericenter distance; smallest distance to the central body.	[km], [AU]
`seq`	Sequence of swingby planets/bodies.	[-]
t_f	Time of flight, or transfer time.	[s], [days]
V_{esc}	Local escape speed/velocity.	[km/s]
V_∞	Hyperbolic excess speed/velocity.	[km/s]

Abstract

In November 2003 the Voyager 1 spacecraft pierced through the Solar system's termination shock, followed by Voyager 2 in August 2007. These events marked the beginning of interstellar exploration by spacecraft, which caused a wave of renewed interest in the outer heliosphere throughout the scientific community. This thesis research investigated the feasibility of a mission to the Solar bow shock (the true interstellar boundary) at ∼200 AU from the Sun; twice as far as the Voyager spacecraft. The techniques used in designing its trajectory were standard gravitational-assist manoeuvres and/or a single close-proximity flyby with the Sun, and both low-thrust and high-thrust propulsion systems were investigated. Minimum time of flight and maximum spacecraft dry mass were primary objectives, whereas the number and quality of minor planet flybys were secondary objectives. A maximum on the time of flight of 15 years was upheld, extendable to a maximum of 25 years in case the first constraint could not be met or proved overly beneficial to the other objectives. This constraint necessitated upholding a maximum of 3 gravitational-assist manoeuvres, which gave rise to a search space of 146 unique flyby sequences. A novel optimization algorithm developed in this thesis (GODLIKE) identified 3 feasible sequences, which were only feasible in combination with a high-thrust propulsion system. The most promising sequence (Earth/Jupiter/Uranus/bow shock) proved capable of reaching the bow shock with minimal risk and impact on the spacecraft's dry mass. The best result found requires a ΔV of 5.86 km/s and approaches approximately 14 minor planets to within 0.03 AU. The total time of flight could unfortunately not be kept below 15 years; technically feasible trajectories could only be generated for times of flight between 23 and 25 years. Despite the long time of flight, the results developed in this research do show that it is possible to traverse the entire heliosphere using time-proven and conventional means.

Preface

"To boldly go where no one has gone before"

This famous quote from Gene Roddenberry's popular television series "Star Trek" left quite the impression on me as a child. Like so many other people I met during my studies, I reluctantly admit that it was probably this quote that ignited my interest in space exploration. After having tasted some applied physics and astronomy, I realized that space exploration is really still in its infancy, even though the field is now more than 50 years old. I quickly grew an affection for the enormous complexities and difficulties involved with space exploration, which finally, after 8 years of study, led to this master's thesis: a detailed study of exactly how far current technology can bring us into space.

Naturally, my affection with complicated technical things did not limit itself to space exploration alone. As the computer revolution steadily progressed, I quickly became hooked on learning all about these fascinating machines. I played with them as much as I could afford, usually getting much more out of them than they were originally designed to do. Inadvertently this "thirst for more" also confronted me with a wide variety of programming languages, starting with QuickBASIC and C++ to Python and Prolog. For this master's thesis, one of my personal goals was to assure myself of the fact that after its completion I would know and understand all the pesky little details of trajectory optimization and design. I figured the best way to force myself to do this was to write all software I needed from scratch, which resulted in a sizable program I dubbed Skipping Stone.

I would like to use this opportunity to thank my supervisor, Ir. Ron Noomen, whom I repeatedly caught telling me exactly those things I needed to hear to keep me inspired and on the right track. I would also like to thank my dad, who as a proud father spent far too much time on correcting my English and coming up with possible improvements for my thesis. Last but definitely not least, I would like to express a special thank-you to my fair lady Corianne Marges, who helped me through those moments of utter despair (I really hate minus signs...) and has given me so much more than even she is aware of.

Rody P.S. Oldenhuis
Delft, January 25$^{\text{th}}$ 2010

Contents

Part I

Overview

1 Introduction

In November 2005, the National Aeronautics and Space Administration (NASA) issued a press release about a new scientific discovery made by the Voyager 1 spacecraft [Stone et al., 2005][McDonald, 2010]. This raised many eyebrows in the scientific community, for who could have imagined that a 30 year old spacecraft could still make important discoveries? Originally the matter was much debated, but later evidence clearly showed that the claims were indeed correct; the Voyager 1 spacecraft had pierced the Solar system's termination shock and had thus entered *interstellar* space. This event caused a small shock wave throughout many related fields, and caused an enormous increase in interest in all of the science surrounding the heliosphere. Soon anyone who had taken an interest realized that this particular field was virtually devoid of sound knowledge, for almost no one had previously been able to make any substantiating claims. So, as with any gold mine, research on the heliosphere soon became quite a popular affair. It inspired the Cassini/Huygens team to analyze their data with the heliosphere in mind, which lead to the discovery that the heliosphere is *spherical* and not squashed like a comet's tail as was predicted by virtually all models, and seems to be in contradiction with the data gathered by the Voyager spacecraft [Piazza and Wessen, 2010]. The area grew so popular in fact that it has resulted in the launch of a whole new mission (NASA's IBEX), which subsequently discovered that the Solar system is being bombarded with neutral particles from only one particular direction instead of *all* directions as existing models had predicted [Stoyanova and Dunbar, 2009].

In short, the few theories we did have now appear to be incorrect. We now know that we actually understand very little of the Sun's immediate environment, its interaction with its neighbors, what this implies for life on Earth. Now that the Voyager missions have demonstrated that interstellar space missions have become a very real possibility, it seems almost inevitable that we will see more spacecraft set on exploring this part of space *in situ* in the very near future.

It is precisely this thought that inspired this research. This thesis aims to design a trajec-

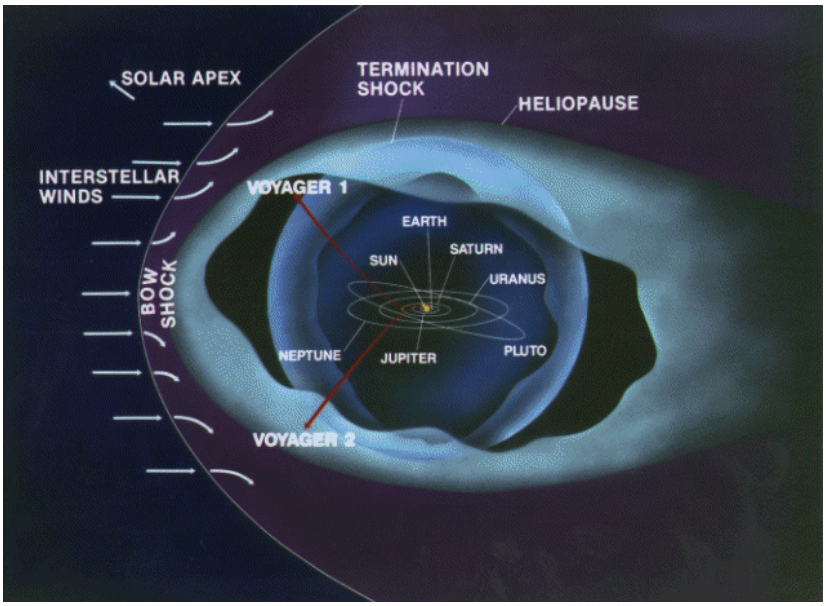


Figure 1.1 Overview of the various shock phenomena associated with the interaction of the Solar and interstellar winds. The locations of both the Voyager spacecraft have also been indicated. Note that the Solar bow shock lies *twice* as far as the Voyager spacecraft have been able to travel in their 30 year lifetime. From [Payne, 2010].

tory for a spacecraft that will travel through the *entire* heliosphere and into true interstellar space – a mission to the *Solar bow shock* – to answer many of the new questions the Voyager missions raised. The Sun moves at a speed of some 300 km/s with respect to the Galactic center [Frisch, 2000b], and so do the other stars and interstellar gas clouds surrounding the Solar system. These speeds all have very different directions however, so that their *relative velocities* give rise to highly supersonic wind speeds. The regions where these winds collide cause various kinds of shock phenomena, of which the Solar bow shock is farthest from the Sun. But it is thought that the stagnation point of the Solar bow shock lies between 200 AU and 230 AU from the Sun, which is more than *twice* as far as the Voyager spacecraft have been able to travel in their 30-year lifetime (see Figure 1). Any spacecraft travelling to the bow shock would encounter not only the shock, but the *entire* planetary region and *all* of the heliosphere, allowing *in situ* measurements on many different poorly understood phenomena.

The Solar and interstellar winds are all composed of plasma, so their motion is strongly affected by interstellar magnetic fields. The Solar bow shock is therefore thought to be not just a supersonic gas shock, but a supersonic magneto-hydrodynamic plasma-shock; particularly, one that we know very little about. One of the many phenomena thought to be associated with the Solar bow shock is that it affects how high-energy cosmic rays enter the Solar system. There even is some speculation that the bow shock serves as a *shielding* mechanism for the most energetic of these rays. If this were true it would have profound implications on life on Earth, and its evolutionary history.

But before anything can be measured, the spacecraft first has to reach the $\sim$200 AU distance. It is hard to imagine just how *vast* a distance that is, but suffice it to say that the

bow shock is an *extremely* challenging target for a space mission. Exactly *how* challenging is illustrated in Figure 1.2. This figure shows the required amount of time to reach the bow shock versus the amount of energy the launch vehicle would have to provide, if the spacecraft were launched directly from Earth to the Solar bow shock. The figure also shows what the most powerful rockets ever constructed can accomplish. Clearly, a direct launch is not a realistic option; even the most powerful rocket available would bring the spacecraft to the bow shock in *more than 120 years*. This thesis investigates whether using *other* techniques can reduce this time to 15 years. It also aims to minimize propellant usage and thus maximize the spacecraft's mass upon arrival, with an absolute lower limit on the mass upon arrival of 50 kg.

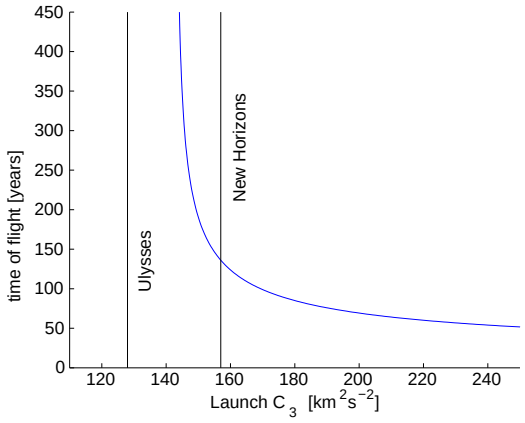
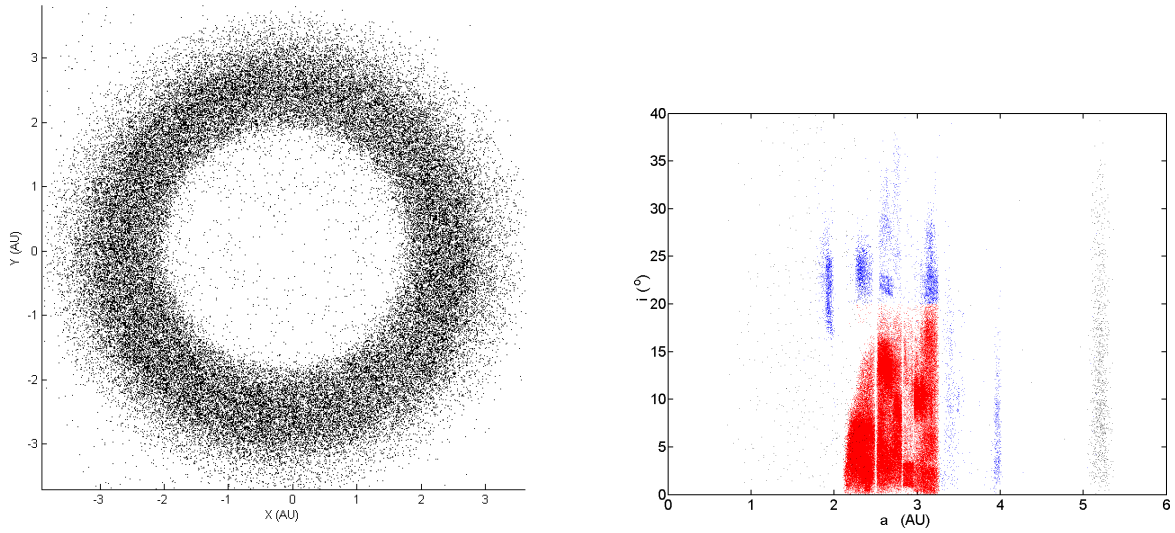


Figure 1.2 The time of flight versus the C_3 parameter for a mission to the Solar bow shock, assuming a direct insertion into a Solar escape trajectory. The most energetic launches ever performed (NASA's *New Horizons* and *Ulysses* spacecraft) have also been indicated.

Even when it found to be possible to penetrate the bow shock in under 15 years, any spacecraft traversing that distance would be completely *idle* for its entire journey and only start producing valuable data 15 years into its journey. Therefore, to increase this mission's value, this research will also investigate whether the spacecraft can perform flybys with several minor planets *en route* to the bow shock. Minor planets, a collective name for asteroids, comets and other non-planetary bodies, comprise the vast majority of bodies in the Solar system. This is illustrated in Figure 1, which shows the most well-known collection of these bodies – the Main Asteroid Belt. Some 400,000 bodies are known to belong to this belt, and some estimates show that the total amount of bodies in this belt can be as high as 1.7 million [de Pater and Lissauer, 2005].

Minor planets are interesting for a variety of reasons. It is generally accepted that they are the remnants of the original planetesimals that formed the planets, making them "living fossils" of the very beginnings of the Solar system. They come in a wild variety of types; comets are normally composed for a large part of water ice ("dirty snowballs") and are relatively easy to analyze because their (periodic) close approach to the Sun vaporizes a portion of their surface, providing both a spectacular sight and an amazing insight into their constituents. Asteroids on the other hand consist mainly of rocky materials and often display the effects of accretion (binary asteroids, "rubble piles", etc.), the process that is thought to have shaped the planets. Then there are the Centaurs, Cubewano's, Plutino's, and many more from a long list of different "personalities" they can assume. But the one thing that makes them most enigmatic is that they often contain a great variety of *organic* materials – the stuff of life. It has often been speculated that the origins of life itself lie in these minor planets.

Considering that there are so many asteroids in the Main Belt alone, it would seem rather easy to launch a spacecraft to one (or even a few) of them. For spacecraft travelling to bodies



(a) Locations of asteroid in the Main Belt

(b) Distribution of semi-major axes versus inclinations

Figure 1.3 Overview of the Main Asteroid Belt

farther out than this belt it would even seem impossible to traverse the belt *without* encountering one, and coming out the other end completely unharmed. This was indeed a scenario feared for when the early Pioneer spacecraft first penetrated the belt. Luckily it turns out that pictures like Figure 1 are rather misleading in this respect; the size of each individual asteroid, as represented by a single pixel in this figure, is greatly exaggerated. Their true *scaled* size would be vastly smaller than what is representable by one pixel, so small in fact that the chance of crashing into an asteroid when travelling through the belt is a good deal less than a millionth of a percent for almost any trajectory[1]. Although this fact is comforting from a risk-assessment point of view, it does imply that it is also quite challenging to steer the spacecraft to even one asteroid in this belt.

But the Main Asteroid Belt is only one of several such belts in the Solar system. It is only the most well-known because it has been known for so long, and it is fairly easy to observe some of the asteroids it contains from Earth. Improvements in telescope quality in recent decades brought to light many more minor planets *not* belonging to the Main Belt. Other collections of such minor planets are the Kuiper Belt, Scattered Disc, Centaurs, and many more. Although *some* of the asteroid from the main belt have been explored by spacecraft, *none* of these other bodies have ever been visited by any man-made probe. This is mostly because these other bodies are *so far away* – which is *precisely* the intent of this mission.

This thesis covers many different aspects of mission- and trajectory design, altogether comprising a substantial amount of material. This thesis has therefore been divided into five separate parts to provide better structure and improve reading comfort. Part I serves as a general introduction to the thesis subject. It also defines and elaborates all the mission

[1] A rough estimate on this probability can be obtained by dividing the total volume of all the asteroids combined (∼4% of the Moon [de Pater and Lissauer, 2005]) by the total volume of the belt (some 30-40 cubic AU) – suffice it to say, that's a *very* small number.

requirements and its constraints. Part II deals entirely with optimization; it discusses various mathematical optimization techniques and their application to trajectory design. Part III aims to meet the mission's objectives by using classical, time-proven chemical rockets. Along the same lines is part IV, which tries to do so with the much more efficient ion-engines. The concluding part V is where the actual design process takes place. It is here where the final design is formulated, elaborated and discussed, and where the conclusions and possible improvements regarding the feasibility are located.

During this research, the amount of material that had to be covered grew a bit out of hand, forcing some of the more "standard material" to the appendices. These appendices are often referred to in the text, but provided that the reader is already familiar with the concepts these appendices embody, they can safely be skipped without any negative impact on the reader's understanding of the research.

2
Mission Targets

The outer Solar system has only been explored by a handful of spacecraft not specifically designed for this particular purpose. Rather, these spacecraft only reached as far as they did because of their high residual orbital energy which they acquired quite by accident during their primary mission. A spacecraft specifically designed to explore the Solar Bow Shock (SBS) would go down in history as the first man-made object to ever *intentionally* reach that far. Because the SBS is so far away, this mission has access to objects or phenomena throughout the *entire* Solar system. In light of this it would seem rather wasteful to *only* visit the SBS. To improve the potential scientific output of such a far-reaching mission, this research also investigates the possibility of making close encounters with one or more minor planets en route to the SBS. Nevertheless, throughout the research, these encounters are treated as the mission's *secondary* objectives, keeping the focus on reaching the SBS. This chapter serves as a general introduction to the mission's primary and secondary objectives. Section 2.1 elaborates the peculiarities and unknowns surrounding the SBS, and section 2.2 introduces some generalities regarding minor planets.

2.1. Solar Bow Shock

Frisch [2000a] wrote a very good introductory text on the SBS. This work is a popularized version of a more in-depth paper on the material [Frisch, 2000b]. Also, much of this material has been summarized in a more readable form by Axford and Suess [1994]. Most of the material in this section is derived from these three sources.

2.1.1 Outer Solar System and Solar Neighborhood

When reading about the "outer Solar system", the authors usually mean the last few in the sequence of planets, the Kuiper belt or Scattered Disc. Naturally, the Solar system doesn't just *end* after the furthest removed Kuiper belt object. The influence of the Sun reaches

much further than that, both gravitationally and via the Solar wind (SW). From the Kuiper belt outward, there is a myriad of interesting bodies and phenomena directly related to the Sun's presence.

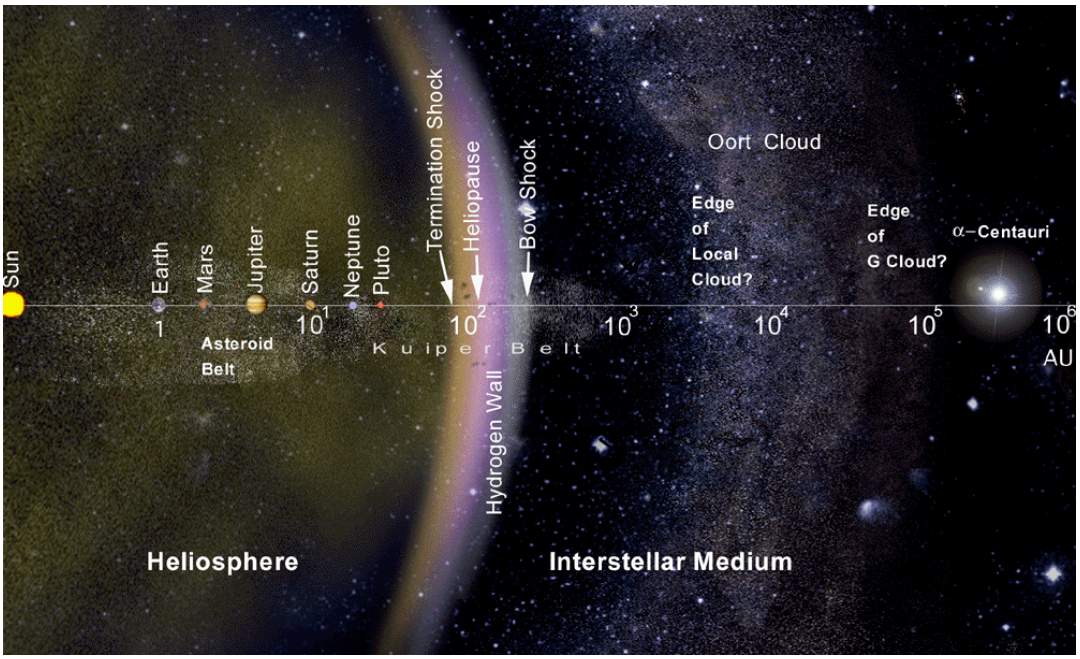


Figure 2.1 The Solar system, on a logarithmic scale. The termination shock, where both Voyager spacecraft have recently pierced through, lies at ∼100 AU. The bow shock lies ∼200-230 AU out. Contrary to popular belief, the bow shock does not define the limits of the extent of the Sun's influence – that is reserved for the Oort cloud, some ∼50.000 AU out. From [Mewaldt and Liewer, 1999].

The *complete* Solar system is given schematically in Figure 2.1. This Figure is a representation of the Solar system, from the Sun out to the nearest star system (α Cen), on a **logarithmic** scale. After the planetary region, several other clusters are encountered due to the various resonances and orbital evolution and history of the various *inner* Solar system bodies. Collectively, these clusters are called the Kuiper belt. Section 2.2 discusses the Kuiper belt in more detail.

Some ∼100 AU out from the Sun, the SW suddenly slows down from highly supersonic to subsonic speed. This sudden transition is called the *termination shock*, which is a direct result from the Sun's motion through the interstellar gas clouds. Both Voyager 1 & 2 have recently penetrated through the termination shock, establishing the location, shock strength, its dependency on the Solar activity (to some extent), and that it is flattened in the downwind direction. The Voyagers are now in a region called the Heliosheath, an analogy with the Earth's magnetosheath. The Heliosheath is simply defined as the region between the termination shock and the Heliopause. The high-speed gasses inside the Heliosheath still originate from the Sun's extended atmosphere (the SW), albeit deflected significantly from the purely radial direction. At the Heliopause, the deflection is maximum and flows parallel to the interstellar wind.

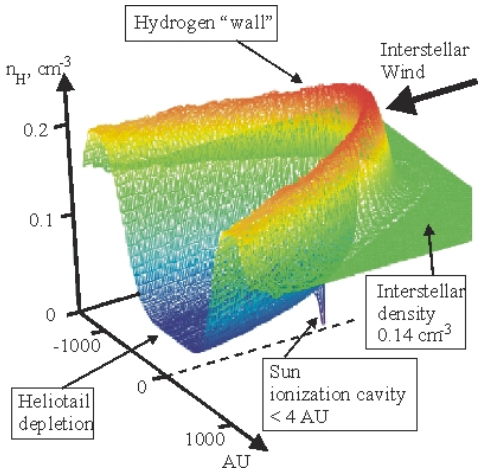


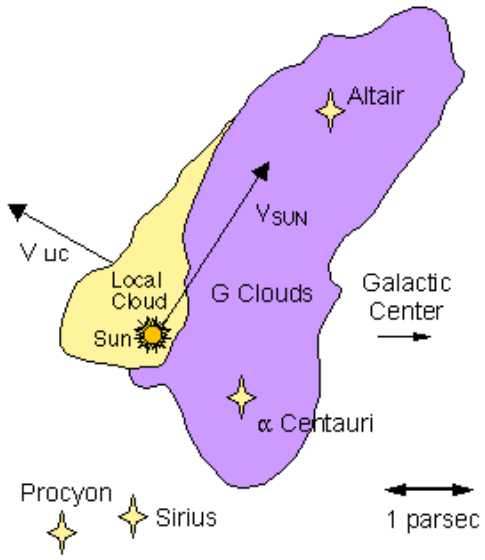
Figure 2.2 A simulated version of the hydrogen wall. Result from a calculation by Gruntman et al. [2001].

It is only after the Heliopause that true interstellar space begins. The Solar system still has not ended (the Sun still is the dominant gravitational body, and the Kuiper belt extends quite far out), but practically all gasses in the region are of interstellar origin. Due to the collision of the SW with the interstellar matter, there is a structural buildup of (interstellar) hydrogen atoms between the bow shock and the Heliopause. This leads to the so-called *hydrogen wall* (see Figure 2.2 for a slightly more detailed view). This buildup is the result of the instantaneous slowing to subsonic speed of the interstellar matter, which is thought to occur in the form of the SBS.

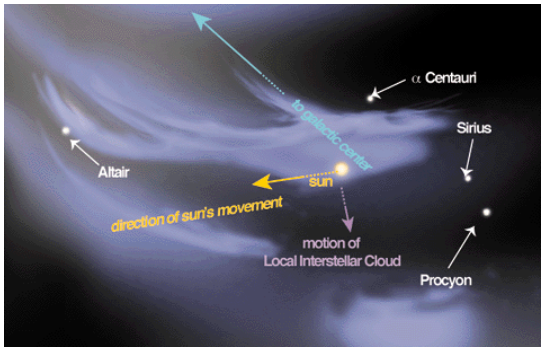
When viewed by some distant observer, the Solar system can be seen to lie inside a region of dust and gas called the *local cloud* or Local Interstellar Medium (LISM). The Sun is about 4 lightyears removed from the edge of this cloud, after which a particularly empty region of space begins. This area of relatively low density (and high temperature) is called the *local bubble*, in which the whole LISM and its neighboring cloud (the G-cloud) are completely embedded (see Figure 2.3). It is the unequal direction of motion of the Sun with respect to its local cloud that gives rise to all the phenomena described above – the Solar system and the LISM move in nearly perpendicular directions.

The whole heliosphere plus the most significant physics involved with the interaction between the SW and the LISM, is shown schematically in Figure 2.4. Actually, it is still uncertain whether there actually *is* an SBS (hence the question mark in the figure) – it has never been observed, and only appears in simulations which are built around uncertain data. Its existence is crucially dependent on the strength of interstellar magnetic fields. A shocks only appears if the supersonic medium has no way of communicating the presence of the other medium it collides onto to its surrounding material. Since $\sim$30% of the LISM is ionized, such disturbances *can* be communicated if a magnetic field is present – with the speed of light. The stronger this magnetic field, the smaller the probability that a shock actually exists. It is estimated that shock formation is completely impossible if the magnetic field is stronger than $(3.5 \pm 0.5)\mu$G. Current estimates on the *actual* strength range from 0 to 5μG.

Despite this (or perhaps, because of this), the SBS forms a very attractive mission target. Going there implies potentially visiting *all* bodies and phenomena of the outer Solar system that have not been previously explored. For most of these regions, only very indirect measurements have been made, and virtually nothing is actually *known*, let alone be observed *in situ*.



(a) Schematic view of the local cloud and the neighboring G-cloud system. The Sun is less than 4 ly removed from the edge of the local cloud. From [Mewaldt and Liewer, 1999].



(b) Three-dimensional view of the local cloud systems. From [Frisch, 2000a].

Figure 2.3 Interstellar clouds in the local neighborhood. The Sun is located in the local cloud, which borders the the G-cloud. Both are embedded in the "local bubble" – a region of very low density and high temperature.

2.1.2 Where Is The Solar Bow Shock?

Since the bow shock surrounds half the Solar system, it really does not matter in which direction the spacecraft flies as long as it is towards the proper hemisphere. However, considering the enormous distance to be covered (≈ 230 AU), the travel time should be kept to a minimum. Therefore, the main location of interest for this mission is the *stagnation point of the local interstellar wind*, because the SBS lies nearest to the Solar system there. As the name implies, this is the point where the relative Local Interstellar Wind (LIW) comes to a complete standstill due to its interaction with the Solar system.

The SBS, as seen from the Sun, lies in the direction where the radial speed of the LISM assumes its largest negative value. Obviously, this direction is equal to that of the relative velocity between the Solar system and the LISM. The determination of the direction of this velocity is however rather difficult. It is only thanks to the Ulysses spacecraft that we now actually know in which direction to look.

Solar Apex

One obvious way to determine the direction of the relative velocity between the Solar system and the LISM is by simply subtracting the Solar system's velocity from that of the LISM, both with respect to the galactic center. As discussed in Mihalas and Binney [1981, chap. 6] and Binney and Merrifield [1998, chap. 10.3], obtaining an accurate estimate on the Sun's direction of motion (the *Solar apex*) is quite difficult. The motion of the Sun with respect to

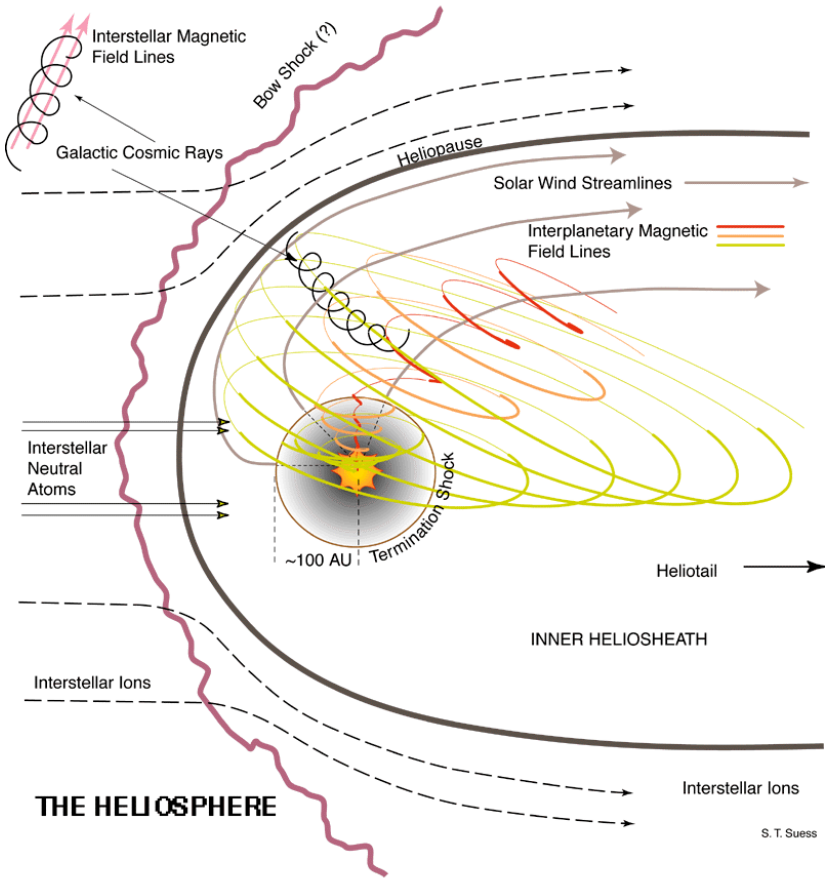


Figure 2.4 Overview of the Heliosphere and the various interaction phenomena between the Solar Wind and the interstellar medium. From [Mewaldt and Liewer, 1999].

its neighboring stars must be determined from the radial velocities of these stars with respect to the Sun. Since these radial velocities are partly composed of those stars' own motion, the measured radial velocities should be averaged. More importantly, the motion of older stars has been perturbed much more by the gravitational close encounters with other stars during their lifetime. As such, these older stars tend to have less coherency in their radial velocity.

Therefore, the exact position of the Solar apex greatly depends on the amount of stars used to average the radial velocities, and the spectral classes of the stars used in the calculation. The location of the Solar apex can therefore only be a weighted average of some specific data sets at best. The precise direction therefore greatly differs from source to source. One of the more reliable values for the Solar motion (largest data set) is ∼16.6 km/s towards $\alpha = 17^h : 49 : 58$ and $\delta = 28° : 07 : 04$ [Binney and Merrifield, 1998]. This puts the Solar apex in the constellation of Hercules.

Motion of the LISM

These values only give the direction of motion of the Sun with respect to its neighboring stars. Typical interstellar clouds can easily have (line-of-sight) velocities of $\sim$ 10 km/s, so

the direction of the relative velocity can easily differ by 30° (or more) in either direction. To obtain a more reliable value, also the velocity of the LISM must be determined.

Because the LISM does not emit (much) radiation itself, its properties must be determined indirectly. Usually this is done by looking at how it affects radiation from objects the cloud occludes. How the cloud affects radiation greatly depends on the constituents of the cloud, so before the (average) direction of motion of the LISM can be determined, an in-depth knowledge of these constituents of the LISM is required.

Radiation is most strongly affected by the electron density in the interstellar cloud. The electron density can be determined by looking at the dispersion of pulsar emissions and Faraday rotations of radio signals from more remote sources. Such measurements usually need to be averaged over distances in the order of ~ 100 parsecs to get a reasonably small error, but even with these measurements the best estimate for the electron density has as large an error as the measured value for it ($\sim (0.03 \pm 0.03)$ cm^{-3}). Errors of this magnitude confine the direction of the stagnation point within a cone of $\sim 12°$. Until 1972, this was the best estimate known of the location of the stagnation point.

Direct Measurements on Neutral Particles

A completely different way of determining the relative velocity of the Solar system with respect to the LISM, is by measuring the flow direction of Energetic Neutral Atoms (ENA's). As mentioned previously, a small fraction of the LISM consists of electrically neutral particles (He, H_2, n, ...). The (average) motion of these particles with respect to the center of the galaxy is of course equal to that of the LISM as a whole, but the neutral particles are interesting for a different reason.

As the (charged) material of the LISM enters the Solar system, the direction of motion of its particles is affected by the shocks arising from the collision of the two highly supersonic flows, and by the interaction of the interstellar and Solar magnetic fields. Therefore, the few interstellar particles that penetrate the Heliosphere, generally do not arrive at Earth from a uniform direction.

This is however not true per se for *neutral* particles. Their motion remains completely unaffected by the presence of magnetic fields, which implies that the (average) direction of such particles can be translated into the direction of the local interstellar wind. Thus, the direction of the local interstellar wind, or equally, the relative velocity of the Solar system and the LISM, can be found from direct measurements on energetic

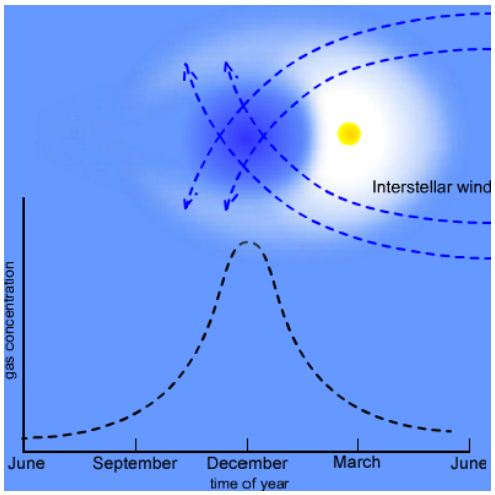


Figure 2.5 Neutral particles deflected by the Sun's gravity. Adopted from [ESPG, 2009].

neutral particles flowing through the Solar system.

These ENA's are primarily hydrogen or helium atoms. The forces acting on a neutral *hydrogen* atom approaching the Sun, are gravity and Solar radiation pressure. A fraction of the incoming neutral atoms is "lost" as a result of charge exchange with the SW protons. Because the intensity of the SW increases with the square of the heliocentric distance, the probability of such charge exchanges strongly increases with decreasing distance to the Sun. The probability for a wind speed of ~ 20 km/s becomes >95% at around ~ 4 AU out.

This charge exchange can convert a fast SW proton into a fast (neutral) hydrogen atom. The proton of interstellar origin that is left over, is thus "picked up" by the interplanetary magnetic fields – these atoms are referred to as Pick-up Ion (PUI). These PUI's have an interesting fate – after they are picked up, they are swept out to the termination shock. There they are accelerated by an unknown process to very high speeds, and hurled back into the Solar system. Near the Earth these accelerated PUI's are observable as *anomalous cosmic rays* – cosmic rays with no obvious origin, and with too little energy to have entered the Solar system on their own accord.

The Solar radiation pressure acting on interstellar *helium* atoms has negligible effects on their motion – they stay neutral, and their motion is dominated by the Sun's gravity. As such they can penetrate to ~ 1 AU or less, providing an easily accessible way to determine the direction of the local interstellar wind. In Figure 2.5 the interaction between the Sun's gravity and the interstellar neutral helium is shown, as well as a (simplified) graph of the measured intensity of He-radiation.

Such measurements have been performed by the Ulysses spacecraft. As found in the analysis by Witte et al. [1996], the LIW has a velocity of (25 ± 2) km/s, in the direction $[\lambda, \beta] = [75.4°, -7.5°]$ (ecliptic longitude/latitude). These results have subsequently been corroborated by additional data from the SOHO, ACE/NOZOMI, and EUVE spacecraft, and by direct observation of PUI's by the AMPTE spacecraft, and more recently, by the first measurements on ENA's by the Interstellar Boundary Explorer (IBEX).

As the direction of motion of the Solar system and LIW only changes notably over the course of thousands of years, the location of the SBS with respect to the Sun can safely be assumed constant throughout the current mission's lifetime. This can be translated into the *Cartesian coordinates of the SBS*, which can be used for first-order analyses. Assuming the SBS lies $R_{\mathrm{SBS}} = 200$ AU from the Sun and the direction $[\lambda, \beta] = [75.4°, -7.5°]$ is exact, the Cartesian coordinates of the SBS are

$$\begin{aligned} x_{\mathrm{SBS}} &= R_{\mathrm{SBS}} \cos\beta \cos\lambda &&\approx 50 \text{ AU} \\ y_{\mathrm{SBS}} &= R_{\mathrm{SBS}} \cos\beta \sin\lambda &&\approx 192 \text{ AU} \\ z_{\mathrm{SBS}} &= R_{\mathrm{SBS}} \sin\beta &&\approx -26 \text{ AU} \end{aligned} \tag{2.1}$$

These coordinates define a point which will serve as the primary target for all first-order or preliminary tests. Witte et al. [1996] gives uncertainties $[\Delta\lambda, \Delta\beta] = [0.5°, 0.5°]$, which translates into an error-circle about the above estimate with a radius of approximately 2 AU. Moreover, as mentioned earlier this chapter, the stagnation point is only an *indication* of

where the shortest distance to the SBS can be expected, but of course, any real mission is not limited to this single point, since the SBS surrounds a good half of the Solar system. Therefore, for subsequent tests, both these angles will be varied by the original 1972 error-estimate of 12°, to see how the quality of the outcomes are influenced by slightly different target points near the stagnation point.

2.2. Minor Planets

This section serves as a general introduction to Minor Planets (MP's). Much of the minor planets' general history has been collected in a very amusing popular book by Schilling [2008]. A good overview of the Kuiper belt can be found in [Delsanti and Jewitt, 2006]. Most of the material in this section comes from these two sources, or references therein.

2.2.1 Orbital Groupings

MP's are far too small to have a significant impact on the orbital evolution of the planets[1]. Rather, the gravity of all the planets in the Solar system dictate the orbital evolution of any MP. Due to a planet's gravitational pull combined with the planet's orbit around the Sun, there are regions in space (or rather, specific orbital elements) that are *destabilized* due to the periodic forcing of this perturbing gravitational force – this phenomenon is generally referred to as an *orbital resonance*.

It is thought that when the Sun was formed, the rest of the nebula out of which it was born conglomerated into planetesimals – relatively large, irregularly shaped rocks. These planetesimals were originally abundantly spread throughout the inner Solar system, but later formed larger clusters (that eventually formed into planets), the combined gravitational force of which caused many of the others to be ejected to the outer Solar system. There are however many regions where this *didn't* happen (or, where the orbital resonances *stabilized* their motion), which gave rise to several clusters of remnant planetesimals.

The most well-known of these groups is the Main Asteroid Belt (MAB) – a group of asteroids between ~ 2 and ~ 4 AU. The gravitational perturbations caused by massive Jupiter prevented these planetesimals from ever forming a planet, although it is believed that originally the total mass of the MAB amounted to about that of the Earth. Today, due to the destabilizing orbital resonances from Jupiter, about 0.1% of that mass is left, which is about a quarter of the mass of the Moon. Although these small bodies (typically ∼10-100 km, with a few exceptions like Ceres, which is ∼438 km) started as planetesimals, they have had a long and eventful history. Due to their relatively close presence to each other, collisions are not uncommon. Over time, these collisions have caused severe physical changes in their internal mass distribution and surface composition. For this reason these bodies are now called *asteroids*, to distinguish them from true, "old" planetesimals.

[1] This is only partly true – *en masse*, and on longer time scales, minor planets can indeed have a significant impact. It has been shown that it is probable that the Solar system's gas giants were created much closer to the Sun than they are today. They migrated outward over the course of billions of years by hurling millions of minor planets to the outer Solar system.

A particularly interesting subgroup in the MAB[2] are the Jupiter Trojans (and Greeks), located at the stable 1:1 resonances (L_4 and L_5 points) separated 60° from Jupiter in its orbit. Although it is generally thought that these asteroids are left-over planetesimals like the members in the MAB, their presence and orbital characteristics can actually better be explained by assuming they were formed in the outer Solar system and were subsequently captured chaotically by Jupiter in the early Solar system, as shown by Morbidelli et al. [2005]. As such, their entire history is probably completely different from any other member of the MAB, which makes them very interesting secondary mission targets. As an even more interesting side note, in the last decade several Trojans have been discovered in Mars' and Neptune's orbit – 4 Martian Trojans and 6 Neptunian Trojans are currently known [Smithsonian Astrophysical Observatory, 2009, link to "Lists and Plots"]. Theories surrounding *their* origins are even more tenuous; so far it has only been shown that those regions are stable enough to keep an asteroid population for longer periods of time (which was the original reason to go look for asteroids there).

In the debate over the location of the hypothesized *planet X*, it had been suggested by various astronomers that the perturbations in Neptune's orbit could be caused by a massive asteroid belt beyond Neptune's orbit. This idea was strengthened when many more short period comets were discovered than could be explained by the Oort cloud alone – a comet belt beyond Neptune would be far more likely a source for so many comets than something as remote as the Oort cloud. Originally started (and subsequently hindered) by the discovery of Pluto, and sparked by the discovery of the object 1992 QB_1 ("CuBeWano's"), the search for more objects belonging to this belt was started.

Today, some ∼1000 such Trans-Neptunian Objects (TNO's) have been discovered. Many of them (including Pluto) are found to be clustered around the 3:2 resonance with Neptune (the *Plutino's*) around $a = 39.5$ AU. Many others are found around the 5:3, 7:4, 2:1 and 5:2 resonances, all between 39.4 and 55.8 AU (see Figure 2.6). Collectively, this belt is called the *Kuiper belt*, or Edgeworth-Kuiper belt (EKB)[3]. These bodies are thought to consist primarily of frozen volatiles (methane and other organic compounds, ammonia, water, ...), unlike the more rocky/metallic asteroids found in the MAB. More importantly, aside from some surface corrosion due to the very long-term exposure to cosmic rays and Solar (ir)radiation, they are mostly unaltered since the formation of the Solar system – they are the "living fossils" of the

[2]There seems to be some disagreement whether the Jupiter Trojans are part of the MAB. They lie relatively close to the MAB so that they frequently show up in most images of the MAB. However, in much of the literature it seems that they are not considered to be part of the MAB and are often treated as a group on their own.

[3]It is highly amusing to read about the "soap-opera" surrounding naming conventions of the minor planets. For example, there has been (and still is) much debate about the proper name of the Kuiper belt; many scientists say that neither Kuiper's nor Edgeworth's name should be given to the belt, but that this honor should belong to Fred Whipple for his correct prediction of the belt. Some scientists refuse to adopt any such naming convention and simply call everything from this belt a trans-Neptunian object, although that term is used for everything that is *not* part of the Kuiper belt by others, leading to much confusion. The most famous clash of the naming conventions is the demotion of Pluto in 2006 to "dwarf planet" (followed by a true rebellion that aimed to undo it).

original planetesimals that formed the planets[4].

One important remark is in place. Dynamical models suggest that there should be a gradual *increase* in the number of objects beyond ∼50 AU, but it appears that after the 1:2 resonance limit (∼46 AU) there is a drastic falloff in the amount of objects (the *"Kuiper Cliff"*). This has been determined to be a real phenomenon and not due to measurement bias, and there is currently no model that can explain why this mass defect exists.

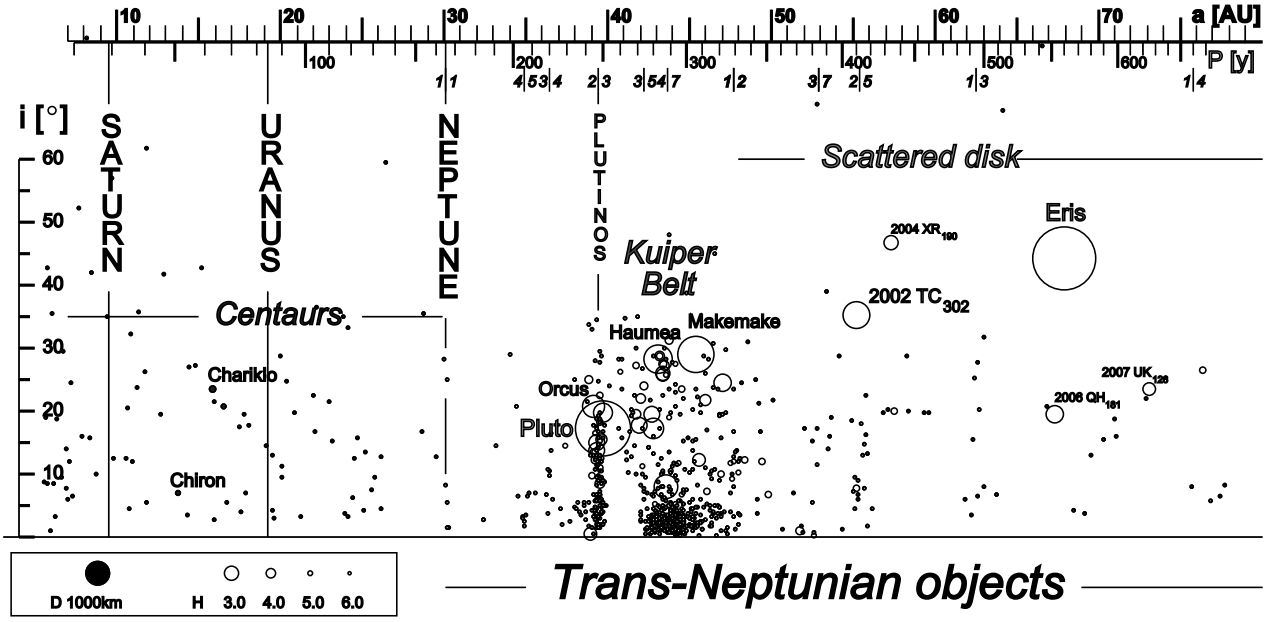


Figure 2.6 All known Cis-Neptunian and Trans-Neptunian objects, plotted semi-major axis versus inclination. The size of the circles represent the object's brightness, which serves as a direct indication of its size. The location of the Centaurs, Kuiper belt and Scattered disc are also indicated, as well as the locations of the orbits of Saturn, Uranus and Neptune for reference. From [Wikipedia.org, 2009e].

Some *very* remote objects have been found even beyond the orbits of the outermost members of the Kuiper belt. The best known of these was the first one to be discovered, the MP Sedna – with its perihelion at 75 AU and aphelion at 850 AU it instantly became the most remote Solar system object known. Since its discovery many more such "anomalous" objects have been found, that do not seem to be part of any stabilizing orbital resonance, and are much too far away to have been formed there. Such objects are said to be part of the *Scattered Disc*, or Scattered Disc Object (SDO), since it is thought that these objects have been transported outward by the migration of the outer planets in the early Solar system.

An interesting exception is the collection of minor planets called the *Centaurs*. They are objects that orbit between (and sometimes cross) the orbits of the planets beyond Jupiter.

[4]There is however some debate over this issue – there is not enough mass in the Kuiper belt to explain stable accretion to bodies as large as Pluto or Eris. This can only be accounted for if the belt was created much closer to the Sun and underwent massive collective outward migration, or, some unknown mechanism is ejecting bodies out of the EKB.

All their orbits are unstable, so that some mechanism must be in effect that regularly replenishes their population. As such they are also designated as SDO's[5], because they are thought to have been migrated inwards instead of outwards. What makes them most interesting is their dual nature – at least three of them are known to exhibit cometary comas, while their size suggests they are more like asteroids. The mechanism driving these comas is highly debated, because it is quite unexpected – they are simply too far from the Sun (between ∼5.5 and ∼30.1 AU) to be affected enough by its heat, and the objects are simply too large to be heated enough by micro-meteorite impacts.

2.2.2 Minor Planet Center

The Minor Planet Center (MPCe) was created in 1947, to answer the need for a central body dealing with MP's. It is located at the Smithsonian Astrophysical Observatory and runs under the auspices of the International Astronomical Union. The most important activity of the MPCe is the collection, organization and publication of the known data on all known MP's. All these data are listed in the Minor Planet Center Orbital Database (MPCORB), which is freely available to the general public (see [Smithsonian Astrophysical Observatory, 2009]). Newly discovered bodies are added to this database almost daily, and new orbit solutions for all known minor planets with improved observational accuracy, are published online about every week. The MPCORB database is published in plain-text format. As of Jan 21st 2010, the MPCORB contains data on 398,999 MP's. Because the MPCORB is the most frequently used source by astronomers in this field, it will also function as the primary source of MP-data for this research.

[5]This classification is upheld by the Minor Planet Center. It is not fully accepted in the scientific community, however – various other institutions use different classifications for Centaurs.

3
Mission Heritage and Outline

The concept of sending a probe to the SBS is not entirely new. This idea to explore the edge of the Solar system and beyond stems from a long history of earlier missions and the conflicts with leading theories their results have stirred up. Also, the idea to go to MP's is not new – that too has been done many times before. However, the combination of these two ideas is new and untried. It is therefore essential to place this new idea in the proper context, and define exactly what the goals of this mission are. This is the purpose of this chapter.

The mission's context and framework are provided in section 3.1. This section briefly looks at the results from past missions that are comparable in some respects, as well as compare the general ideas for this mission to more recent research on the subject. Then, a broad overview of the mission's goals, aims and requirements is given in section 3.2. Building on the discussion in the previous chapter, this section clarifies what aspects of the Solar bow shock and other parts of the Solar system the current mission is to investigate, and states those goals clearly and unambiguously. Finally, in section 3.3, all the required work is broken down into small manageable parts, and a list of the initial assumptions necessary to start designing the mission is presented.

3.1. Mission Heritage and Other Related Work

3.1.1 Outer Solar System

The Pioneer 10 & 11 and Voyager 1 & 2 spacecraft are the only spacecraft ever to have reached further into space than Pluto's orbit. Sadly, both Pioneer spacecraft stopped functioning before they could reach beyond the planetary region. However, both the Voyager spacecraft are in relatively good health, and were quite recently shown to have pierced through the termination shock [Stone et al., 2005]; this makes these three decade old missions still relevant for the current mission.

Inspired by the Voyager results, NASA has launched a new small probe (IBEX) into high Earth orbit, which is currently exploring the Heliopause and termination shock via indirect means [Stoyanova and Dunbar, 2009]. Aside from several proposals in the literature, these two missions constitute the entire history of exploration of the outer Solar system.

Voyager 1 & 2

Arguably the most well-known space probes ever to have been launched are the Voyager 1 & 2 spacecraft. This is because no other spacecraft ever explored so many unknown bodies as the Voyager spacecraft did; before and since. They were launched in a time when the world's fastest computer was larger than a decent apartment and less powerful than a mobile phone today. As such, the entire field of trajectory optimization was virtually non-existent at the time, which makes their achievements even more impressive.

The Voyager spacecraft could visit all the outer planets because of a very fortunate alignment of the planets (see Figure 3.1). This alignment occurs only once every 176 years; a very fortunate coincidence indeed. Today they are ∼100 AU removed from the Earth, and are increasing that distance with ∼ 3.5 AU per year (∼60,000 km/h).
They were not the first probes to visit the outer Solar system however; the Pioneer 10 & 11 spacecraft both predate the Voyagers. Current estimates indicate that Voyager 1 and Pioneer 10 hold the record for the two farthest removed man-made objects (see Figure 3.2). However, both Pioneer spacecraft have long since failed and can no longer be used. Both Voyager spacecraft are the only probes in the outer Solar system that are still collecting scientific data.

In December 2004 the Voyager 1 spacecraft made a major breakthrough in the exploration of space – it pierced through the Solar system's termination shock [Stone et al., 2005]. Later, in August 2007, also the Voyager 2 spacecraft penetrated the termination shock, much earlier than expected [Angrum et al., 2007]. This fact alone proves that the Solar system's shape is much more "squashed" than has previously been assumed.

In contrast to Voyager 1, the Voyager 2 spacecraft crossed the termination shock at least 5 times over the course of a couple of days due to "sloshing" of the shock. Also in contrast to Voyager 1, the Plasma Science instrument onboard Voyager 2 spacecraft was still fully functional. These facts allowed the spacecraft to perform many more valuable measurements on the shock than its twin. The processed and interpreted data shows that the shock does not behave as expected – the measurements clearly show a much cooler temperature behind the termination shock than expected values for the measured wave strength. This is most likely due to the energy transfer from SW-kinetic to the high-energy anomalous cosmic rays, although the mechanism is still unclear [Angrum et al., 2007].

The Voyager missions were designed primarily to explore the Jovian and Saturnian systems. The additional flyby of Neptune and Uranus by Voyager 2 was a "last-minute bonus", since at launch these flyby's were not part of the mission plan. As planetary missions, they were

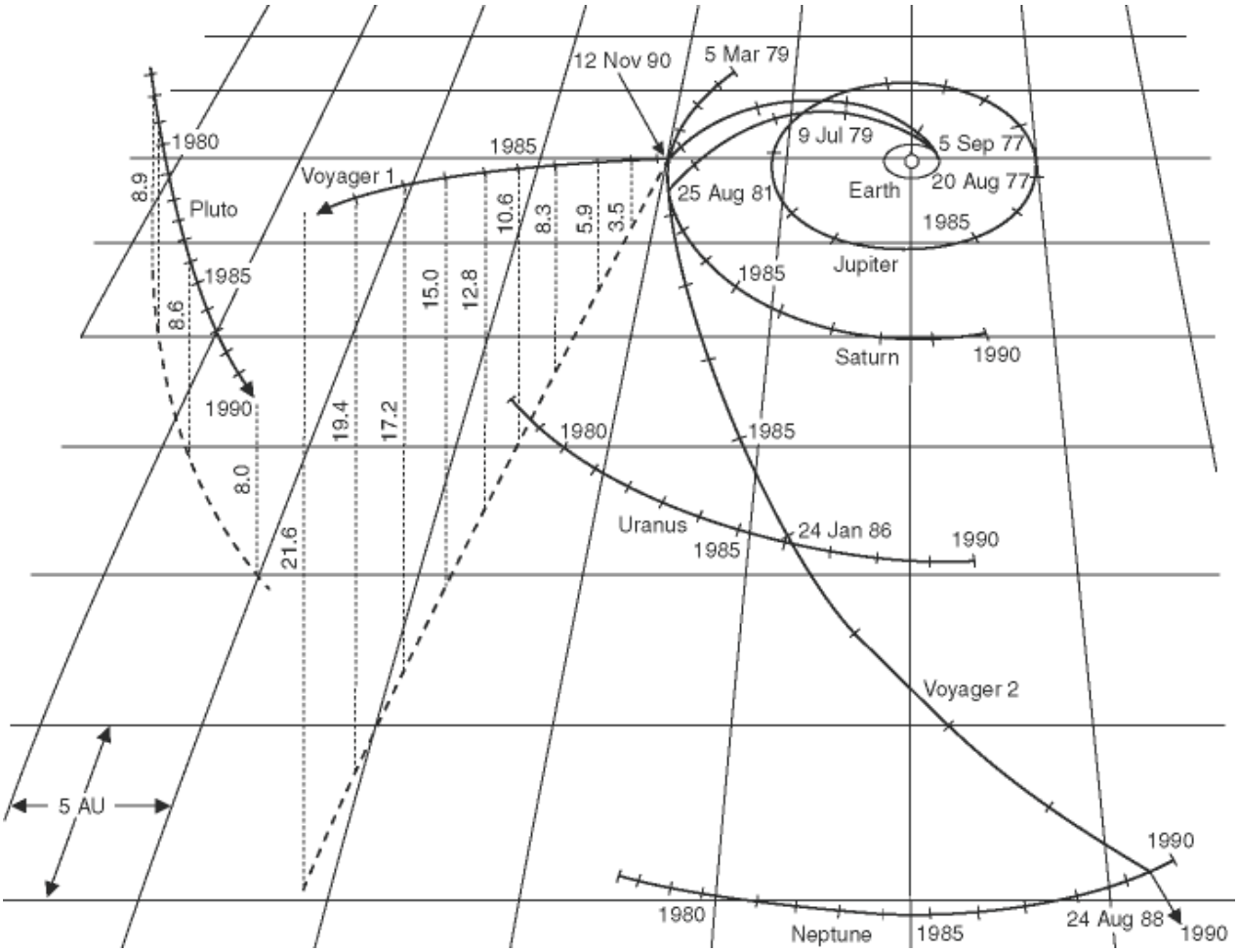


Figure 3.1 Three-dimensional view of the trajectories of both Voyager spacecraft. The tick marks on all orbits and trajectories indicate one year. From [Russell, 2008].

never designed to explore interstellar space[1], as their trajectory clearly indicates (∼30 years to cover ∼100 AU is far from optimal in that respect). Currently it is expected that Voyager 1 (the farthest removed of the two) will cross the Heliopause and enter true interstellar space in 10 to 20 years. However, the analysis of household data collected throughout the probe's life predicts that most of its instruments will start to fail around 2015, and the probe will certainly die completely before the year 2025 [Angrum et al., 2007, Spacecraft Lifetime]. As such, reaching the Heliopause with a functioning Voyager spacecraft would be yet another fluke. But it is absolutely certain neither of them will ever reach the SBS while still functioning.

IBEX

A much less direct way of observing the interstellar neighborhood, is by observing the energy and direction information embedded in the ENA's flowing into the Solar system. This is precisely the idea behind NASA's IBEX [Stoyanova and Dunbar, 2009]. The IBEX mission is

[1]The interstellar mission extension was again a fluke – funding for this was not planned (or even conceived for that matter) at the time of launch. Only after their many successes did funding for the mission extension come available, procuring even more mission firsts (albeit on a somewhat longer timescale).

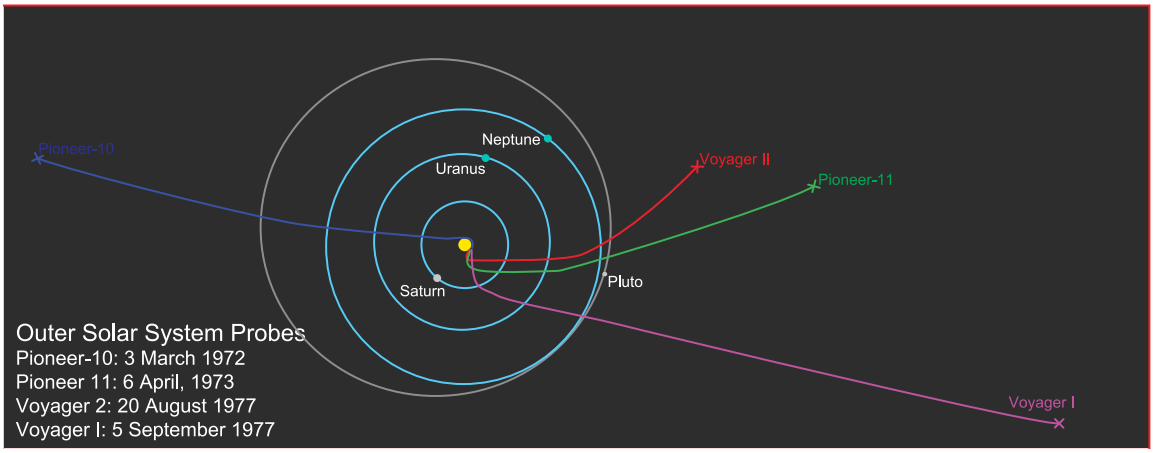


Figure 3.2 Current positions of all probes that ever left the Solar system's planetary region. Adopted from [Wikipedia.org, 2009f].

the latest in NASA's Small Explorers spacecraft series – a series of spacecraft optimized for cost and short development time.

The IBEX spacecraft orbits the Earth in a highly elliptical orbit ($r_a \approx 200,000$ km, $r_p \approx 7000$ km), to achieve a long stay time outside of the Earth's magnetic field. This is necessary, because the interaction between the Earth's magnetic field and the SW creates the same sort of ENA's the spacecraft is designed to observe [McComas et al., 2009]. The spacecraft observes the ENA's with two single-pixel cameras. It relies on the spacecraft's spin and precession of its orbit during one year, to cover the full sky (see Figure 3.3). The spacecraft's spin-axis is adjusted every orbital period (∼8 days) such that it always points at the Sun.

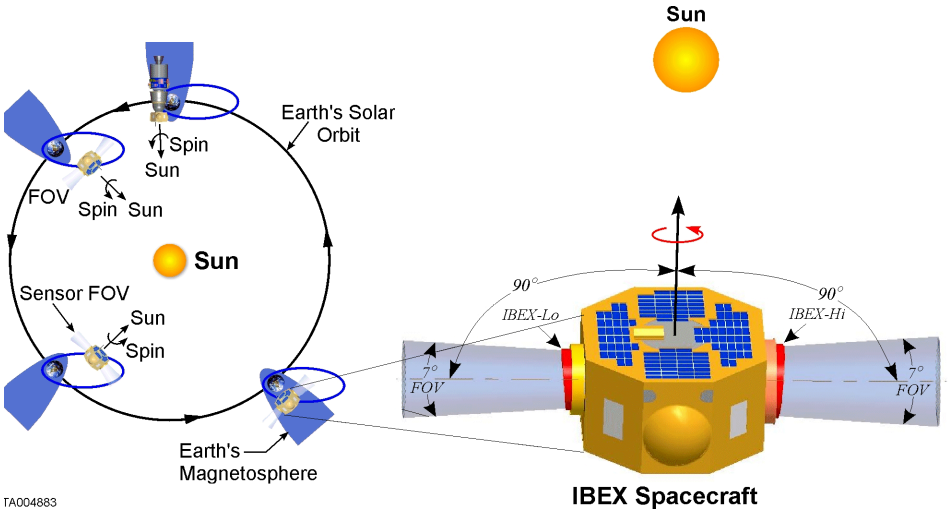


Figure 3.3 Overview of the IBEX mission. The spacecraft is equipped with two ENA-cameras; one observing the high-end of the neutral atoms energy spectrum (IBEX-Hi), and one observing the lower end (IBEX-Lo). The full sky is observed about twice a year due to the changing spin direction and precession of the spacecraft's orbit during the year. From [McComas et al., 2009].

The main science questions to be addressed by IBEX are [McComas et al., 2009]:

1. What is the global strength and structure of the termination shock?
2. How are energetic protons accelerated at the termination shock?
3. What are the global properties of the solar wind flow beyond the termination shock and in the Heliotail?
4. How does the interstellar flow interact with the Heliosphere beyond the Heliopause?

Thus, the scientific objectives for the IBEX mission are closely related to the current mission. If successful in (partially) answering these questions, many of the *global* properties of the Heliosphere beyond the termination shock will be clarified. However, IBEX can teach us nothing about the hydrogen wall, nor about the SBS.

In a press release issued in August 2009, the IBEX team announced they had processed the spacecraft's full-sky data. The results they found indicated that the ENA's seem to emanate from a narrow band surrounding the Solar system. This was a quite unexpected result, as previous models predicted that these ENA's *should* come from all directions, and only slightly faster near the upwind direction. This result has only very recently been explained by incorporating a "magnetic mirror" effect into the heliosphere model [Stoyanova and Dunbar, 2009]["mission news" section].

If anything, these events again indicate that there is still much work to be done in this area. IBEX's further operation depends mainly on whether the mission will continue to receive funding now that its main data products have been delivered, so it is still uncertain whether IBEX is able to address the time-dependent qualities of these phenomena. But in light of these recent unexpected results, this is fortunately quite likely to be the case.

Interstellar Probe Mission

In 2002, A small team at NASA/JPL did a Phase A study on the possibilities for an interstellar mission. The outcome of that project was the *Interstellar Probe* (IP), described in full in a technical report. The mission was unfortunately never realized due to technical limitations and budgetary constraints. The mission designers assumed a very large (200 m radius) light weight (~ 1 g/m^2) Solar sail as the probe's main means of propulsion, a means which is still in the highly experimental stages and would be far too risky to use on a near-future flag-ship mission. Also, because of the Solar sail, the IP was designed to have a total mass of less than $\sim$250 kg, including the sail and its support. Such an extremely low mass would only leave room for a few miniaturized and highly specialized instruments. Indeed, the straw man instrument set was specifically suited to measure the effects of the interactions between the SW and the LIW, and determine the properties of the interstellar space beyond the bow shock, but not much else. This implied the IP would not produce *any* useful data for at least 15 years after launch – the expected travel time to the SBS, if all would go well.

Fortunately, a shortened version of the report is available online [Mewaldt and Liewer, 1999]. The IP's mission requirements and scientific objectives were practically the same as for the

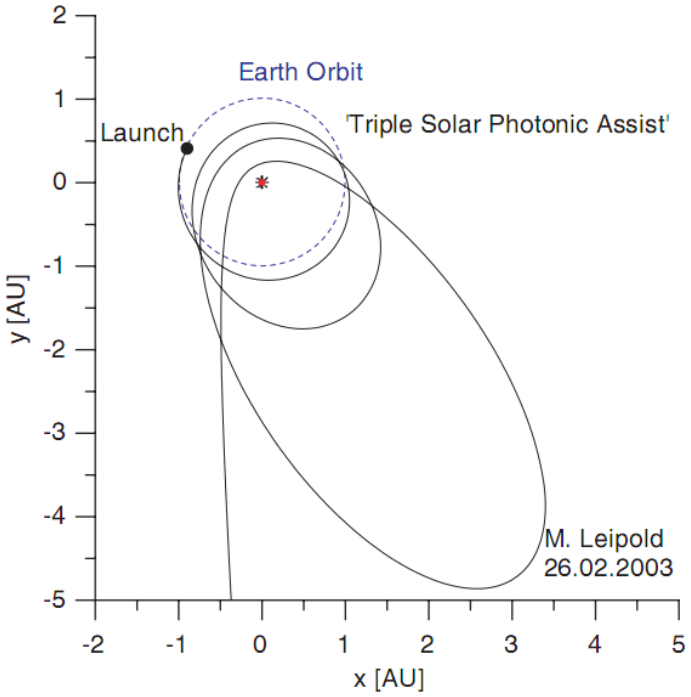


Figure 3.4 Multiple "photonic" assist trajectory. This was found to be an optimal trajectory for the Heliopause Explorer reference study. This is Figure 9 from [Leipold et al., 2005].

current mission's primary target, so that these can (partly) be copied to the current mission (see section 3.2.2). However, what makes the IP most relevant to this mission is its trajectory towards the SBS. The IP design managed to cover the 200 AU leg in about 15 years, which implies an *average* speed of 200 AU / 15 years $\approx$ 63.5 km/s – quite an achievement indeed. It accomplished this by approaching the Sun very closely ($\lesssim 0.25\ AU$), increasing both the radiation flux on the sail (and thus the thrust generated) and the efficiency of this acceleration (any acceleration will result in a greater speed if it is applied closer to the central body, see appendix A).

The IP was the first of several such interstellar precursor missions developed at NASA. A very comparable study has been done by Leipold et al. [2005], resulting in the Heliopause Explorer (HE) design. This mission relied on an even larger Solar sail (245 $\times$ 245 m) and a close approach to the Sun, but tried to reduce the mass and required shielding by performing several close-perihelion orbits before turning the trajectory hyperbolic (see Figure 3.4). This particular mission was also used as the reference mission in a technology reference study done at the European Space Agency (ESA), by Lyngvi et al. [2005]. This report indicated that the primary "pains" for these interstellar precursor missions are indeed the Solar sail, the required level of onboard autonomy, and communications.

These interstellar precursor missions will be used in several ways for the current research. As mentioned, the mission requirements and science objectives are identical for these missions and can be re-used. Moreover, all these missions identify a Solar sail as the *only* means of achieving a distance >200 AU in less than 20 years – a statement which seems to be based

on intuition rather than sound research. This statement was indeed challenged in [Walker et al., 2006], and will be challenged further in this research. Therefore, the travel times and average speeds the IP and HE are based on will serve as "numbers to meet if not beat" by the current mission.

3.1.2 Missions to Minor Planets

There have been many missions to MP's to date. Famous examples include the multi-national five spacecraft Halley Armada (to Halley's comet), the Japan Aerospace Exploration Agency (JAXA)'s Hayabusa (to asteroid Itokawa), NASA's Deep Impact (to comet 9P/Tempel) and ESA's Rosetta (to comet Rosetta67P/Churyumov-Gerasimenko and two asteroids). Most of these missions are not worth considering in great detail here since those missions are quite unrelated to the current mission, both in mission target and orbital design. There is however, one exception: NASA's *New Horizons*, which will be discussed briefly in the next section. The following section will shortly review the most relevant results from all these missions.

New Horizons

At the start of this millennium, the only planet in our Solar system which remained completely unexplored by spacecraft was Pluto. The very high energy demands for a mission to Pluto (due to its large distance and unfavorable out-of-plane position) made that such a mission has very high budgetary requirements. It is mostly this fact which led to the cancellation of various proposals in the past, and it was not until the late 1990's that after considerable pressure from the Planetary Society, mission planning for *New Horizons* began.

A NASA initiative, this mission is to explore the "last planet" in our Solar system[2], as well as one (or hopefully, several) Kuiper belt object(s). It is expected the spacecraft reaches the Pluto/Charon system halfway through 2015. The trajectory is designed as to perform a flyby at Pluto, with a closest approach of ∼13,0000 km. Also, Pluto's moons Charon, Hydra and Nix will be flown by, with a closest approach to Charon of approximately 29,000 km. The spacecraft is equipped with a visible, infrared & ultraviolet imager/spectrometer, a telescopic camera, a passive radiometer which also allows radio science experiments, SW, plasma & energetic particle spectrometers, and a student-developed dust counter. This instrumentation set allows it to map nearly all of Pluto's surface at many wavelengths, investigate various aspects of its thin atmosphere, and the interactions of Pluto with its surroundings [Dunbar and Boone, 2009].

If all goes well, New Horizons will continue its flight into the Kuiper belt. Investigations are currently underway to find suitable flyby candidates; at this time it is not yet known which KBO's can be flown by. Unfortunately, the spacecraft's maneuvering capabilities are quite limited, which rules out the original plan of performing a flyby at Eris. The search for

[2]Shortly after its launch in 2006, the IAS re-classified Pluto as a *dwarf planet*, much to the dismay of the mission designers and investigators. Officially, New Horizons has thus degenerated into a mission to an MP. Naturally, the mission's PI (Alan Stern) actively defies this re-classification and still regards Pluto as the Solar system's 9^{th} planet.

good candidates is hindered by the fact that the accessible region lies quite close to the plane of the Milky Way, complicating surveys for dim KBO objects in that area.

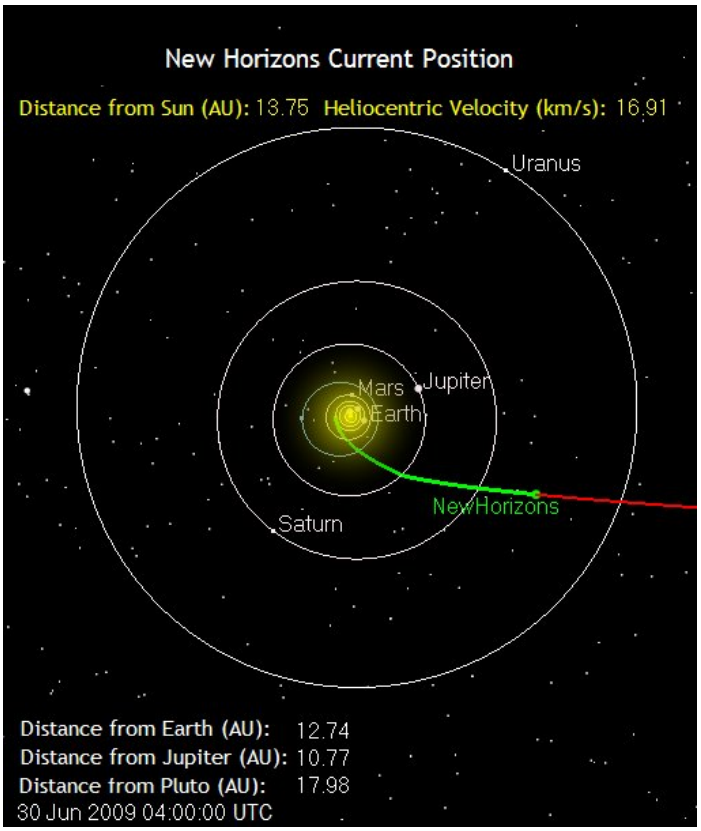


Figure 3.5 The current position of the New Horizons probe with respect to all the planets. From [Wikipedia.org, 2009b].

Aside from it being a very interesting mission in its own right, its results are not very relevant to the current mission and can in no way have influence on this mission's requirements. What makes New Horizons worth mentioning here however is its *trajectory*. The probe's current approximate position and planned trajectory are shown in Figure 3.5, along with the positions of the planets. Note that most of its kinetic energy came from the Gravity Assist Manoeuver (GAM) at Jupiter. The mission's trajectory is most notable for the fact that it was one of two probes that have ever been launched *directly* to Jupiter, that is, without first performing GAM's at planets in the inner Solar system. This hints at the enormous launch capacity that is possible with today's launchers, a fact that can possibly be used in the design of the current mission. Most importantly, even with this unique feature, New Horizons still falls in the *New Frontiers* mission category, which are generally smaller and less expensive than *Flagship* missions.

Results from All MP-missions

The list of all MP's that have ever been visited by spacecraft is quite short – short enough to be listed in Table 3.1. Many of these bodies were not even main targets for the spacecraft

that visited them – they just happened to be in the right place at the right time[3].

Table 3.1 A list of all minor planets ever to have been visited by spacecraft. Here, $d_{\min}$ indicates the minimum flyby distance to the listed body. From [Doblas et al., 2009] and various related web sites.

(a) Asteroids

Name	Category	Family	Spec. Type	$d_{\min}$ [km]	S/C
132524 APL	MAB		S	101,867	New Horizons
5535 Annefrank	MAB	Augusta	S	3,079	Stardust
9969 Braille	MAB	Mars Crosser	Q	26	Deep Space I
433 Eros	MAB	NEA	S	∼0	NEAR Shoemaker
951 Gaspra	MAB	Flora	S	∼1,600	Galileo
243 Ida+Dactyl	MAB	Koronis	S	2,390	Galileo
25143 Itokawa	MAB	Apollo	S	0	Hayabusa
2685 Masursky	MAB	Eunomia	S	∼1,600,000	Cassini/Huygens
253 Mathilde	MAB		C	1,212	NEAR Shoemaker
2867 Šteins	MAB		E	800	Rosetta

(b) Comets

Name	Category	$d_{\min}$ [km]	S/C
19P/Borrelly	Short period	2,253	Deep Space 1
C/2006 P1	Non-Periodic	257,495,000	Ulysses (through tail)
21P/Giacobini-Zinner	Short period	?	International Cometary Explorer
26P/Grigg-Skjellerup	Short period	200	Giotto
1P/Halley	Short period	596	-many-
9P/Tempel	Short period	-0.03	Deep Impact
81P/Wild	Short period	300	Stardust (through tail)

The most detailed and significant results are due to the Stardust, Hayabusa and Deep Impact missions, which were all indeed dedicated missions. The short-period comets 9P/Tempel and 81P/Wild (impacted by Deep Impact and sampled by Stardust respectively) provided a very direct look at composition of the surface and subsurface. Stardust's samples were found to contain many different organic compounds. These organics were in turn found to be very rich in oxygen and nitrogen compared to meteorites found on Earth. Also, there was an excess of deuterium and Nitrogen-15, hinting that some of the samples have interstellar (or perhaps proto-stellar) origins [Sandford et al., 2006]. Research on these samples is still underway.

In conclusion, to come to a more complete understanding of both the origins and future of the Solar system, many more MP's will have to be explored. Several other space missions are currently underway (Rosetta, Dawn, New Horizons), but surprisingly, New Horizons is the only spacecraft that will intentionally perform a flyby at a KBO; the other missions will only investigate Main Belt members. Although Ceres (one of Dawn's targets) has spectral type CG-class, it will comprise the only non-S-class asteroid that will be explored in the near future. It would however be quite interesting to also explore some of the other classes; M, E,

[3]Asteroid flyby's are however part of the design process for most newer missions.

or P-types (metallic) or the relatively rare A, D, T, Q, R, and V-types. Quite little is known about them, making them ideal targets for the current mission.

3.2. Current Mission

3.2.1 Justification

The interaction of the Sun with its surroundings is hardly understood, mostly because its "surroundings" are so far away and therefore hard to observe and quantify. It is nevertheless quite interesting (not to mention important) to get to know more about this particular part of the grand mechanism that protects the Earth from the perils of the universe. Indeed, the popularity of the Heliosphere and the outer Solar system in general as study objects have steadily increased since the Voyager spacecraft pierced through the termination shock – the existence of the IBEX mission is a small silent witness of that fact.

It is hard not to notice that the list in Table 3.1 is surprisingly short. With over 400,000 MP's known to exist, it would seem rather easy to come close to, explore and map many more of them, even in a single mission. The idea of a multi-asteroid mission is gaining in popularity, partially fuelled by the promising results of past Global Trajectory Optimization Competitions (GTOC's), albeit very slowly; the *idea* was already coined in the 1970's. But as with any space mission, the cost for a *single* mission easily runs in the billions of dollars, necessitating a large degree of conservatism when it comes to mission planning and experimenting with innovative techniques. This necessary conservatism is the main reason behind the short length of Table 3.1 – for each new mission to an MP, only *one* asteroid/comet candidate is selected that is "most representative" of a whole class of similar bodies (scientifically speaking), in addition to being at the right place at the right time to be visited by a spacecraft launched with the least expensive launcher. The most innovating step is currently being made by the Dawn spacecraft, which will be the first spacecraft that will explore *two* asteroids in detail.

The main objective for the current mission will be the exploration of the outer Solar system, with the SBS in particular. Considering the vast distance (and associated time of flight) to the SBS, the spacecraft will not generate any useful data for 15 years (the expected time of flight). The spacecraft is however very likely to come across several MP's en route to the SBS, and with a proper design of its trajectory, it could perform simple flybys at MP's from *all* orbital groupings. With only a moderate increase in cost, the total science output of this mission will be several orders of magnitude greater. This fact is simply too promising to leave out of consideration, which justifies investigating this possibility in this research.

3.2.2 Mission Requirements

Primary Requirements

This mission's primary requirements are to shed light on the most prominent of the numerous unknowns surrounding the outer Solar system. The main scientific objectives to be satisfied are (in order of importance):

1. Establish the existence of the Solar bow shock, and measure the wave's location, orientation, magnitude, shock strength and temperature, and its dependency on interstellar magnetic fields and/or Solar activity.

2. Investigate the process responsible for the acceleration of the pick-up ions to anomalous cosmic rays.

3. Explore the contents of the interstellar medium, and its implications for the origin, evolution history and future of matter in our Solar system, galaxy and the universe.

4. Explore the impact of the Solar system on the interstellar medium, to validate, correct or reject existing models of the interactions of stellar systems with their environment.

5. Establish the capability of existing and proven technologies to enable fast and affordable exploration beyond our own Solar system.

As discussed previously, the existence of the SBS is still tentative, and based primarily on models derived from other stellar systems, indirect observations and theory. Its existence has however never been verified experimentally, which places this question high up on the list of requirements. Naturally, if it exists, all of its properties should be determined to refine current models and/or hypotheses.

Also, if this mission is a success, its accomplishments will establish that interstellar exploration is no longer a mere bonus of missions dedicated to planetary systems. If this mission completes its journey, space exploration will have reached the stage where interstellar phenomena can be investigated *in-situ* with an investment in time and cost comparable to planetary missions. It will then have opened up a whole new window of exploration opportunities; a sizeable accomplishment indeed.

Secondary Requirements

En route to the SBS the spacecraft could perform a variety of additional tasks. Exactly which of these *secondary* objectives can actually be addressed largely depends on the details of the final outcome, so satisfying these objectives will receive very little attention until the very last parts of this research. The secondary objectives (which will be clarified below) that could be considered for this mission are (in order of likelihood):

1. Explore various bodies in both the inner and outer Solar system, in search of clues to their origin and their role in the formation of this and other planetary systems.

2. Quantify the exact magnitude of the effect associated with the Pioneer and flyby anomalies, and investigate one or more of the proposed origins of these phenomena.

3. Investigate the mass distribution and the likely evolution history of objects in the Kuiper belt, in search of the origin of the belt's prevalent mass defect after the 1:2 resonance.

4. Investigate the validity of Einstein's equivalence principle, and quantify the spatial-temporal variation of the fine structure constant α in order to validate or reject grand unifying theorems proposed in some string theories.

As mentioned before, the design of the spacecraft's trajectory will try to take into account possible flybys with MP's, which is a *secondary* requirement only to keep the emphasis in this research on reaching the SBS. As mentioned in section 2.1.1 a sharp drop in the average amount of KBO's after the 1:2 resonance has been observed. Although it is very unlikely that this mission has the capacity to shed light on the possible causes for this phenomenon, at the very least it could verify whether this drop is an actual phenomenon.

In addition to performing MP-flybys and addressing related questions, this mission has the potential to investigate some aspects of at least three outstanding questions in physics. One is the *Pioneer Anomaly*; named after the Pioneer probes that first detected the phenomenon, all deep-space exploration probes launched to date have observed a very small but non-vanishing Sunward force *other than gravity*. There are many hypotheses (often quite far-fetched) that try to explain this effect, but none of these hypotheses seem to be consistent with observations. The spacecraft for this mission could be equipped with dedicated instruments (radio science experiments, sensitive accelerometers, etc.) to shed light on the nature of this effect, although this would place additional constraints on the spacecraft – for example, the effect can only be detected if the spacecraft does not have corrective thrusters, since these are a too large an error source.

As suggested by Maleki and Prestage [2001], performing a close Solar approach ($\sim$4 Solar radii or less) will make it possible to test certain theories that try to unify quantum mechanics and general relativity, proposed in some string theories. Currently more a theoretical construct than physical theory, this would mean that for the first time in its 50 year existence at least *part* of string theory can actually be tested experimentally. The proposed mission would accomplish this by performing measurements on the fine structure constant α, to see if it undergoes spatio-temporal changes when in a strong gravitational field; a close proximity Solar flyby is ideally suited for that purpose. As this is one of the longest outstanding questions in theoretical physics, it is certainly worthwhile to see whether it is possible to equip the spacecraft with the necessary instrument. The required instrument however consists of 3 individual atomic clocks, which can be expected to require a large fraction of the payload mass. Also, it is not yet known whether this spacecraft will indeed perform a close Solar flyby.

Along the same lines is a general relativity experiment, performed on the Cassini spacecraft by Bertotti et al. [2003]. According to general relativity, photons are delayed and deflected by the curvature of space-time caused by any massive body. The quantity $\gamma - 1$ (where γ is the Lorentz factor, $\gamma = c/\sqrt{c^2 - u^2}$) is a measure of how much gravity deviates from being a purely geometric effect, and thus how much gravity is influenced by the presence of other fields. Bertotti et al. [2003] found values of $\gamma - 1$ by analyzing radio signals emitted by the Cassini/Hugens probe as they passed very close to the Sun. They found $\gamma = 1 + (2.1 \pm 2.3) \times 10^{-5}$, which is within the predicted range. However, as the error term is larger than the measured value, this value *begs* to be determined with higher precision. The current mission is expected to have a long time of flight, but during its final leg to the SBS it will stay relatively close to the ecliptic. This means the probe's radio signals will periodically (with a 1 year period) move very close to the Sun as seen from the Earth, which can possibly result in more accurate estimates of this quantity.

3.3. Design Strategy

3.3.1 Mission Scenarios

Since this thesis research is quite broad in scope, it seems more than appropriate to first break the work down into several subtasks. Doing this will provide better overview, a structure to fall back on, and most importantly, the possibility to have a completed research (albeit partially) if only *one* of the subtasks can be finished satisfactorily. Such a structure also allows to add more scenarios, and broadens the scope of the research, in case all subtasks can be finished in a reasonable amount of time.

To this end, the whole research is broken down into 4 distinct scenarios, each of which is a complete mission design in itself. This breakdown follows naturally from the scientific objectives mentioned in section 3.2. The order of the scenarios is such that each one increases the complexity of the previous one. So if research on one scenario is complete, the next scenario would only require relatively small additions to all previous scenarios. These scenarios will be referred to throughout this thesis. They are:

Scenario 1: Bow Shock This will serve as the simplest scenario, in which all objectives, except reaching the SBS, will be left out of consideration. This case only aims to satisfy the primary requirements, and leaves all secondary requirements out of consideration entirely. It will also serve as a reference scenario, because it may be expected that the optimum for this scenario will be similar to that of the others.

Scenario 2: Bow Shock, Including Coincidental Minor Planets This scenario extends **Scenario 1** by checking whether the trajectory that leads the spacecraft to the SBS can be slightly altered, so that the spacecraft can also perform a flyby at a few MP's. These MP's need to come *close* to the spacecraft's initial trajectory, while they were not selected *a priori* or even taken into consideration in the optimization of the trajectory. As such, they will be *coincidental* opportunities.

Scenario 3: Bow Shock, Including Preselected Minor Planets This scenario is somewhat different from the other two. Here, the procedure to optimize the trajectory to the SBS will have additional constraints imposed on it, arising from *pre-selecting* one or several of the *most* interesting MP's that could be visited by the spacecraft. The final trajectory will be forced to go past one or several *fixed* other bodies which, depending on the specific bodies, potentially render the trajectory infeasible. Since this is not easily detectable *a priori*, pruning processes become more important for this scenario.

Scenario 4: Bow Shock, Including Maximum Science Output from Minor Planets This scenario combines **Scenarios 2** and **3**, by including the *quantity* and *scientific value* of all visited MP's as an additional optimization objective. The optimum found for this scenario shall have the *largest possible amount* of fly-by's at the *most interesting* unexplored MP's, while travelling to the SBS. As such, this scenario has a ***very tough*** set of constraints that must be met. To try and meet these constraints, several of the assumptions made in section 3.4 will most likely have to be altered to loosen these constraints. This scenario is the most challenging and most difficult to carry out. However,

if successful, the resulting mission is *guaranteed* to have the maximum possible science output in the given framework.

3.3.2 Approach

As shown by Mewaldt and Liewer [1999] and Leipold et al. [2005], a mission to the far edges of the Solar system is a feasible concept when assuming the spacecraft is equipped with a Solar sail. However, as mentioned earlier, Solar sails are still in their infancy and will in all likelihood not be employed in a mission of this magnitude in the near future. It therefore seems much more valuable to investigate whether a mission to the SBS is possible with more *mature* means of propulsion – ion engines or chemical thrusters. This is indeed one of the main aims of this research; to investigate whether an interstellar mission is within the capabilities of existing, fully mature technologies.

As this mission will try to accomplish its objectives by using an ion engine or chemical thrusters, the vast majority of the very high orbital energy required will have to come from somewhere other than its propulsion system. The technique that will be used as the primary source of energy is the (powered) GAM (see section A.4). Using this technique, energy can be extracted from the orbital motion of the planets, which will provide the bulk of the energy required to make the larger changes in the spacecraft's trajectory. However, past experience with this technique indicates that something more is needed – it would require too much time and too many gravity assists to build up the required energy. Therefore, the technique explored in both the IP and HE missions (a short duration, but very close approach to the Sun) will also be considered in this mission to serve as the "final kick" to the end of the Solar system. For all scenarios, the final leg close to the Sun will most likely prove crucial in reaching the SBS.

In short, for mission scenarios 1 and 2, a few (if any) gravity assist manoeuver will be used to put the spacecraft on a trajectory that will let it pass very close to the Sun. For mission scenarios 3 and 4, the gravity assists will be used primarily to try and approach several MP's – the gravity assists will change the trajectory towards a close Solar approach much more slowly. Mission scenarios 3 and 4 will therefore require more gravity assists and a longer total travel time.

3.4. Design Constraints

As demonstrated by the Voyager missions, *any* spacecraft flying to the outer Solar system is able to reach the SBS eventually. The challenge however is to reach it in an *acceptable* amount of time with an *acceptable* end mass. What "acceptable" means for these parameters is of course open to interpretation, but aside from that, it is imperative to make this mission as appealing as possible. This implies some set of constraints must be defined that will maximize this mission's appeal.

As these constraints will also drive the design process, most choices of appealing constraints will not allow a feasible mission design. Therefore, the choice of their values is a trade-off be-

tween appeal and feasibility. As such, they will be based on many previous missions, designs proposed or considered more recently, and the experience of the author with such designs. The selected constraints numerous and therefore subdivided into *time constraints*, *launch constraints* and *constraints on the spacecraft*.

3.4.1 Time Constraints

- Launch will take place somewhere between Jan. 1st, 2015 and Dec. 31st, 2025.
- For mission scenarios 1 and 2, the total mission duration will be constrained to a maximum of 15 years. As mission scenarios 3 and 4 are considerably more complicated, they will be constrained to a maximum of 25 years.

These constraints are the "hardest" constraints used throughout the design process. Any design that violates these constraints will be considered a complete failure.

3.4.2 Positions of the Planets During Assumed Launch Window

With the launch window now known, the location of the SBS's stagnation point found in section 2.1.2 and the method to obtain the ephemerides for the planets given in appendix B, there is enough information to make a rough sketch of the mission environment. To make a visualization of this, the positions of the outer planets during the assumed launch window have been plotted in Figure 3.6. As can readily be concluded, Neptune is not a very attractive planet to use for a GAM that directly precedes the leg to the SBS. Therefore, Neptune can already at this stage be abandoned completely. But there does not seem to be any such restriction on any of the other planets; Mercury until Uranus must all be taken into consideration for possible swingby sequences.

3.4.3 Launch Constraints

The maximum allowable value for the V_∞ provided by the launch vehicle ($C_3 = V_\infty^2$) will be set to

$$C_3^{\max} = 225 \text{ km}^2\text{s}^{-2} \tag{3.1}$$

This value for C_3 greatly exceeds the capacity of today's launch vehicles. To put this value in perspective, the two most powerful launchers ever used were those for the Ulysses mission[4] and the New Horizons mission, whose launchers provided approximately $C_3 = 128$ km^{2}s^{-2} and $C_3 = 157$ km^{2}s^{-2}, respectively[5]. Keep in mind however that C_3 is the *square* of V_∞; the chosen value of $V_\infty^{\max} = 15$ km/s lies fairly close to Ulysses' and New Horizons' $V_\infty = 11.3$ km/s and $V_\infty = 12.5$ km/s, respectively. This high value was chosen to prevent over-constraining the problem initially and facilitate locating promising locations in the search space; initial trials showed that the feasibility of solutions often depends crucially on the launcher's C_3.

[4]Note that Ulysses was launched with the Atlantis Space shuttle, so its *overall* C_3 did not come from its launch vehicle alone; most of it came from the spacecraft's own upper stages.

[5]These values were calculated by solving the Lambert problem corresponding to the mission's first leg, and taking the magnitude of the resulting departure velocity with respect to Earth.

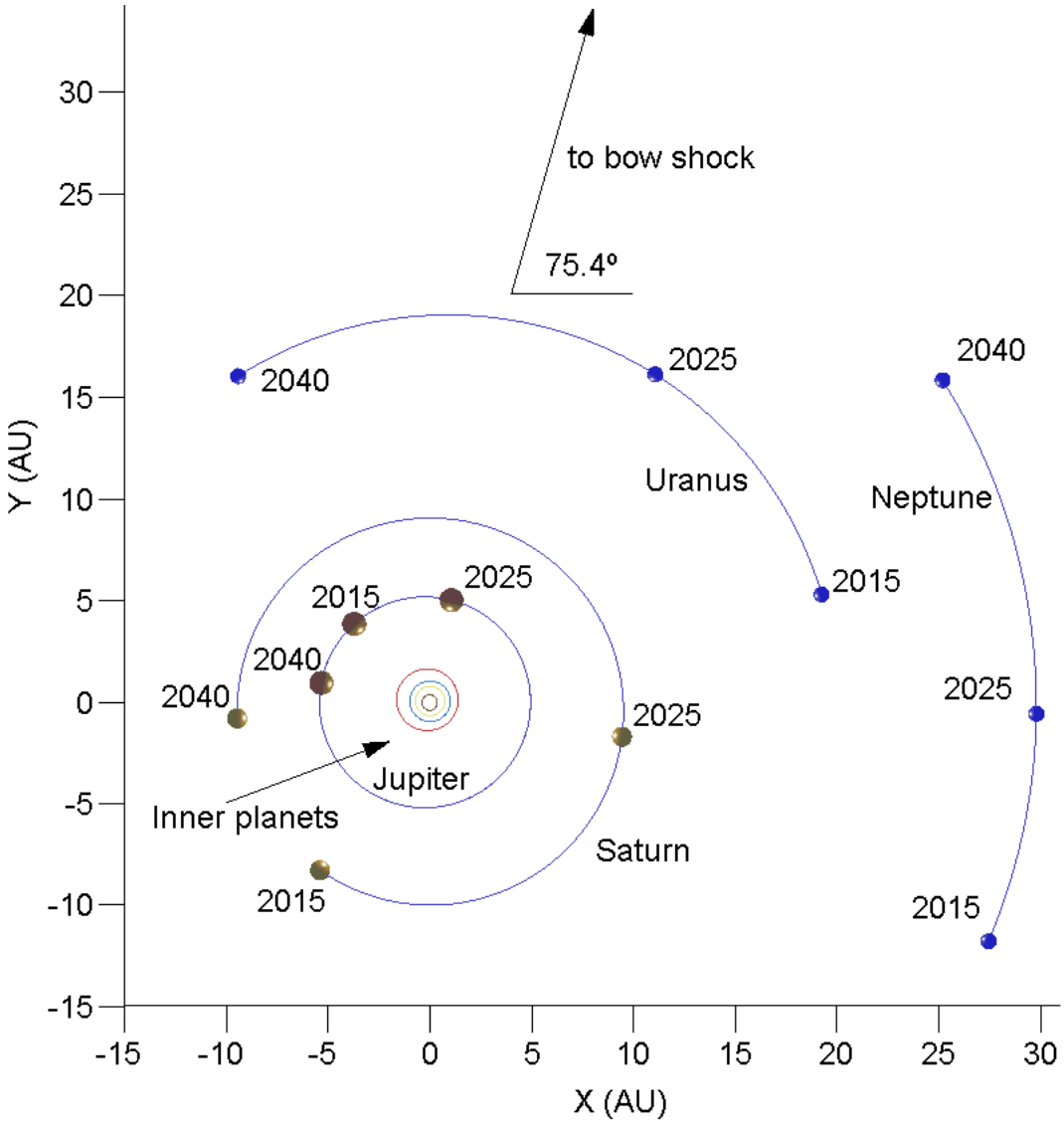


Figure 3.6 The positions of the outer planets during the assumed launch window. Also the direction way to the bow shock is indicated. Although this is a 2-D plot, it is a fairly accurate representation of the real configuration, since the direction to the bow shock only descends ∼7°in the negative z-direction.

Still, minimizing its value (or at least putting it in the feasible range) is therefore necessarily part of the optimization process.

But even when the final solution requires a C_3 higher than what is possible with today's launch vehicles, the spacecraft's engines themselves can provide the "missing" V_∞. Naturally, this would negatively impact the overall ΔV-requirements for this mission. Although the C_3-parameter is the defining parameter for launches of interplanetary missions, it is not often listed as a characteristic parameter of specific types of launchers. The characteristic is often given in the form of the launcher's capacity for *mass to Geostationary Transfer Orbit (GTO)*, m_{GTO} – the mass a launcher is able to insert into GTO. A few examples of these masses are given in Table 3.2. With the help of Figure 3.4.3 and the equations elaborated in appendix A,

Table 3.2 Various launchers and their m_{GTO} capacities. From [Isakowitz, 1991].

Launcher	m_{GTO} [kg]
Ariane 5 (ESA)	6800
H-2 (JAXA)	4000
Atlas II-AS (NASA)	3606
Titan IV (NASA)	4540
Proton D-1/SL-12 (RKA)	5500
Long March 3B (CNSA)	4500

this characteristic mass can be used to estimate the spacecraft's wet mass m_{wet} for any given value of C_3:

$$\begin{aligned} V_{\text{req}} &= \sqrt{\frac{C_3}{2} + V_{\text{esc,p}}^2} \\ V_{\text{LV}} &= \sqrt{V_{\text{esc,p}}^2 - 2\epsilon_{\text{GTO}}} \\ \rightarrow \Delta V_{\text{add}} &= V_{\text{req}} - V_{\text{LV}} \qquad (3.2) \\ \rightarrow m_{\text{wet}} &= m_{\text{GTO}} \cdot \exp\left(\frac{-\Delta V_{\text{add}}}{g_0 I_{\text{sp}}}\right) \qquad (3.3) \end{aligned}$$

where V_{req} is the required speed to meet the C_3-demand, V_{LV} is the speed that can be provided by the launcher based on a geostationary transfer orbit, $V_{\text{esc,p}} = \sqrt{2\mu_E / r_{\text{GTO},p}}$ is the escape velocity at the pericenter of a GTO, $\epsilon_{\text{GTO}} = -\mu_E/2/a_{\text{GTO}}$ where $a_{\text{GTO}} = (r_{\text{GTO},p} + r_{\text{GTO},a})/2$ is the characteristic energy of a GTO, and the last equation follows from Equation A.31 (Tsiolkovskii's equation). Note that usually, $r_{\text{GTO},p} = 100$ nautical miles ≈ 185 km is often upheld. With this, $V_{\text{esc,p}}$ and ϵ_{GTO} are also constants.

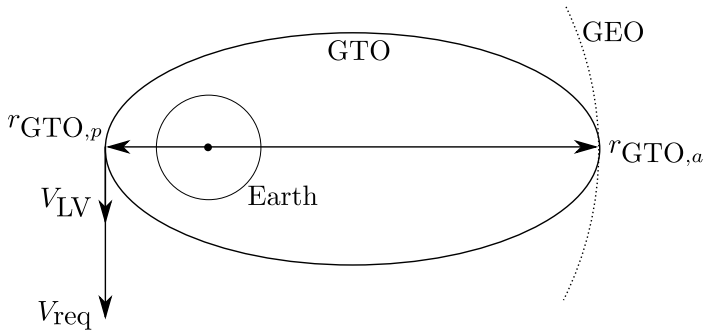


Figure 3.7 Several quantities in a GTO, used to compute the spacecraft's required escape speed V_{req} from a given launcher's m_{GTO}.

Last but not least: in this research there will be no restriction on the launch *site* to be used, that is, the direction of the heliocentric velocity just after Earth-departure is left completely unconstrained; this is left here as future work.

3.4.4 Constraints on Spacecraft

This mission puts some very high demands on the spacecraft: it has to face both extremely high temperature when it starts its final leg close to the Sun, and intensely cold interstellar space, when the Sun is no more than another bright star. It has to be resistant to wear (it has to function perfectly for ~ 25 years), highly autonomous (the communication delay will eventually reach the order of days), light weight (ion engines do not produce much thrust), energy efficient (it can be assumed that most of the power will be taken by the communication subsystem), and resistant to intense radiation (when outside the protection of the SW it will be exposed to much more intense cosmic rays). Therefore it is crucial, already in this stage, to assume very conservative values for the spacecraft parameters. Doing so leaves much room in later stages to adjust these assumptions to more optimistic values, if the simulations show that that is permissible.

Mass

The spacecraft wet and dry mass are very important design parameters – they define the amount of propellant available to the spacecraft, which subjects the problem to a hard constraint. These masses will initially be assumed equal to

$$m_{\text{wet}} = 3,000 \text{ kg} \qquad \text{and} \qquad m_{\text{dry}} = 50 \text{ kg} \tag{3.4}$$

The wet mass m_{wet} has been loosely based on the assumption of a heavy lift launch vehicle (Ariane 5, Atlas V, Delta IV, ...) that also delivers a very high escape velocity to the spacecraft. The dry mass m_{dry} has been initially set to an unrealistically low value, just to test the feasibility of the mission in general – if some design is found to be impossible for even these unrealistic values, that design had better be abandoned.

The given values are indeed quite far from realistic. Part of the goal of the optimizations is to decrease the value of m_{wet} to try and reduce launch cost, and increase the value of m_{dry} so that the largest amount of useful mass will be delivered to the SBS. However, the given values provide an initial guideline and allow for much design freedom (multiple ion engines, disposable Solar arrays, ...)

For completeness: the *wet mass* is defined as the spacecraft mass *after* the Earth escape, so it is different from the launch mass. The *dry mass* is the spacecraft mass that is left when all consumables have been used up.

Specific Impulse

Another driving design parameter is the *specific impulse*, I_{sp}. This quantity is a measure of the total deliverable amount of energy by a specific engine. The different mission scenario's will look at both high-thrust and low-thrust systems (see parts III and IV), and different values for I_{sp} will have to be assumed for both types:

$$I_{\text{sp}}^{\text{high}} = 300 \text{ s}, \tag{3.5}$$

$$I_{\text{sp}}^{\text{low}} = 3000 \text{ s}. \tag{3.6}$$

These values are rather conservative, especially the value for I_{sp}^{low} – most modern ion thrusters have a *much* higher I_{sp}. The most promising engine at the moment, the Dual-Stage 4-Grid ion engine (DS4G), has demonstrated to be capable of exhaust velocities up to 210 km/s [Walker, 2006], giving it a specific impulse of $\sim 20,000$ s. Nevertheless, the value of $I_{sp}^{low} = 3000$ s will be maintained throughout the optimizations, and only scaled upward when none of the optimizations results in a satisfactory design.

Power

Since the current mission will operate at > 200 AU, using Solar power for mission operations is out of the question – Radioisotope Thermoelectric Generators (RTG's) will have to be used for that purpose. The Cassini/Huygens, New Horizons, Galileo and Ulysses missions all used (several) GPHS-RTG generators, each capable of generating ∼200 W continuously (on average) for over 15 years.

However, these generators are no longer produced. All new missions requiring nuclear power supplies will have to use either Multi-Mission Radioisotope Thermoelectric Generator (MMRTG) or Stirling Radioisotope Generator (SRG). These types of generators produce less power (typically ∼110 W, Beginning Of Life (BOL)) but are applicable to a broad range of different missions. These generators are currently still in the prototype phase, but the estimated masses are ∼34 kg for the SRG, and ∼45 kg for the MMRTG [Whatmore et al., 2009].

Since the final leg in the current spacecraft's trajectory is likely to take it very close to the Sun, it will seem quite wasteful not to use that energy in some way to the spacecraft's advantage. Thus, during the design of the low-thrust mission to the SBS, two options will be considered; one that uses nuclear power exclusively (Nuclear Electric Propulsion (NEP)), and one that partly relies on Solar arrays (Solar Electric Propulsion (SEP)) that are discarded when they no longer have any use. For both types (and the high-thrust mission), it will initially be assumed that the spacecraft is equipped with 2 SRG's, fixing the total power output to 220 W BOL and the required mass to 70 kg.

For the hybrid type (Solar + Nuclear), it should be noted that the total available power depends on the distance to the Sun. The Solar irradiance generally follows an inverse-square law, so the available power from the Solar panel can be described by the relation

$$P_r = \frac{P_0}{r^2}, \tag{3.7}$$

with r the distance to the Sun in AU, and P_0 the power level near Earth (1 AU). No "typical" value can be found for P_0, since the Solar panel area is typically highly dependent on the specific mission. Initially, a simple initial value of $P_0 = 5$ kW will be assumed, roughly corresponding to ~ 4 m^2.

Degradation of RTG's All spacecraft power systems are subject to degradation. In case of nuclear batteries like the RTG or SRG, degradation of the battery's performance is due to the decay of the nuclear material and slow precipitation of phosphorous doping in the

n-type leg of the RTG's thermocouple [Patel, 2005, chapter 20.5]. Decay of nuclear material generally follows an exponential curve, described by the relation

$$N_t = N_0 \left(\frac{1}{2}\right)^{t/t_{1/2}}, \tag{3.8}$$

where N_t is the quantity of nuclear material left after a time t in years, N_0 the initial quantity, and $t_{1/2}$ is the material's *half-life*. For the MMRTG or SRG, the fissile material is ^{238}Pu, which has a half life of 88.7 years. This means that the battery's power output will be roughly half of what it was at launch, after 88.7 years. So degradation of RTG's is dominated by the erosion of the n-type leg, an effect which can be quantized by analyzing the power output of the RTG's onboard the Voyager 1 spacecraft. Using the data from Table 20.2 in [Patel, 2005], the best-fitting exponential curve is found to be described by

$$\begin{aligned} \eta_t &= Ae^B t \\ &\approx 97.67 \cdot e^{-0.01772t}, \end{aligned} \tag{3.9}$$

with t the time after launch in years and η_t the power output in % of the original value. Note that substituting $t = 0$ in this equation does not yield 100%, so the model which will be used in this research's analysis will be adjusted to

$$\begin{aligned} \eta_t &= Ae^B t \\ &\approx 100.0 \cdot e^{-0.01772t}. \end{aligned} \tag{3.10}$$

This adjustment is also assumed to account for the slight improvements of the MMRTG's with respect to the 22-year old RTG's on board Voyager 1.

Degradation of Solar Panels When simulating the NEP/SEP hybrid mission, also some degradation factor for the Solar panels needs to be assumed. As mentioned in Patel [2005, Table 8.2], common causes for degradation in Solar panels are radiation, contamination by spacecraft propellants, micro-meteoroid damage, ultraviolet rays, erosion due to interactions with charged particles, and many more less prominent causes. Mentioned table also lists degradation factors, which had been derived for an Earth-orbiting, MEO satellite with a lifetime of 15 years. Although such a mission has no relation to the current mission, and most of the listed degradation factors do not apply, these combined factors will give a careful, conservative initial value for a Solar panel's degradation. Using said factors, the degradation model to be used becomes

$$\begin{aligned} \eta_t &= L^t \\ &= \left((0.98 \cdot 0.99 \cdot 0.98 \cdot 0.97 \cdot 0.98 \cdot 0.98 \cdot 0.98 \cdot 0.99 \cdot 0.98)^{1/15}\right)^t \\ &= 0.9886^t, \end{aligned} \tag{3.11}$$

with t the time after launch in years and η_t the power output as a fraction of the original value. A secondary effect which affects the performance of Solar panels, is that a panel's

efficiency depends slightly on the Solar irradiance; the dimmer or more indirect the Sunlight is, the less efficient the Solar panel becomes. This effect is mentioned in Wertz [2001], where it is taken into account by slightly modifying the inverse-square law in Equation 3.7 to

$$P(r) = \frac{P_0}{r^{1.7}}. \tag{3.12}$$

However, in the more recent book [Patel, 2005, section 8.9.1], a somewhat more accurate description is given. The plotted curve in mentioned source is very well approximated by the equation

$$\begin{aligned} \eta_I(I) &\simeq 1 - e^{-I/100}, & (3.13)\\ \Rightarrow \eta_I(r) &= 1 - e^{-\frac{I_0}{100r^2}}, & (3.14) \end{aligned}$$

with I the intensity of the incident light, and $I_0 = 1367$ W m^{-2} the Solar constant. Using this equation, the power as a function of time and distance can be written as

$$P(t,\, r) = \frac{\eta_t(t)\eta_I(r)P_0}{r^2} = \frac{0.9886^t \left(1 - e^{\frac{-I_0}{100r^2}}\right) P_0}{r^2}. \tag{3.15}$$

If the power output at which the Solar arrays will be jettisoned is written as P_{jettison}, the corresponding jettison distance r_{jettison} must be written as a function of time,

$$\begin{aligned} P(t,\, r) &= \frac{0.9886^t \left(1 - e^{\frac{-I_0}{100r^2}}\right) P_0}{r^2} = P_{\text{jettison}} \\ &\Rightarrow \frac{\left(1 - \sum_{n=0}^{\infty} \frac{u^n}{n!}\right)}{r^2} = \frac{P_{\text{jettison}}}{P_0 \cdot 0.9886^t}, \quad u = \frac{-I_0}{100r^2} \\ &\Rightarrow \frac{1}{r^2} + \frac{I_0}{100r^4} - \frac{P_{\text{jettison}}}{P_0 \cdot 0.9886^t} \approx 0 \\ &\Rightarrow r_{\text{jettison}}(t) \approx \sqrt{\frac{2I_0/100}{-1 + \sqrt{1 + 4\frac{I_0 P_{\text{jettison}}}{100 P_0 \cdot 0.9886^t}}}}. \end{aligned}$$

For the sake of completeness, the relation between the jettison distance $r_{\text{jettison}}(t)$ and time t is shown graphically in Figure 3.8 for several values of P_{jettison}. The initial available power at 1 AU, P_0, was taken equal to 10 kW.

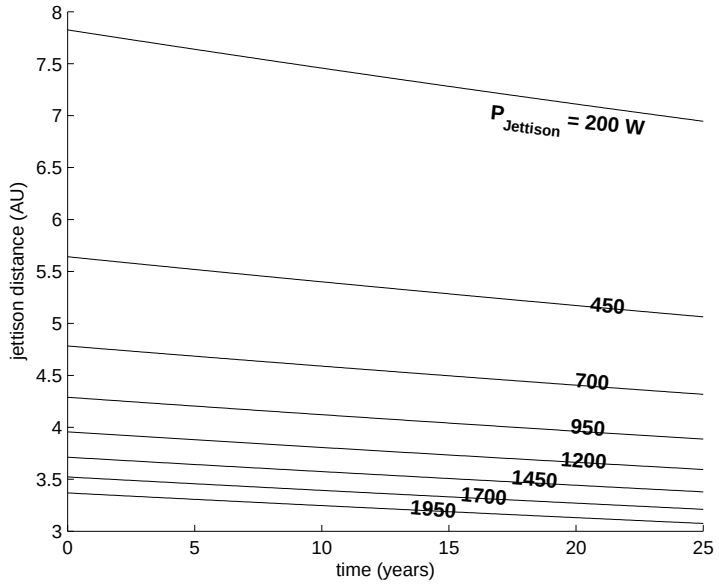


Figure 3.8 Solar panel jettison distance as a function of time and the power level at which the panels are to be jettisoned.

4 Software

The availability of optimization software for trajectory design to the general public is surprisingly limited. Virtually all existing software is either not accessible to the general public, closed-source and expensive, copyrighted by the respective space agency (see [Bombardelli and Izzo, 2007a,b]), or simply very badly programmed. There are however two open-source programs traditionally used by all M.Sc. students in the Delft University of Technology, who work on interplanetary trajectory optimization problems. These two programs started as a generalization of the design process of various types of missions, that previous students had considered in their thesis work. Over several generations of M.Sc. students, these programs have grown substantially in size and capability, and are still constantly being extended and updated. They are called Genetic ALgorithm Optimization of a MUltiple Swing-by Interplanetary Trajectory (GALOMUSIT) and OPTimization of Interplanatary trajectories by Delft University Students (OPTIDUS).

4.1. GALOMUSIT & OPTIDUS, an Overview

The GALOMUSIT program is actively being developed by students at the Faculty of Aerospace Engineering at the Delft University of Technology. As of early August 2009, GALOMUSIT is capable of optimizing both low- and high-thrust interplanetary trajectories to the eight planets of the Solar System, Pluto, and a small set of asteroids and comets within our Solar System. The most recent development is the inclusion of the possibility to use *deep-space manoeuvres* at one or many locations along the trajectory. It can handle a variety of different mission types, including one-way trajectories or orbit insertion at a selected target body. It uses a fairly advanced genetic algorithm to perform its optimizations.

OPTIDUS on the other hand, is more suited for very specific missions. That is, the user can optimize *any* mission, since the OPTIDUS has a modular architecture – it has many subroutines available that can be used at will, with customizable options and settings. It

also hosts a variety of problem-specific routines, which the user can modify to suit his/her needs. Its contents is shown in Figure 4.1. It currently has two main versions; one specifically designed for single-objective optimization, and one for Multi-Objective Optimization (MOO). Originally it used a Genetic Algorithm (GA) with a variable population size, but this has recently been extended to include more advanced optimization algorithms such as Particle Swarm Optimization (PSO) and Differential Evolution (DE). Also, it has been modified to include the possibility of parallel computing.

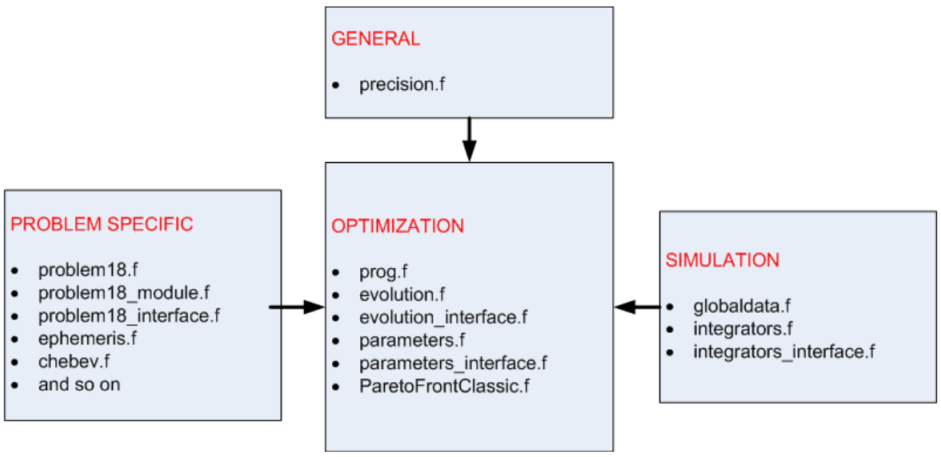


Figure 4.1 Categorization of FORTRAN-files within OPTIDUS. Note that this is an outdated picture; its contents has changed some in the past year, but it still shows the modular buildup of the program. This is [Heiligers, 2008, Figure 13.1].

4.2. Disadvantages

Although elaborate and quite advanced pieces of software, there are some obvious disadvantages to them. Many students newly introduced to these two programs found themselves asking the following questions:

Why are there two distinct programs? Their current functionality more or less overlaps; is there really still a need for two separate programs? Can't they be combined into a single, *all-purpose* trajectory optimization program?

Why are they written in FORTRAN? Although a feature-rich and powerful programming language, it is no longer the *de facto* standard in the field and is actually disappearing, slowly but surely. Also, new developers normally have to learn FORTRAN from scratch while they are already proficient in one or more other languages.

Why is updating their documentation not part of the "job" of the developers? A potential problem for *any* open-source program; the **documentation**. The programs' documentation is chronically outdated, and it periodically requires dedicated students to plough through all the routines written by others all over again and document them.

Why is there no utility that facilitates multiple users working on the same program? Currently students are forced to make copies of the respective `FORTRAN`-files and *discuss* who works when on what part. This can obviously lead to confusion and unnecessary (duplicate) work, is normally a large error source and can even lead to accidental overwrites, incompatibilities and un-compilable programs. This sort of process has been rigourously standardized for decades.

Why are they terminal-applications? Although console/terminal applications are considerably more easy to develop, that kind of thinking has time and again proven to be unable to withstand the ravages of time. In the age of graphical user interfaces, many new users have never even seen a terminal, and must spend a considerable amount of time on becoming acquainted with it.

Why is it not possible to make plots of the optimization results? The results from an optimization are normally best displayed graphically, both for in reports and to inspect the validity and/or value of the results. Creating even fairly trivial plots now requires copying text-files and importing these into another application. This process seems completely unnecessary.

While some of these matters usually disappear after some exposure to the programs, for the majority of these issues that seems to be more due to "getting used to it" than using actual solutions.

4.3. Complete Re-write: Skipping Stone

To address these issues[1], a new program will be developed entirely from scratch. Higher-level languages such as the numerical analysis program MATLAB® by The MathWorks®, or its open-source clones (FreeMat, GNU/Octave, Scilab) are ideally suited for students with non-programming backgrounds. Such students need to use or co-develop a program that changes extremely often, or requires much tweaking to "get it right". The `M`-language has many advantages:

- It is an interpreted language, meaning that programs no longer have to be compiled before they are executed. This saves much time and makes it possible to develop large programs quite rapidly. Moreover, substantial changes can be implemented and used immediately, even while the main program is running.
- It supports object-oriented programming, functions, sub-functions, nested-functions, parallel computing, and has many basic functions already built-in and as optimized as can be. It also takes away the need for code optimization regarding memory usage, as this is usually automated by the interpreter.
- Programming in `m`-language is *dynamic*. One of the benefits of this is that most built-in functions (and user-developed ones) can be called with a *variable* amount (and type) of arguments, without the need to overload each function. For instance, the command

[1] But mostly just for the fun of it!

`sum(a)` will add all the values in the columns of matrix `a` (default behavior), while `sum(a, 2)` will sum the rows. Another benefit is nested functions – using nested functions will take away the need to pass many arguments to functions, as nested functions share a common memory space with its parent function; much like global variables, but then without the drawbacks. This both facilitates implementation, improves readability, and is more efficient in terms of memory usage.

- Because it is an interpreted language, debugging is also relatively "painless"; including a few commands before runtime or in loop iterations can generate entire plots, figures, animations or tables showing *graphically* what the contents of each variable (or part thereof) is.

- Standard MATLAB® includes many useful tools and toolboxes. For instance, the GUIDE™ tool allows the user to construct large Graphical User Interfaces (GUI's) and program their functionality *graphically*. That generally means that a developing large and complex GUI's can be done quite swiftly. More importantly, if anything needs to be *changed* on an existing GUI, it is a matter of click-and-drag, tweak, and run.

- The MathWorks® has set up a website on which users of the software can exchange M-files with each other. Naturally, this is equipped with a user-based rating system, so that it is fairly easy to pick out well-programmed functions or classes. These can be downloaded free of charge, and after some testing, can readily be added to the program's functionality.

Naturally, these benefits come at a price:

- MATLAB® is copyrighted and proprietary software of The MathWorks®. As such, M-files written in MATLAB® can not be run on systems on which MATLAB® is not installed. Licenses can indeed be quite expensive, especially when considering cluster-computations – licences cost €500 for individual users, and up to ten times that amount for academic group licenses. And that is just for the basic version, i.e., without any toolboxes.

- Because it is an interpreted language, it can be quite slow at times. Especially when badly programmed, the difference between its speed and that of a compiled language can climb up to a factor of 100 or more. This is especially noticeable in (small) loop structures; here, the associated overhead (loop-unwinding at runtime, and not using cache memory) can take up the majority of the time.

- There are some issues regarding the language itself, that programmers who are used to another language generally find hard to get used to. For instance, function calls and array indexing have the same syntax (e.g., `f(x)` can mean both the x^{th} element of array `f`, or a function call to function `f` with `x` as its input).

- Code written in newer versions of the program can often not be run on previous versions. Although the reverse is often true (old code runs on new versions), many users tend to stick to old versions because of the licensing problem, necessitating time consuming re-writes of newer code to make it compatible with older versions. For instance, the

`classdef`-command (used to write classes) is unknown to versions R2006 and earlier – classes in R2006 took on a very different form.

Fortunately, many of these disadvantages are mitigated by the following facts. MATLAB® is available to all graduate students in Delft University of Technology, and many students indeed have had much exposure to programming in MATLAB®. Also, it is widely used in the sciences and industry, and more recently also on large supercomputer-clusters, so that a thorough experience in developing MATLAB®-programs is becoming a more and more valuable skill to have. There are also many open-source clones of the language, most notably GNU/Octave – this is a piece of software that aims to mimic MATLAB® and improve upon it on obvious points, so that most (if not *all*) `M`-code developed in MATLAB® can be run in GNU/Octave.

Most importantly, the issue of loss of computation speed has been addressed in versions R2006 and later by various means. These later versions are equipped with the *Just-In-Time* (JIT) accelerator, which compiles small segments of code just prior to execution. This technique drastically improves the aforementioned overhead problem in loop structures, as these loops are now run directly from the CPU's cache. Also, if this is not enough, it is fairly easy to write C++ or `FORTRAN` programs in a fashion that their (compiled) functionality can be used directly in MATLAB® (*Matlab EXecutables*, MEX). There is hardly any difference in execution speed between the MEX-file and the same routine or function written in the "pure" language.

Considering all of these advantages and disadvantages, it seems that the "4^{th} generation" `M`-language indeed has its drawbacks, but will likely see many more improvements in the nearby future. It definitely is a language that will see a steadily growing user base. In that light it seems very worthwhile to develop the new program in MATLAB®. As a working title, Skipping Stone will be adopted; a name reminiscent of what interplanetary MGA-trajectories look like. Ideally, Skipping Stone is to become a completely open-source project, protected by a GPL. This would render it intellectual property of the Delft University of Technology, while enabling users worldwide to use, modify and redistribute it according to their needs; but such ambitions are for later concern.

For the sake of completeness, a simple black-box diagram of Skipping Stone is given in Figure 4.2. As this figure shows, this program will be more like GALOMUSIT than OPTIDUS. However, much effort will also be put in implementing a wide variety of functions in such a fashion as to make them portable (plugin-style). These functions can then be (re)used by both Skipping Stone and users with more specific missions; behavior that resembles the modularity aspects found in OPTIDUS.

As this program is developed from scratch, all associated theory will be discussed in slightly more detail than strictly necessary. Indeed, parts III and IV of this thesis are entirely devoted to all of its implementation details, considerations on efficiency, and of course, the exact definitions of all terms in Figure 4.2. As a final remark: this re-write comprises a considerable amount of work, and it is unrealistic to assume that *every* bit of functionality that is now available though GALOMUSIT and OPTIDUS will find its way into Skipping Stone.

Therefore, this re-write should be regarded more as a first step in a different direction, to be perfected and expanded in much the same sense as GALOMUSIT and OPTIDUS are being developed now.

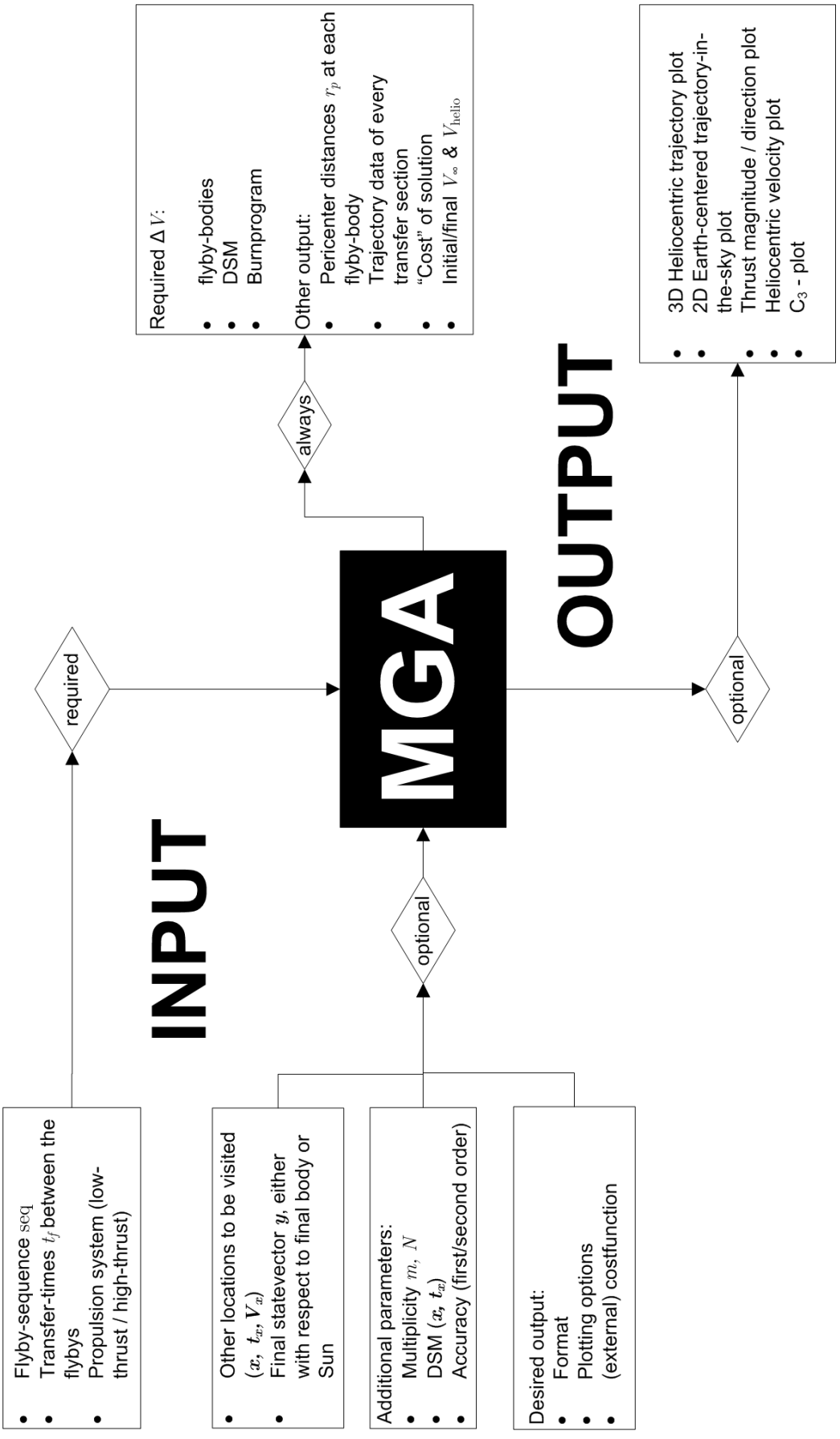


Figure 4.2 The basic operations of Skipping Stone.

Part II

Optimization

5 Fundamentals of Optimization

Optimization is a subject that is encountered quite frequently by most modern scientists and engineers, regardless of their specialization. A mastery of optimization techniques enables one to find the best possible design for solar arrays, the maximum economic utility in a complex and growing market, the very best shape of a piston in a combustion engine or even such "simple" problems as finding the best shape of an egg-box that breaks the least number of eggs on the average.

Although optimization theory inherently tries to minimize computational burden, most problems still require many calculations before they are fully optimized. The breadth of optimization theory has therefore exploded since the invention of the digital computer, and continues to do so with the rapid improvement in the available computational power. Ever more complex optimization problems can be solved nowadays thanks to these improvements, but mainly because optimization theory has received so much attention in the last few decades.

The purpose of this research is to find the best possible trajectory for a mission to the SBS and the maximum number of minor planets. Therefore, optimization is indeed the primary focus of the research and deserves a central role in this thesis.

5.1. Local and Global Minima

A distinction must be made between *local* and *global* minima of a function. Local minima of a function $f(x)$ are defined as [Sun and Yuan, 2006, section 1.4]

$$\hat{x} \text{ is a local minimizer if } \exists\delta > 0 \rightarrow \left\{ f(\hat{x}) \leq f(x) \forall x \in \mathbb{R}^N; x \neq \hat{x}; |x - \hat{x}| < \delta \right\}. \quad (5.1)$$

$$\hat{x} \text{ is a } \textit{strict} \text{ local minimizer if } \exists\delta > 0 \rightarrow \left\{ f(\hat{x}) < f(x) \forall x \in \mathbb{R}^N; x \neq \hat{x}; |x - \hat{x}| < \delta \right\}. \quad (5.2)$$

On the other hand, a *global* minimum is defined as

$$\hat{x} \text{ is a global minimizer if } f(\hat{x}) \leq f(x) \forall x \in \mathbb{R}^N,\ x \neq \hat{x} \tag{5.3}$$

$$\hat{x} \text{ is a } \textit{strict} \text{ global minimizer if } f(\hat{x}) < f(x) \forall x \in \mathbb{R}^N,\ x \neq \hat{x} \tag{5.4}$$

These definitions are shown for a one-dimensional function in Figure 5.1. Note that the definitions of global and local minima as given above automatically include also the function values at the end of the domain of the function, if these function values are indeed lower than anywhere else in the function's domain.

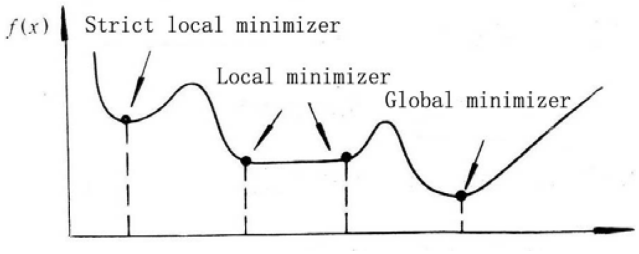


Figure 5.1 The definition of a local and a global minimizers, for a simple one-dimensional function. Adopted from [Sun and Yuan, 2006, Figure 1.4.1].

Since the goal of optimization is to find the absolute minimum value of a function within some search space, the function's *global* minimum is usually the minimum of interest. However, contrary to *local* minima, global minima are notoriously hard to find. The most advanced optimization methods are essentially very efficient *hill climbers* – when given some initial estimate, they iteratively try to find new values for the decision variable that always decrease the function value[1]. That implies that if the *global* minimum is of interest, those methods implicitly assume an initial estimate *near the global minimum* has been provided. If this is not the case (which is usually true), those hill climbers can only find a *local* minimizer of the function. Local optimization methods are the subject of chapter 6, whereas global optimizers will be discussed in chapter 7.

5.2. Unconstrained Optimization

An unconstrained optimization problem can be stated as

$$\min f(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^N, \tag{5.5}$$

that is, find the minimum of the *objective function* $f(\boldsymbol{x})$, while its N-dimensional variable, or *decision variable*, $\boldsymbol{x}$ is free to range over all of $\mathbb{R}^N$. In other words, the *search-space* for the objective function $f(\boldsymbol{x})$ is infinitely large. When stated like this, the objective of this optimization problem is to find the *global* minimum of $f(\boldsymbol{x})$.

Any problem that needs to be *maximized* instead of minimized, can be treated with the same

[1] This is not true *per se* for methods used for constrained functions – see chapter 6 for details.

methods as used in minimization, provided that the objective function $g(\boldsymbol{x})$ is re-written into the form

$$f(\boldsymbol{x}) = -g(\boldsymbol{x}),$$

essentially turning the maximization problem into a minimization problem by a simple reversal of sign.

An example of an unconstrained problem is to find the altitude h of a rocket where the aerodynamic stresses on the hull are maximal ("max-q"). It is obvious that there is such a maximum, since the aerodynamic stresses are zero when the rocket is on the launch pad, and also when it is in freefall in the vacuum of space, but larger than zero during the entire ascent. The (largely oversimplified) objective function for this this problem would look like

$$\min f(h) = \min \left(-\frac{1}{2} C_d \rho(h) V^2(h) S \right),$$

where C_d is the rocket's drag coefficient, S its reference area, and $\rho(h)$ and $V(h)$ the air density and rocket's speed as a function of its altitude, h. In this case, h is essentially unbounded (although flying underground is a bit impractical), thus the problem is essentially an unconstrained one. Of course, this is a simple example which can also be solved analytically, but a natural extension of this problem would be to find the flight path for a given rocket that minimizes max-q while maximizing payload-in-orbit – which would be a relevant problem when launching the Space Shuttle.

5.3. Constrained Optimization

Most objective functions representing realistic optimization problems have regions in their search space where the objective function is either non-smooth, discontinuous or otherwise not well-behaved. In the max-q example above, flying underground ($h < 0$) would result in very strange values for the objective function. Also, in other problems, the decision variable may partly represent a quantity like "project cost", in which case "infinite cost" for the optimum value of the objective function would be quite undesirable. Therefore, some *constraints* are posed on most realistic optimization problems. Constraints can be given as *bound constraints*, where the decision variable $\boldsymbol{x}$ is limited to some subspace X, *inequality constraints*, or *equality constraints*. A general constrained optimization problem then has the form

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \quad , \boldsymbol{x} \in X, \\ \text{s.t.} \quad & c_i(\boldsymbol{x}) \quad = 0, \, i \in E, \\ & c_i(\boldsymbol{x}) \quad \leq 0, \, i \in I, \end{aligned} \tag{5.6}$$

where the $c_i(\boldsymbol{x})$ are the constraint functions, and E and I indicate the i-indices for (E)quality and (I)nequality constraints, respectively.

When **all** the constraint functions are linear, the process of solving this problem is usually referred to as *linear programming*. When **some or all** of the constraints are non-linear,

it is called *non-linear programming*. In this terminology, the term "programming" does not refer to the same term used in computing (e.g., converting some calculation to computer language), but rather to the mathematical program to be followed (i.e., satisfying all the constraint functions while minimizing the objective function).

5.3.1 Converting Constrained Problems into Unconstrained Problems

Unfortunately, non-linear programming is mostly restricted to problems where the first and second derivatives are either known analytically or are not too costly to compute numerically. Moreover, both the objective function and the (nonlinear) constraint functions are required to be continuous, convex and smooth (continuous derivatives).

For many realistic problems however, this is not the case. Often the objective function or constraint functions are not so well behaved. Also, many problems involve thousands of independent variables, in which case the corresponding gradient and Hessian matrices contain a few million elements. If no explicit expressions can be derived for these matrices, they need to be computed numerically. Computing numerical derivatives usually takes up the vast majority of computation time, so in these cases, using a different method would be preferable.

Fortunately, it is possible to convert constrained problems into unconstrained ones, so that much simpler methods can be used. This conversion relies on two principles: *coordinate transformations* of the original decision variables, and the concept of *penalty functions*.

Bound Constraints

If M of the decision variables have single-bound constraints imposed on them,

$$x_i \geq LB_i, \text{ or} \tag{5.7}$$
$$x_i \leq UB_i, \tag{5.8}$$

with $i = 1, \ldots, M$, they can be transformed by introducing a new variable z_i,

$$x_i = LB_i + z_i^2 \text{ or} \tag{5.9}$$
$$x_i = UB_i - z_i^2 \tag{5.10}$$

respectively, essentially transforming the optimization along the dimensions $1, \ldots, M$ into an unconstrained optimization (because obviously $y^2 \geq 0$). Likewise, for those decision variables that have compound bound constraints,

$$LB_i \leq x_i \leq UB_i, \tag{5.11}$$

the following transformation may be applied:

$$x_i = LB_i + (UB_i - LB_i)\frac{\sin(z_i) + 1}{2}. \tag{5.12}$$

These transformations were suggested by D'Errico [2006] and were found to work well on virtually all bound-constrained problems he tested. The sine-function introduced in Equation 5.12 may give rise to some problems, such as multiple solutions where originally there was only one. But of course, these solutions are all completely equivalent. However, one may avoid this problem altogether by using the transformation

$$x_i = LB_i + (UB_i - LB_i)\frac{2\arctan(z_i)}{\pi}, \tag{5.13}$$

instead. However, this too was tested by D'Errico [2006] and was found to be less accurate, especially if the converged value for z_i is large (taking the tan of a number very close to $\pi/2$ is much more sensitive to round-off error). Therefore, the transformation Equation 5.12 is preferable.

(In)equality Constraints

To handle the (nonlinear) (in)equality constraints, *penalty functions* can be used. A penalty function is an additional function, not originally part of the optimization statement, that adds a (usually large) value to the objective function. This value is a function of the amount of constraint violation at a certain trial point $\boldsymbol{x}$.

When the original problem

$$\begin{array}{rll} \min & f(\boldsymbol{x}) & , \boldsymbol{x} \in X, \\ \text{s.t.} & c_i(\boldsymbol{x}) & = 0,\, i \in E, \\ & c_i(\boldsymbol{x}) & \leq 0,\, i \in I, \end{array} \tag{5.14}$$

is rewritten in *penalized form* [Wolff, 2004],

$$\min_x L(\boldsymbol{x},\, \boldsymbol{\lambda}), \tag{5.15}$$

with

$$L(\boldsymbol{x},\, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{N} \lambda_i P_i(\boldsymbol{x}), \tag{5.16}$$

in which

$$P_i(\boldsymbol{x}) = P(f(\boldsymbol{x}),\, c_i(\boldsymbol{x})), \tag{5.17}$$

and λ_i the *penalty parameters*, the constraints $c_(\boldsymbol{x})$ are still included in the optimization problem, but now that problem is an unconstrained one. To quantify the amount of constraint violation per constraint function, define [Sun and Yuan, 2006, chapter 10]

$$c_i^{(-)}(\boldsymbol{x}) = c_i(\boldsymbol{x}),\, i \in E, \tag{5.18}$$

$$c_i^{(-)}(\boldsymbol{x}) = \max\{c_i(\boldsymbol{x}),\, 0\},\, i \in I. \tag{5.19}$$

and

$$C = \{c_i(\boldsymbol{x}) | c_i(\boldsymbol{x}) = 0,\, i \in E; c_i(\boldsymbol{x}) \leq 0,\, i \in I\}\,. \tag{5.20}$$

Then, a point $\boldsymbol{x}$ is only feasible if $\boldsymbol{x} \in C$. Using these definitions,

$$\text{if } c_i(\boldsymbol{x}) \leq 0 \text{ then } c_i^{(-)}(\boldsymbol{x}) = 0, \text{if } c_i(\boldsymbol{x}) > 0 \text{ then } c_i^{(-)}(\boldsymbol{x}) \neq 0. \tag{5.21}$$

that is, the penalty function is zero if all the constraints are satisfied, and nonzero when any of the constraints is violated. The specific functions $P_i(\boldsymbol{x})$ are normally problem dependent. One property that they all must have in common, is that the values of the penalty functions need to be *much* larger than $f(\boldsymbol{x})$ when the constraint violation is severe.

A variation of this method are the *interior-point penalty functions* or equivalently, *barrier functions* $h(\boldsymbol{x})$. Such functions work in much the same way as penalty functions, with the exception that they penalize the objective function on the *interior* of the search space, and have the additional property that

$$\lim_{c_i \to 0_-} = +\infty, \tag{5.22}$$

that is, the value of the (unconstrained) objective function approaches infinity when the trial solutions approach the boundaries of the feasible region. Such methods are much more strict in the sense that the barrier functions do not permit *any* constraint violation *ever* during the optimization. They also require that the initial estimate $\boldsymbol{x}_0$ is already a fully feasibly solution (e.g., all constraints are initially satisfied). However, many of the methods used in low-thrust trajectory optimization can generate initial solutions *close to* the feasible region, but it cannot be guaranteed that they will always be *inside* of it (except for exceedingly simple test cases) (see for example, the first estimates used by Sims and Flanagan [1999]). Therefore, in this thesis, only the aforementioned penalty function method is used.

5.4. Conditions for Optimality

5.4.1 Conditions for Unconstrained Problem

From elementary calculus, it is known that in a local minimum $\hat{\boldsymbol{x}}$ of unconstrained functions, the following conditions must hold simultaneously [Sun and Yuan, 2006]:

$$\nabla f(\hat{\boldsymbol{x}}) = 0, \tag{5.23}$$

$$\nabla^2 f(\hat{\boldsymbol{x}}) \quad \text{is} \quad \text{positive semi-definite.} \tag{5.24}$$

Note that these conditions only hold for functions $f(\boldsymbol{x})$ that are twice continuously differentiable on $\mathbb{R}^N$. For discontinuous or non-smooth functions, the conditions for minimality usually depend on the exact definition of the function $f(\boldsymbol{x})$.

The first of the conditions 5.23 – 5.24 determines whether the point $\hat{\boldsymbol{x}}$ is indeed a stationary point (a necessary condition for minimality), the second of these conditions is required to

determine whether the point $\hat{\boldsymbol{x}}$ is indeed a *minimum* (again, a necessary condition). Only if *both* conditions hold simultaneously is the point $\hat{\boldsymbol{x}}$ a local optimum.

In practice, the conditions given here serve to check whether an optimized point is indeed a local minimizer. Most functions, even quite simple ones, have first and second derivatives complicated enough to make solving Equation 5.23 analytically quite tedious or even downright impossible. Therefore, most of the time, these conditions are not used to *solve* the minimization problems, just to *check* the convergence of an algorithm, or *check* the end-result from some numerical method.

5.4.2 Method of Lagrange-Multipliers

The famous mathematician J.L. Lagrange showed that at a minimizer $\hat{\boldsymbol{x}}$ of a constrained function $f(\boldsymbol{x})$, the following conditions must hold [Sun and Yuan, 2006, Lemma 8.2.6]:

$$\nabla f(\hat{\boldsymbol{x}}) = \sum_{i \in E} \lambda_i \nabla c_i(\hat{\boldsymbol{x}}) + \sum_{i \in I} \lambda_i \nabla c_i(\hat{\boldsymbol{x}}), \tag{5.25}$$

where

$$\begin{aligned} \lambda &= 0, i \in E, \\ \lambda &\geq 0, i \in I, \end{aligned}$$

are the *Lagrange-multipliers*. This optimality condition is more easily expressed by introducing the *Lagrangian function* $L(\boldsymbol{x}, \boldsymbol{\lambda})$:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{N} \lambda_i c_i(\boldsymbol{x}). \tag{5.26}$$

The stationary points of the Lagrangian function then correspond to the minimizers $\hat{\boldsymbol{x}}_j$ of the original constrained function. Indeed, solving the equation

$$\nabla_{x,y,\lambda} L(\boldsymbol{x}, \boldsymbol{\lambda}) = 0$$

is known as the method of Lagrange multipliers. Note that it is (usually) not necessary to know the actual values of the λ_i to determine whether the optimality conditions hold or not. For example, the stationary point condition requires that the (vectorial) value of the objective function is a linear combination of the (vectorial) values of the constraint functions at the minimizer $\hat{\boldsymbol{x}}$ – a fact that can easily be checked without prior knowledge of the actual scalar multiples required for such a linear combination.

5.4.3 Karush-Kuhn-Tucker Conditions

H.W. Kuhn and A.W. Tucker generalized the method of Lagrange multipliers in 1951. They did this taking into account both inequality and equality constraints, while W. Karush derived a similar theorem in his M.Sc. thesis in 1939, but only for inequality-constrained functions.

Therefore, the following theorem is now known as either the *Kuhn-Tucker* or the Karush-Kuhn-Tucker (KKT)–conditions. They are a set of conditions imposed on the objective function and all of its (non-)linear constraints that hold at a certain point $\boldsymbol{x}$ if and only if that point is a local minimizer of the objective function, $\boldsymbol{x} = \hat{\boldsymbol{x}}$. They can be stated as [Sun and Yuan, 2006, Theorem 8.2.7]

$$\begin{aligned}
\nabla f(\hat{\boldsymbol{x}}) - \sum_{i=1}^{m} \hat{\lambda}_i \nabla c_i(\hat{\boldsymbol{x}}) &= 0 && (5.27)\\
c_i(\hat{\boldsymbol{x}}) &= 0, \quad \forall i \in E, && (5.28)\\
c_i(\hat{\boldsymbol{x}}) &\geq 0, \quad \forall i \in I, && (5.29)\\
\hat{\lambda}_i &\geq 0, \quad \forall i \in I, && (5.30)\\
\hat{\lambda}_i c_i(\hat{\boldsymbol{x}}) &= 0, \quad \forall i \in I, && (5.31)
\end{aligned}$$

Equation 5.27 is called the stationary point condition, because it corresponds to Equation 5.26 in the sense that

$$\nabla f(\hat{\boldsymbol{x}}) - \sum_{i=1}^{m} \hat{\lambda}_i \nabla = \nabla_x L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\lambda}}) = 0,$$

where $L(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\lambda}})$ is the corresponding Lagrangian. Furthermore, Equations 5.28 and 5.29 are called the feasibility conditions, Equation 5.30 the nonnegativity condition for multipliers, and Equation 5.31 the complimentary conditions. The latter conditions state that $\hat{\lambda}_i$ and $c_i(\hat{\boldsymbol{x}})$ cannot be nonzero simultaneously, or equivalently, that Lagrange multipliers corresponding to inactive constraints are zero [Sun and Yuan, 2006].

The KKT-conditions are a *necessary*, but not *sufficient* set of conditions for optimality. For that, also information of the second derivatives at the KKT-point $\hat{\boldsymbol{x}}$ is required. However, the KKT-conditions do provide a basic check on whether the optimized point $\hat{\boldsymbol{x}}$ is actually an optimal point.

6 Local Optimization Methods

The methods discussed in this chapter focus on finding *local* minimizers of the (constrained) objective function $f(\boldsymbol{x})$. All of these methods require an initial estimate that is assumed to be *close* to the true optimum of the objective function, and iteratively find better approximations to this "true" optimum while trying to satisfy all the constraints. Although there are many of such methods, only two commonly used methods are discussed here. They are the *Nelder-Mead algorithm* and *Sequential Quadratic Programming*.

6.1. Nelder-Mead Algorithm

The Nelder-Mead method[1] is named after its inventors, [Nelder and Mead, 1965]. It is a very elegant and (usually) efficient optimization algorithm, with the strong advantage that it does not require any information on the gradients of the objective function $f(\boldsymbol{x})$ or its constraints $c_i(\boldsymbol{x})$.

Nelder and Mead [1965] introduced the notion of a *simplex* – a polytope with $N+1$ vertices $\boldsymbol{y}_i$, $i = 1 \dots N$ in $\mathbb{R}^N$. For a function of two variables ($\boldsymbol{x} \in \mathbb{R}^2$), this means a simple triangle, or for a function of three variables ($\boldsymbol{x} \in \mathbb{R}^3$), a tetrahedron, etc (see Figure 6.1).

6.1.1 Unconstrained Optimization

In its simplest form, the following steps are iterated until these $N+1$ vertices have converged to a locally stationary point $\hat{\boldsymbol{x}}$ (adopted from Jeffrey et al. [1998], see also Figure 6.1):

Step 1 - Re-order Evaluate the objective function at each of the vertices $\boldsymbol{y}_i$, and re-order and re-label such that

[1] This method is also referred to as the *Flexible Polyhedron* method, *Simplex method* or *Amoeba method*, since the behavior and motion of the simplex somewhat resembles that of an Amoeba when it is moving.

$$f_1 \leq f_2 \leq \ldots \leq f_{N+1},$$

where $f_i = f(\boldsymbol{y}_i)$. Because the objective is to minimize the function $f(\boldsymbol{x})$, the point $\boldsymbol{y}_1$ is the *best* point, and $\boldsymbol{y}_{N+1}$ the *worst*.

Step 2 - Centroid Compute the *centroid* $\boldsymbol{C}$ of the best N points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ (all points except the worst). The centroid is defined as

$$\boldsymbol{C} = \sum_{i=1}^{N} \frac{\boldsymbol{y}_i}{N}.$$

Step 3 - Reflection Compute the value of the objective function $f_r = f(\boldsymbol{y}_r)$ at the *reflection point* $\boldsymbol{y}_r$:

$$\boldsymbol{y}_r = \boldsymbol{C} + \rho\left(\boldsymbol{C} - \boldsymbol{y}_{N+1}\right) = (1+\rho)\boldsymbol{C} - \rho\boldsymbol{y}_{N+1}.$$

where $\rho = 1$ in most implementations. If $f_1 \leq f_r \leq f_N$, accept the reflection point by replacing $\boldsymbol{y}_{N+1}$ with $\boldsymbol{y}_r$, and go to **Step 1**. Otherwise, continue with the next step.

Step 4 - Expansion If $f_r < f_1$ (better than the previous best), compute the *expansion point* $\boldsymbol{y}_e$ according to

$$\boldsymbol{y}_e = \boldsymbol{C} + \chi\left(\boldsymbol{y}_r - \boldsymbol{C}\right) = (1+\rho\chi)\boldsymbol{C} - \rho\chi\boldsymbol{y}_{N+1},$$

and compute the corresponding value of the objective function $f_e = f(\boldsymbol{y}_e)$. The value $\chi = 2$ is used in most implementations. If $f_e < f_r$, replace $\boldsymbol{y}_{N+1}$ with $\boldsymbol{y}_e$ and go to **Step 1**. Otherwise, simply accept $\boldsymbol{y}_r$ and go to **Step 1**.

Step 5 - Contraction If $f_r \geq f_n$, perform a *contraction* between $\boldsymbol{C}$ and the best of $\boldsymbol{y}_r$ and $\boldsymbol{y}_{N+1}$. There are two possible contractions:

Contract Outside If $f_N \leq f_r < f_{N+1}$ (f_r is still better than $\boldsymbol{y}_{N+1}$), calculate

$$\boldsymbol{y}_{co} = \boldsymbol{C} + \gamma\left(\boldsymbol{y}_r - \boldsymbol{C}\right) = (1+\rho\gamma)\boldsymbol{C} - \rho\gamma\boldsymbol{y}_{N+1},$$

and the corresponding function value $f_{co} = f(\boldsymbol{y}_{co})$. Usually, $\gamma = \frac{1}{2}$. If $f_{co} \leq f_r$, replace $\boldsymbol{y}_{N+1}$ with $\boldsymbol{y}_{co}$ and go to **Step 1.** Otherwise, go to **Step 6**.

Contract Inside If $f_r > f_{N+1}$ (worse than the previous worst), calculate

$$\boldsymbol{y}_{ci} = \boldsymbol{C} - \gamma\left(\boldsymbol{C} - \boldsymbol{y}_{N+1}\right) = (1-\gamma)\boldsymbol{C} + \gamma\boldsymbol{y}_{N+1},$$

and the corresponding function value $f_{ci} = f(\boldsymbol{y}_{ci})$. If $f_{ci} \leq f_{N+1}$, replace $\boldsymbol{y}_{N+1}$ with $\boldsymbol{y}_{ci}$ and go to **Step 1.** Otherwise, go to **Step 6**.

Step 6 - Shrink This step shrinks the simplex towards the best point $\boldsymbol{y}_1$. To accomplish this, evaluate the objective function at the points $\boldsymbol{v}_i = \boldsymbol{y}_1 + \sigma(\boldsymbol{y}_i - \boldsymbol{y}_1)$, $i = 2, \ldots, N+1$. The new simplex at the next iteration will consist of the points $\boldsymbol{y}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{N+1}$. The value $\sigma = \frac{1}{2}$ is used in most implementations.

The effect on the simplex of Reflection, Expansion, in– and outside contraction are shown in Figure 6.1. The shrink operation would simply generate a smaller triangle of identical shape, with $\boldsymbol{y}_1$ fixed.

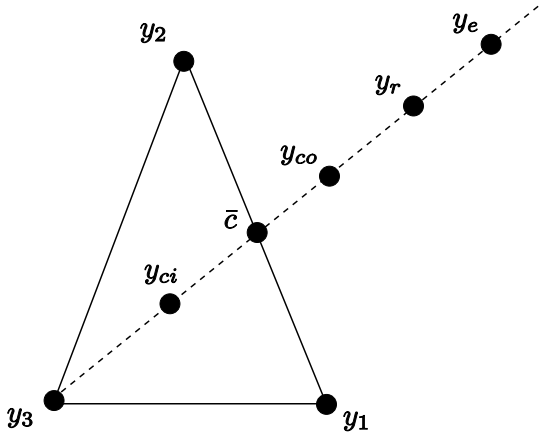


Figure 6.1 The four basic non-shrink operations used by the Nelder-Mead algorithm, for a two-dimensional problem. Adopted from [Price et al., 2002, Figure 1].

The parameters ρ, χ, γ and σ are confined to [Nelder and Mead, 1965]

$$\begin{aligned} 0 \leq \quad \rho \quad &\leq 1 && (6.1)\\ \chi \quad &> 1 && (6.2)\\ 0 \leq \quad \gamma \quad &\leq 1 && (6.3)\\ 0 \leq \quad \sigma \quad &\leq 1 && (6.4) \end{aligned}$$

In most implementations, the value of these parameters is taken to be

$$\begin{array}{llll} \rho = 1 & \chi = 2 \\ \gamma = \frac{1}{2} & \sigma = \frac{1}{2} \end{array} \qquad (6.5)$$

and the gain of changing their value is hardly noticeable [Singer and Singer, 2004].

Initial Simplex

The initialization of the first simplex can be done in several ways. Most implementations use

$$\begin{array}{ll} \text{if } \boldsymbol{y}_0(i) \neq 0 & \\ \qquad \boldsymbol{y}_i = \boldsymbol{y}_0 + 0.05\boldsymbol{y}_0(i)e_i & \\ \text{else} & \\ \qquad \boldsymbol{y}_i = \boldsymbol{y}_0 + 0.00025e_i & \end{array} \qquad (6.6)$$

and of course, $\boldsymbol{y}_{N+1} = \boldsymbol{y}_0$ itself. Here, e_i are the unit base vectors in the i^{th} dimension. These initial values have been empirically found to work satisfactorily in many situations. However, somewhat "less arbitrary" starting values, equal to [Wolff, 2004, and references therein]

$$\begin{aligned} \boldsymbol{y}_i^0 &= \boldsymbol{y}_0 + pe_i + \sum_{k=1}^{N} qe_k,\ i = 1 \ldots N, \\ p &= \frac{a}{N\sqrt{2}} \left(\sqrt{n+1} + n - 1 \right), \\ q &= \frac{a}{N\sqrt{2}} \left(\sqrt{n+1} - 1 \right) \\ &= p - \frac{aN}{N\sqrt{2}}, \end{aligned} \tag{6.7}$$

with e_i the unit base vectors and N the number of variables, were found to be slightly more efficient. The net effect of these second starting values is a slightly larger initial simplex, and that its initial vertices are not exclusively perturbations in the directions of the unit base vectors, but more randomly oriented around the initial estimate.

Ordering rule

[Jeffrey et al., 1998] give the following *tie-braking* rules in case some of the function values in **Step 1** are equal:

Non-shrink ordering rule When a non-shrink step occurs, the worst vertex $\boldsymbol{y}_{N+1}$ is always discarded. The accepted point created during iteration k, denoted by $\boldsymbol{v}^{(k)}$, becomes a new vertex and takes position $j+1$ in the vertices $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\ldots$, $\boldsymbol{y}_{N+1}$, where

$$j = \max_{0 \le \ell \le N} \left\{ \ell \,\middle|\, f(\boldsymbol{v}^{(k)}) < f(\boldsymbol{y}_{\ell+1}^{(k)}) \right\},$$

while all other vertices retain their relative ordering from iteration k.

Shrink ordering rule If a shrink step occurs, the only vertex carried over from simplex k to simplex $k+1$ is $\boldsymbol{y}_1$. Only one tie-braking rule is specified, for the case in which $\boldsymbol{y}_1^{(k)}$ and one or more of the new points are tied as the best point: if

$$\min \left\{ f(\boldsymbol{v}_2^{(k)}), \ldots, f(\boldsymbol{v}_{N+1}^{(k)}) \right\} = f(\boldsymbol{y}_1^{(k)}),$$

then $\boldsymbol{y}_1^{(k+1)} = \boldsymbol{y}_1^{(k)}$. Beyond this, whatever rule is used to define the original ordering may be applied after a shrink.

Improving Algorithm Efficiency

The efficiency of the basic implementation given by the steps above can be improved upon significantly. Singer and Singer [2004] performed an extensive research on the efficiency of the NM algorithm, resulting in three large improvements of the standard algorithm. These improvements are briefly discussed in the next three paragraphs.

Sorting In **Step 1**, the trial vectors $\boldsymbol{y}_i$ are sorted and re-labelled so that the sequence $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N+1}$ gives $f_1 \leq f_2 \leq \ldots f_{N+1}$. In most implementations, this sorting and relabelling is carried out by some language specific sorting algorithm *in each iteration*. Doing so indeed gives a shorter and simpler looking code, but it also implies that **Step 1** is performed in (at most) $(N+1)^2$ flops. However, for all non-shrink steps, a maximum of only *two* new trial vectors are inserted into the new simplex. Therefore, this inefficiency can be avoided by only sorting the trial vectors for the initial simplex, storing this specific sequence, and in all following iterations only *update* the original sequence instead of performing a complete re-sort. Updating takes (at most) $(N+1)$ comparisons, which is significantly faster than the original $(N+1)^2$ flops.

Centroid Usually, computing the location of the centroid $\boldsymbol{C}$ of the best N points is re-done every iteration. This calculation again requires $(N+1)^2$ flops to execute (when implemented naively). Because it is completely predictable where the vertices of the $(i+1)^{\text{th}}$simplex will be in every new iteration, the location of the original centroid $\boldsymbol{C}_i$ can be stored and updated according to

$$\boldsymbol{C}_{i+1} = \begin{cases} \boldsymbol{C}_i + \frac{1}{N+1}\left(\boldsymbol{y}^i_{N+1} - \boldsymbol{y}^{\boldsymbol{i+1}}_{\boldsymbol{N+1}}\right) & \text{without shrinkage} \\ \boldsymbol{y}_1 + \sigma\left(\boldsymbol{C}_i - \boldsymbol{y}_1\right) + \frac{1}{N+1}\left(\boldsymbol{y}^i_{N+1} - \boldsymbol{y}^{\boldsymbol{i+1}}_{\boldsymbol{N+1}}\right) & \text{with shrinkage} \end{cases} \tag{6.8}$$

which requires only (N+2) or (2N+4) flops to execute.

Convergence Conditions The most significant improvement can be accomplished in the evaluation of the algorithm's convergence criteria. In general, the following three criteria must be checked:

Domain Convergence Test This test checks whether the current simplex is equal in size or smaller than some predefined minimum size. This test is usually denoted as `term_y`, as it terminates based on the variation of the current vertices $\boldsymbol{y}_i$.

Function Value Convergence Test This test checks whether the function values f_i associated with all the current vertices $\boldsymbol{y}_i$ are closer together than some predefined minimum value. This test is usually denoted as `term_f`, as it terminates based on the variation in the current function values f_i.

Fail Test This test just compares the current amount of Function Evaluations (FE's) or iterations against some user-defined maximum amount. If this limit is exceeded, the algorithm *failed* to converge on the other two criteria and thus *failed* to find an optimum of the objective function.

The fail test is the simplest of all and is already very efficient – it takes only a few flops to compute. The other two tests however, are the real bottlenecks regarding the overall efficiency of the algorithm. There are a few variations for each, depending on the specific implementation.

There are essentially two variations of the `term_f` test used in practice:

$$\begin{array}{rcl} \texttt{term\_f} & := & 2 \cdot \frac{|f_{N+1} - f_1|}{|f_{N+1}| + |f_1|} \quad \leq \quad \texttt{reltol\_f} \\ \texttt{term\_f} & := & \max_{j \neq 1} |f_j - f_1| \quad \leq \quad \texttt{tol\_f}, \end{array} \tag{6.9}$$

where `reltol_f` is some relative error tolerance and `tol_f` an absolute error tolerance. It should be obvious that the first type uses only a few flops, whereas the second type uses $(N+1)$ flops, which is undesirable even when N is relatively small. However, the real bottleneck in practice is the `term_y` test. Tests used in practice generally fall in three groups:

$$\texttt{term\_y} := \max_{j \neq 1} \|\boldsymbol{y}_j - \boldsymbol{y}_1\|_\infty \leq \texttt{tol\_y}, \text{ or} \tag{6.10}$$

$$\texttt{term\_y} := \frac{\max_{j \neq 1} \|\boldsymbol{y}_j - \boldsymbol{y}_1\|_1}{\max\{1, \|\boldsymbol{y}_1\|_1\}} \leq \texttt{reltol\_y}, \text{ or} \tag{6.11}$$

$$\texttt{term\_y} := \|\boldsymbol{y}_{N+1} - \boldsymbol{y}_1\|_2 \leq \texttt{reltol\_y} \cdot \|\boldsymbol{y}_{N+1}^{(0)} - \boldsymbol{y}_1^{(0)}\|_2, \tag{6.12}$$

where again `reltol_y` indicates a relative tolerance and `tol_y` and absolute tolerance, $\boldsymbol{y}_i^{(0)}$ indicate the corresponding values of the initial simplex, and $\|\cdot\|_i$ indicates the i^{th}-norm of the indicated vector. The first two of these tests use $(N+1)^2$ flops, whereas the third uses $(N+1)$ flops, which is slightly more efficient.

However, this convergence test may be done by using only a *single* flop, by introducing the *volume ratio* $V(\bar{S})/V(S)$ of the previous and current simplex, respectively. It can be derived that this ratio is equal to

$$\frac{V(\bar{S})}{V(S)} = \begin{cases} \rho & \text{if transform = reflect,} \\ \rho\sigma & \text{if transform = expand,} \\ \chi & \text{if transform = contract inside,} \\ \rho\chi & \text{if transform = contract outside,} \\ \gamma^{N+1} & \text{if transform = shrink.} \end{cases} \tag{6.13}$$

Since the values for ρ, χ, γ and σ are constant, the factors given above can be pre-computed. When using these factors the volume ratio can be updated in a single flop. Rewriting this ratio into a linearized form yields

$$\texttt{term\_x} := \frac{LV(S)}{LV(S_{\text{init}})} \leq \texttt{reltol\_x}, \tag{6.14}$$

where $LV(S) = \sqrt[(N+1)]{V(S)}$ is a linearized volume and S_{init} denotes the initial simplex. It should be noted that taking $V(S_{\text{init}}) = 1$ does not alter the convergence test, but simplifies computations and enables also this linearized test to be executed in a single flop, provided the $(N+1)^{\text{th}}$-root of the constants in Equation 6.13 are pre-computed.

6.1.2 Potential Pitfalls

As can be seen, the NM algorithm is surprisingly simple. Also, it places almost no demands on the objective function $f(\boldsymbol{x})$; the algorithm's effectiveness remains unchanged even if the objective function is discontinuous, non-differentiable, non-smooth, non-convex, etc. Another advantage is that in every iteration, the objective function is only evaluated at most *three times* for non-shrink steps. Even in the initialization and shrink steps, the objective function is evaluated $(N + 1)$ times, which is (compared to having to compute numerical derivatives) still quite cheap if N is reasonably small.

There are a few drawbacks with this algorithm however. One is that it is not very well understood how or why this algorithm converges to a local minimizer. Jeffrey et al. [1998] managed to prove the convergence for a one dimensional case, but failed to do so even for simple two dimensional cases. It was also found by Mckinnon [1998] that the method can indeed converge to a point *that is not a minimizer* of that function. In short, the method lacks a proper convergence theory, which is still an open problem today. For this reason, it can not be *guaranteed* that the method indeed converges to a minimizer, although an overwhelming amount of experimental evidence mitigates this problem.

Despite its problems, the method is quite popular, mainly because it is so flexible and easy to use. For that reason, much research has gone into creating modifications of the basic implementation that do not share the same problems. Indeed, Price et al. [2002] and Burmen and Tuma [2007] designed modified NM algorithms for which the convergence could be *proved*. However, the performance of these algorithms (and the Nelder-Mead as a whole) was later also shown to decrease linearly with the problem's dimensionality.

Another potential problem is the simplex initialization and shrink step in case the number of variables N is very large. The objective function must be evaluated N times in these steps. Although shrink steps are made quite infrequently in practice, it can be quite costly and potentially even wasteful. Also, if derivatives of the function are *known*, they are not used in any way to the algorithm's advantage. In such cases, a different algorithm will most likely be more efficient.

In conclusion, the Nelder-Mead direct search algorithm is a powerful optimization algorithm for functions that are somehow "problematic" for gradient-based methods. Its only drawback seems to be that its performance decreases with the number of dimensions of the objective function. This renders it completely useless for optimization problems of high dimensionality, but quite usable for numerous other problems of low dimensionality, which are also quite common[2].

[2]An example of a problem that should definitely *not* be optimized with the Nelder-Mead algorithm, is the *method of patched micro conics*, to be discussed in section 13.2 – it is not unusual for this problem to have a few hundred dimensions. But, the low or high-thrust Multiple Gravity-Assist (MGA) problems, to be discussed in chapters 9 and 12, respectively, usually have $5 \sim 25$ dimensions; optimizing these problems with the Nelder-Mead algorithm is certainly worth a try.

6.1.3 Extension: Constrained Optimization

As described above, the Nelder-Mead algorithm performs an *unconstrained* optimization – no boundaries or constraints are imposed on the variables $\boldsymbol{y}_i$. This can however be accomplished by employing coordinate transformations and penalty functions, as described in section 5.3.1. Specific penalty functions that are generally applicable for the penalized optimization problem,

$$\min_x \quad L(\boldsymbol{x},\, \boldsymbol{\lambda}) \quad , \quad \text{with} \tag{6.15}$$

$$L(\boldsymbol{x},\, \boldsymbol{\lambda}) \quad = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i^k P_i(\boldsymbol{x}), \tag{6.16}$$

with λ_i^k adaptive penalty parameters that depend on the iteration k, have been suggested by Luersen et al. [2004] (this work was cast into a summarized and more readable form by Wolff [2004]). The penalty functions $P_i(\boldsymbol{x})$ are a set of adaptive linear penalty functions $P(\boldsymbol{x})$, defined as

$$P_i(\boldsymbol{x}) = \begin{cases} \max\{0,\, |c_i(\boldsymbol{x})|\}\,, & \text{if } i \in E, \\ \max\{0,\, c_i(\boldsymbol{x})\}\,, & \text{if } i \in I. \end{cases} \tag{6.17}$$

The values of the λ_i^k are to be determined in a two-step process:

1. Determine the vertex with the smallest value of the Lagrangian:

$$\boldsymbol{x}^{\text{best}} = \arg\left(\min_{x \in 1,\, \ldots,\, N+1} L(\boldsymbol{x},\, \lambda^k) \right) \tag{6.18}$$

2. If $L(\boldsymbol{x}^{\text{new}},\, \lambda^k) \leq L(\boldsymbol{x}^{\text{best}},\, \lambda^k)$ set

$$\lambda_i^{k+1} = \lambda_i^k + s_k \cdot P_i(\boldsymbol{x}),\; i = 1,\, \ldots,\, N+1, \tag{6.19}$$

where s_k is a positive step size, to be determined by

$$s_k = \begin{cases} s_L & \text{if } k \mod k_s = 0, \\ s_s & \text{otherwise}, \end{cases} \tag{6.20}$$

where s_s is some prescribed standard step size, $k_s > 0$ is a prescribed constant, and $0 < s_s < s_L$ a larger step size used for "sudden augmentation" – the (larger) step size s_L is used every k_s iterations, suddenly enforcing the constraints stronger than usual. This scheme for the adaptive penalty parameter balances the search between being strongly directed to the nearest feasible region (more greedy) and exploring the search space in a more global sense (less greedy).

6.2. Sequential Quadratic Programming

One of the most powerful (and popular) methods for solving nonlinear constrained functions, is Sequential Quadratic Programming (SQP). Essentially, this method approximates the objective function in a certain (small) region of its domain by a quadratic model. This localized, quadratic and constrained model can be solved analytically. The solution to this simplified model then defines the *step size* for the current iteration. Obviously, for convex functions that can reasonably be approximated by quadratic models, this methods requires very few iterations to find the local minimum, making this algorithm potentially very efficient.

In general, a Quadratic Program (QP) is an optimization problem with a quadratic objective function, and only linear constraint functions. QP's can easily be solved analytically. SQP approximates the given Non-Linear Programming (NLP) by a QP at the current decision vector, and steps towards the solution of this QP-subproblem at the next iteration. This process is repeated until convergence – the NLP is sequentially approximated by QP-subproblems. See Gockenbach [2003] for a very good introduction to this subject.

6.2.1 Lagrange-Newton Method

A minimization problem with only *equality* constraints has the form

$$\min_{x \in \mathbb{R}^N} \quad f(\boldsymbol{x}) \tag{6.21}$$

$$\text{s.t.} \quad c(\boldsymbol{x}) \quad = 0, \tag{6.22}$$

with $c(\boldsymbol{x}) \, (c_1(\boldsymbol{x}), \, \ldots, \, c_m(\boldsymbol{x}))^T \in \mathbb{R}^N$. For such a problem, a point $\hat{\boldsymbol{x}}$ is only a KKT-point of the function $f(\boldsymbol{x})$ if

$$\nabla_x L(\boldsymbol{x}, \, \boldsymbol{\lambda}) \quad = \quad \nabla f(\boldsymbol{x}) - \nabla c(\boldsymbol{x})^T \lambda = 0, \tag{6.23}$$

$$\nabla_\lambda L(\boldsymbol{x}, \, \boldsymbol{\lambda}) \quad = \quad -c(\boldsymbol{x}) = 0. \tag{6.24}$$

For a given iterate $\boldsymbol{x}_k$ at iteration k and a corresponding (estimated) Lagrange multiplier $\boldsymbol{\lambda}_k$, a Newton-Raphson step that solves Equations 6.23 and 6.24 is $((\delta \boldsymbol{x})_k, \, (\delta \boldsymbol{\lambda})_k)$, which satisfies

$$\begin{bmatrix} W(\boldsymbol{x}_k, \, \boldsymbol{\lambda}_k) & -A(\boldsymbol{x}_k) \\ -A(\boldsymbol{x}_k)^T & 0 \end{bmatrix} \begin{bmatrix} (\delta \boldsymbol{x})_k \\ (\delta \lambda)_k \end{bmatrix} = - \begin{bmatrix} \nabla f(\boldsymbol{x}_k) - A(\boldsymbol{x}_k) \lambda_k \\ -c(\boldsymbol{x}_k) \end{bmatrix} \tag{6.25}$$

where

$$A(\boldsymbol{x}_k) \quad = \quad \nabla c(\boldsymbol{x}_k)^T, \tag{6.26}$$

$$W(\boldsymbol{x}_k, \, \boldsymbol{\lambda}_k) \quad = \quad \nabla^2 f(\boldsymbol{x}_k) - \sum_{i=1}^{m} (\lambda_k)_i \nabla^2 c_i(\boldsymbol{x}_k). \tag{6.27}$$

These equations can be cast in algorithmic form by the following steps:

Step 1 Assume initial values for $\boldsymbol{x}_1 \in \mathbf{R}^n$, $\lambda_1 \in \mathbf{R}^m$, $\beta \in [0, 1]$, $\epsilon \geq 0$, $k := 1$ are all known.

Step 2 Compute $P(\boldsymbol{x}_k, \lambda_k)$ (see Equation 6.29). If its value is less than or equal to ϵ, convergence has been achieved; terminate the algorithm. Otherwise, use Equation 6.25 to find $(\delta\boldsymbol{x})_k$ and $(\delta\boldsymbol{\lambda})_k$. Set $\alpha = 1$.

Step 3 if

$$P(\boldsymbol{x}_k + \alpha(\delta\boldsymbol{x})_k, \boldsymbol{\lambda}_k + \alpha(\delta\boldsymbol{\lambda})_k) \leq (1 - \beta\alpha)P(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \tag{6.28}$$

go to **Step 4**. Otherwise, set $\alpha \leftarrow \alpha/4$ and repeat **Step 3.**

Step 4 Set

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha(\delta\boldsymbol{x})_k \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \alpha(\delta\boldsymbol{\lambda})_k, \end{aligned}$$

and go to **Step 2.**

Here, the penalty function $P(\boldsymbol{x}, \boldsymbol{\lambda})$ is defined as

$$P(\boldsymbol{x}, \boldsymbol{\lambda}) = \|\nabla f(\boldsymbol{x}) - A(\boldsymbol{x})\lambda\|_2^2 + \|c(\boldsymbol{x})\|_2^2. \tag{6.29}$$

This algorithm is described in much more detail in Sun and Yuan [2006, section 12.1]. There, several important proofs are outlined regarding the convergence properties of this algorithm. The most important results are that this algorithm indeed converges to the function's minimizer $\hat{\boldsymbol{x}}$, and that it does so *superlinearly*. Moreover, rewriting Equations 6.25 through 6.29 shows that the Lagrange-Newton method essentially sequentially solves QP's, which makes it a basic SQP method.

6.2.2 SQP, Reduced-Hessian Matrix Method

Although the Lagrange-Newton method already is a SQP method, it is only defined on *equality* constrained optimization problems. A general SQP method must be able to solve problems of the more general type, defined by Equation 5.6. Also, the gradient and Hessian matrices required in each iteration are usually rather expensive to compute, which leaves much room for improvement. Moreover, using Equation 6.29 as the method's penalty function leads to the *Maratos-effect* – the effect that even though an iterate produces a superlinear step, it will not get accepted by the demand ofEquation 6.28 which reduces the method's rate of convergence to linear or even sub-linear. These issues have been addressed by employing techniques such as the *watchdog-technique* or the modified Broyden-Fletcher-Goldfarb-Shanno formula's (BFGS)-method. Such techniques have lead to various different flavors of the SQP method (see [Sun and Yuan, 2006, Sections 12.2 through 12.8]. As always, each method has its own strengths and weaknesses. Here, only the *Reduced Hessian Matrix Method* will be elaborated.

Using the notation

$$
\begin{aligned}
W_k &= W(\boldsymbol{x}_k, \boldsymbol{\lambda}_k), && (6.30)\\
A_k &= A(\boldsymbol{x}_k) = \nabla c(\boldsymbol{x}_k)^T, && (6.31)\\
g_k &= \nabla f(\boldsymbol{x}_k), && (6.32)\\
c_k &= c(\boldsymbol{x}_k), && (6.33)\\
d_k &= (\delta \boldsymbol{x})_k, && (6.34)\\
\hat{\boldsymbol{\lambda}}_k &= \boldsymbol{\lambda}_k + (\delta\lambda)_k, && (6.35)
\end{aligned}
$$

Equation 6.25 can be rewritten as

$$
\begin{bmatrix} W_k & -A_k \\ -A_k^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} d_k \\ \hat{\boldsymbol{\lambda}}_k \end{bmatrix} = \begin{bmatrix} -g_k \\ c_k \end{bmatrix}. \tag{6.36}
$$

If the QR-factorization of A_k is written as

$$
A_k = \begin{bmatrix} Y_k & Z_k \end{bmatrix} \begin{bmatrix} R_k \\ \mathbf{0} \end{bmatrix},
$$

the linear system in Equation 6.36 can be written as

$$
\begin{bmatrix} Y_k^T W_k Y_k & Y_k^T W_k Z_k & -R_k \\ Z_k^T W_k Y_k & Z_k^T W_k Z_k & \mathbf{0} \\ -R_k^T & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} p_k \\ q_k \\ \hat{\boldsymbol{\lambda}}_k \end{bmatrix} = \begin{bmatrix} -Y_k^T g_k \\ -Z_k^T g_k \\ c_k \end{bmatrix} \tag{6.37}
$$

The quantities p_k, q_k and $\hat{\boldsymbol{\lambda}}_k$ are straightforward to isolate:

$$
\begin{aligned}
R_k^T p_k &= -c_k, && (6.38)\\
\left(Z_k^T W_k Z_k\right) q_k &= -Z_k^T g_k - Z_k^T W_k Y_k p_k, && (6.39)\\
R_k \hat{\boldsymbol{\lambda}}_k &= Y_k^T g_k + Y_k^T W_k \left(Y_k p_k + Z_k g_k\right). && (6.40)
\end{aligned}
$$

The last two rows in the matrix from Equation 6.37 allow for the possibility to approximate the matrix $Z_k W_k Z_k^T$, without computing it entirely. These last two rows give

$$
\begin{aligned}
\begin{bmatrix} Z_k^T W_k Y_k & Z_k^T W_k Z_k \\ -R_k^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} p_k \\ q_k \end{bmatrix} &= \begin{bmatrix} -Z_k^T g_k \\ c_k \end{bmatrix} && (6.41)\\
\rightarrow \begin{bmatrix} Z_k^T W_k Z_k^T \\ A_k^T \end{bmatrix} d_k &= \begin{bmatrix} -Z_k^T g_k \\ c_k \end{bmatrix} && (6.42)
\end{aligned}
$$

At each iteration, the search direction is obtained by solving the approximated, linear system

$$
\begin{bmatrix} B_k Z_k^T \\ -A_k^T \end{bmatrix} d_k = \begin{bmatrix} -Z_k^T g_k \\ c_k \end{bmatrix}. \tag{6.43}
$$

The matrix $B_k = Z_k W_k$ is then essentially an approximation to the system's Hessian. For reasons outlined in Sun and Yuan [2006, pages 555–556], it is best to use this symmetric B_k-matrix to replace $Z_k^T W_k Z_k$ in Equation 6.42, and a zero matrix for $Z_k^T W_k Y_k$. Doing so

will result in a stable, locally 2-step superlinearly convergent method, and allows the use of the BFGS formulas to update the approximation B_k at each iteration. Also, it makes sure that if the initial estimate is feasible (all constraints satisfied), all subsequent steps will also be feasible and the overall convergence will remain 2-step superlinear.

The BFGS-update in this context is

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{6.44}$$

where

$$\begin{aligned} s_k &= Z_k^T \left(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k \right), \\ y_k &= Z_{k+1}^T g_{k+1} - Z_k^T g_k. \end{aligned}$$

Two-step superlinearity means that the algorithm is superlinear, but only if *two* consecutive steps are considered; one step is "fast", the next one "slow". It has been shown that this is a property inherent to the reduced-Hessian matrix method, and can not be improved upon any further. Other SQP methods are somewhat more efficient in this respect, and also do not require the initial estimate to be feasible to guarantee superlinear convergence.

However, the reduced Hessian matrix method is still preferable. This is because the cost of computing the Hessian analytically, normally grows quadratically with the number of dimensions. Several of the optimization problems for this mission demand that SQP is used rather than the constrained Nelder-Mead algorithm, since their large dimensionality renders the latter algorithm useless (see sections 13.2 and 13.3). However, their large dimensionality also implies that computing their Hessian is extremely demanding, even more so when more constraints are added. If the Hessian were to be calculated for these problems at each iteration, it might even be more overall efficient to use the constrained Nelder-Mead algorithm instead. Therefore, reducing the computational burden of what essentially is a sidestep in the algorithm, will greatly benefit the efficiency of this SQP method.

7

Global Optimization Methods

The local methods discussed in the previous chapter are very efficient methods to find the minimum of an objective function. Especially when gradient information is available for the given objective function, local methods will nearly always find a minimum of the constrained nonlinear objective function with high precision and little FE's. But as the name implies, local methods will only find a minimum of the objective function near the provided initial value. Also, as is the case for SQP, derivative information is *required*, which implicitly demands the function, its first two derivatives, and the constraint functions, and their first two derivatives are all continuous. Also local methods assume that a good initial value can indeed be found.

However, for most optimization problems, just finding any minimum of the objective function is not enough; it is the *best* solution *possible* which is of interest. For many realistic cases, finding an initial value near this *global optimum* is an art in itself, or can inherently not be done. Also, many real-world objective functions are either discontinuous, noisy, un-differentiable, defined to return some *integer* function value, expensive to evaluate so that gradient information is even more expensive, or have some other combination of such "calculus-unfriendly" properties.

The only practical solution to find the global minimum of such functions is to use a *meta-heuristic optimization method*. Such methods use a whole *population* of trial solutions spread over the whole search space. On such populations, they apply some operators that are partly based on *chance*, and partly on the function values of the individual trial solutions, to generate a new, "better" population in the next iteration and thus slowly *evolve* towards the best possible function value.

One of the first ones to be invented (and still the most popular of them all) is the *genetic algorithm*. However, with the increasing availability of very powerful and cheap computational power, the development of such methods has taken a huge leap since then, which has resulted

in a true deluge of new methods. Of all of them, *differential evolution*, *adaptive simulated annealing* and *particle swarm optimization* were found to be the more powerful ones. These methods (and two more), will be the subject of this chapter.

7.1. Multi-start

The most *intuitive* way to find a global optimum, is to use a local optimization method for a large number of randomly generated initial values, spread out over the whole search space. For quite some time, this rather simplistic approach (the Multi-Start (MS)) was the only global optimization "algorithm" optimization specialists had at their disposal. It has the same drawbacks as the specific local method used for each trial value, with the added problem of the very large amount of FE's it requires.

In this thesis, this method is only included for completeness. It also serves as a "base method", to compare the efficiency of the other methods discussed in this chapter with. The Nelder-Mead (section 6.1) algorithm will be used as the local optimizer. Stepwise, this multi-start method works as follows:

Step 0 – Initialize a population of NP initial values $\boldsymbol{x}_i \in \mathbf{R}^N$, randomly chosen from the interval $[lb, ub]$.

Step 1 – Insert each initial trial $\boldsymbol{x}_i$ into the Nelder-Mead algorithm, which is allowed to perform at most $t_{\max}$ FE's.

Step 2 – The new population consists of the individuals $\boldsymbol{x}_i$, $i = 0 \ldots N$ that result from this optimization. Repeat from **Step 1**, until the total amount of allowed function evaluations $T_{\max}$ is exhausted.

Naturally, lb and ub are also N-dimensional vectors. Thus, this algorithm has basically only one control parameter – $t_{\max}$. Note that **Step 2** effectively is a *restart*; the local method is restarted around the best point it previously found. This is a crude way to avoid premature convergence during the optimization. When the simplex method converges, it means that the hyper-area of the simplex rapidly decreases around its local minimum, possibly excluding promising minima close to those points. However, when the algorithm is restarted around its best point, the hyper-area of the new simplex is simply reset to a much larger value, increasing the probability that the new simplex will be attracted to other minima in the neighborhood of the previous minimum.

When following the above procedure, it may be expected that for problems of high dimensionality, the performance of this multi-start method is quite poor, as is the case for the Nelder-Mead algorithm. The control parameter $t_{\max}$ remains to be optimized in some sense. This is the subject of section 7.8.

7.2. Genetic Algorithm

The most well-known meta-heuristic algorithm still is the GA. It is based on the principles of Darwinian evolution; "individuals" (trial solutions) have a certain "fitness" (their function value), and those that are fitter than others have a higher probability to procedure offspring. Reproduction is sexual, in the sense that the "genes" (components from the trial solution) from two parents are intermixed to produce children that are similar to but different from both parents. Also, the children are subject to "mutation" (a spontaneous, random change in one of its variables), that give them properties that were not present in either of its parents.

Since it is the oldest meta-heuristic algorithm, a vast volume of literature is available on this subject. Also, GA's can come in many different flavors, each with its own unique advantages and disadvantages. For this thesis, the classic works by Goldberg [1989] and Michalewicz [1996] are used. In contrast to more recent work, these classics describe the GA in a simple, straightforward and yet elaborate manner, and keep its operators limited to the simple crossover and mutation operators. Newer concepts such as variable population sizes, elitism, various kinds of immigration operators, etc. will not be considered here, for the simple reason that these operations make the basic GA much more complicated, and although they are powerful, they do not provide the power some of the other meta-heuristic algorithms described in this chapter have.

Step-wise, a basic GA works as follows:

Step 0 – Initialize an N-dimensional population of trial solutions with NP individuals, all randomly selected from the interval $[lb, ub]$. Also evaluate the objective function for every individual from this initial population.

Step 1 – Create a *mating-pool*, either through a *fitness-proportionate* or *tournament* selection process:

Fitness proportionate (roulette-wheel) selection:

- normalize all function values, by dividing each one by the sum of all function values:
$$\hat{fit}_i = \frac{fit_i}{\sum_{i=1}^{NP} fit_i}$$
- sort the population according to their normalized fitnesses $\hat{fit}$.
- Fill the mating pool with individuals ind_i, with *probability* $\hat{fit}_i$:
$$pool_i = ind(rnd < fit_j)$$

where rnd is a random number from [0, 1].

tournament-selection:

- select P individuals from the population at random. The one with the smallest function value, will be inserted into the mating pool.
- repeat this step until the mating pool is full.

Step 2 – Convert each individual to a string of numbers, e.g., concatenate the values of all decision variables for each individual to a single string of numbers (with a certain fixed precision per decision variable). That is, convert for example

$$[2.45152, 60.68948, 132.10592]$$

into

$$[002451520606894813210592]$$

Step 3 – Select two individuals at random from the mating pool, that will function as parents; say individuals 2 and 8.

Step 4 – Break both parents in two parts, at some random location CR:

$$\begin{aligned} parent1 &= [x_{18}, x_{28}, \ldots, x_{CR8}, \ldots, x_{N8}] \\ parent2 &= [x_{12}, x_{22}, \ldots, x_{CR2}, \ldots, x_{N2}] \end{aligned}$$

Step 5 – Let the parent's genes *crossover* at the point CR, with a certain probability p_{cross}, to create two children:

$$\begin{array}{ll} \texttt{if } rnd < p_{cross} \rightarrow & \\ \qquad child1 &= [x_{18}, x_{28}, \ldots, x_{CR2}, \ldots, x_{N2}] \\ \qquad child2 &= [x_{12}, x_{22}, \ldots, x_{CR8}, \ldots, x_{N8}] \\ \texttt{else} & \\ \qquad child1 &= parent1 \\ \qquad child2 &= parent2 \\ \texttt{end} & \end{array}$$

where rnd is a random number in [0, 1]. Repeat from **Step 3** until NP children have been created.

Step 6 – *Mutate* all children, with a certain (small) probability p_{mutate}. This selects a few random indices from ALL children (say, index M in the first child), and replaces the associated values with a new integer, randomly selected from the interval [0, 9]):

$$\begin{array}{ll} \texttt{if } rnd < p_{mutate} \rightarrow & \\ \qquad child1 &= [x_{18}, x_{28}, \ldots, x_{CR2}, \ldots, xM2, \ldots x_{N2}] \\ \qquad child2 &= [x_{12}, x_{22}, \ldots, x_{CR8}, \ldots, x_{N8}] \\ \texttt{end} & \end{array}$$

Step 7 – Convert the number-strings from the children back to individual values for the decision variables. If a *child* exceeds the boundaries set by *lb* or *ub*, simply re-initialize it altogether.

Step 8 – Evaluate the objective function for all of the children. For each four potential new members (two parents and their two children), the two individuals with the best fitnesses will become part of the new population.

Step 9 – Repeat everything from **Step 1** onward until the convergence criteria are met.

The original GA (described in [Goldberg, 1989]) used a *binary* representation of the population instead of the real-valued numbers assumed above. That is, each individual is represented by *bit strings* instead of real-number strings. Crossover and mutation are also carried out bit-wise, that is

$$\begin{aligned}
parent1 &= [10110011010010\ldots00110110001001]\\
parent2 &= [11001110110011\ldots00110011111001]\\
\text{crossover} &\rightarrow\\
child1 &= [10110011010010\ldots00110011111001]\\
child2 &= [11001110110011\ldots00110110001001]\\
\text{mutation} &\rightarrow\\
child1 &= [10110011010010\ldots00100011111001]\\
child2 &= [11011110110011\ldots00110110001000].
\end{aligned}$$

Whether to use binary representation or real numbers normally depends on the problem. Theoretically, the smaller the alphabet, the better the chances of convergence to the global solution. Indeed, [Michalewicz, 1996] found that for lower dimensionality (N is small) it is more efficient to use the binary representation. However, when the population size is enormous, it is overall more efficient to use real numbers, because the relatively costly conversion to binary and back is avoided. But such "rules-of-thumb" usually need to be tested for each new problem.

The GA as given above has two control parameters: the crossover-probability p_{cross} and the mutation probability p_{mutate}. Michalewicz [1996] advises $p_{cross} = 0.5$ and $p_{mutate} = 0.01$ and prefers real-number representation, whereas Goldberg [1989] prefers $p_{cross} = 0.25$ and $p_{mutate} = 0.02$ and a binary representation. Every other author seems to prefer different choices for these parameters (usually based on intuition, experience or other authors), so for this thesis they will have to be selected based on something more substantial. This will be done in section 7.8.

Since the tournament-selection method essentially does the same as the roulette-wheel selection, but then without the (costly) sorting operation, a binary tournament-selection procedure will be used; *two* randomly chosen individuals will compete with each other for a place in the mating pool. Optimizing this particular part of the GA will be left as a recommendation.

7.3. Differential Evolution

A meta-heuristic method that is quickly gaining popularity is DE. The DE algorithm first got attention by winning first place in the Genetic-type algorithms class, on the First International Contest on Evolutionary Optimization in May 1996, during the IEEE International Conference on Evolutionary Computation [Izzo et al., 2006]. A number of improvements were made in the following years, transforming DE into the "next-to-default" heuristic optimizer it is today.

A complete analysis on DE is given in the book written by the inventors of the method, [Price et al., 2005]. For the current section, reference is most frequently made to a newer book written on the subject, by Feoktistov [2006]. The simplest form of the algorithm is summarized in [Feoktistov, 2006, chapter 2]. In this simple form it is called *neoteric Differential Evolution*. The neoteric algorithm is summarized schematically in Figure 7.1. Stepwise, it is defined as

Step 0 – Initialize an N-dimensional population of trial vectors with NP individuals, all randomly chosen from the interval $[lb, ub]$.

Step 1 – For each individual, select 3 random other individuals ($\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$) from the population. One of these ($\boldsymbol{r}_1$) will serve as the *base vector* β, and the two others ($\boldsymbol{r}_2$ and $\boldsymbol{r}_3$) will produce the *differentiation vector* δ, where $\delta = \boldsymbol{r}_3 - \boldsymbol{r}_2$.

Step 2 – Create a new trial vector $\omega = \beta + F \cdot \delta$.

Step 3 – The new individual (*newind*) is

$$newind = \begin{cases} \omega & \text{if} \quad rnd \leq C_r, \\ ind & \text{otherwise.} \end{cases}$$

where *ind* is the original individual, and *rnd* is a random number taken from the interval [0, 1].

Step 4 – If any *newind* exceeds the boundaries set by *lb* or *ub*, simply re-initialize it.

Step 5 – When all the individuals have been processed this way, evaluate the objective function with the new population. If the function value of any new individual is less than that of the corresponding old individual, replace the old individual with the new one.

Step 6 – Repeat from **Step 1** until convergence is attained.

where *lb* and *ub* are also N-dimensional vectors. Thus, neoteric DE has only 2 control parameters – the constant of differentiation F, and the probability of crossover C_r.

Despite its name, DE has very little to do with Darwinian evolution, in contrast to the GA. Taking the *difference* between the two differentiation vectors is very reminiscent of taking a multi-dimensional derivative, aside from dividing by the length of the vector. But as the two

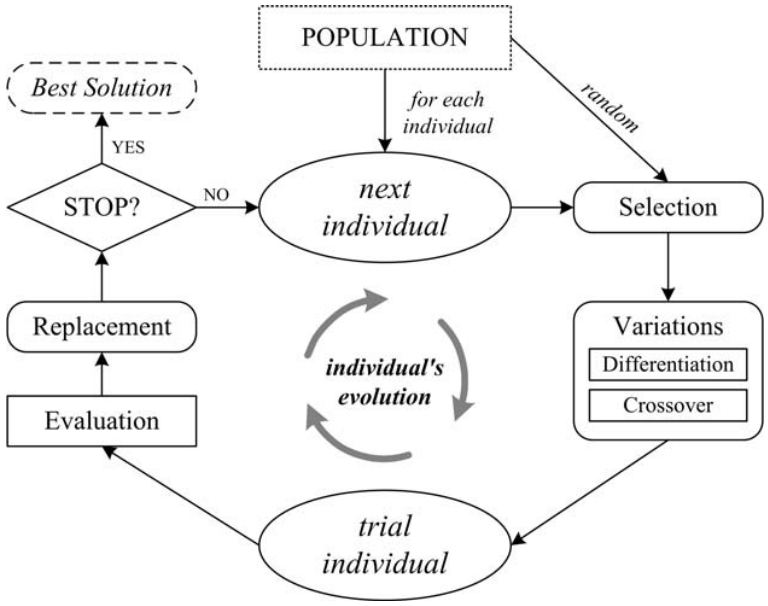


Figure 7.1 Schematic of the full evolution cycle of *Neoteric Differential Evolution*. This is Figure 2.2 from [Feoktistov, 2006].

vectors that form the differentiation vector are usually quite far apart (certainly not infinitesimally far), this "derivative" is more a global measure of how much the objective function changes *on average* over that interval. The derivative is computed at each iteration between two new, *randomly* selected vectors. So on average, the solutions will tend to go to where the average slope is zero, and the function globally minimal. Sometimes this operation is called the *global pseudo-derivative*, and it is the key to the power of the DE algorithm; it allows for a *directed evolution* to take place. Note also how elegant DE is compared to the bulky GA; it doesn't require converting the individuals to strings and back, it automatically implements a selection and mutation routine; it is simply *much* easier to implement.

One of the properties of DE is that the diversity of the population decreases with each iteration. In the neoteric form above, DE is too "greedy" to be a really robust algorithm, that is, it is likely to suffer from premature convergence. Neoteric DE brings an increasing amount of individuals close to the various local minima found, and a decreasing amount to the less promising points in the search space. A powerful exploit of this property is described in [Feoktistov, 2006, chapter 6]. It is called *transversal Differential Evolution*, which comes down to repeating **Step 2** and **Step 3** from the neoteric version above n *times*, with the exception that in **Step 3**, the function is evaluated for the new individual, and only if it is less than the previous new individual does the current new individual replace the previous new individual. Effectively, this allows each individual to evolve n times (instead of only once) before it is inserted into the new population, increasing the effect of the diversity of the whole population on one individual at each step.

This can be interpreted as making transversal rotation of the whole population at each iteration. This principle is displayed in Figure 7.2. Transversal DE adds one other control parameter to the process, namely n.

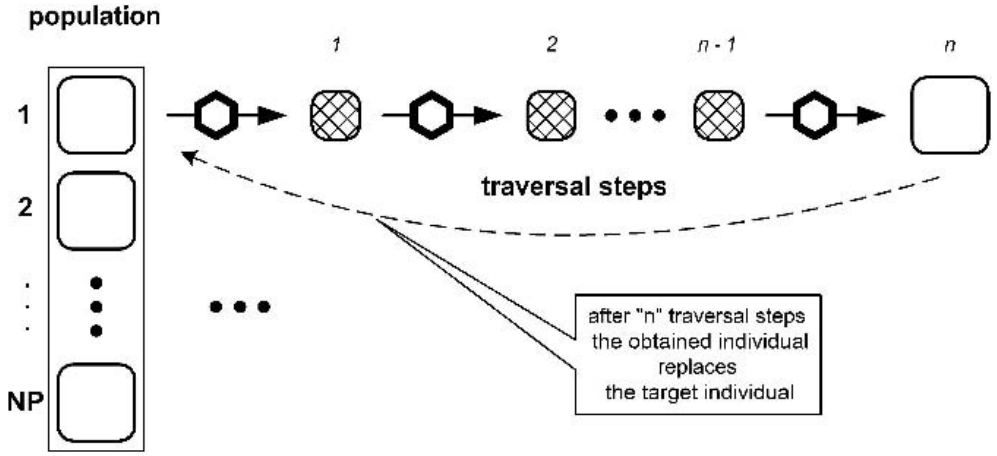


Figure 7.2 Schematic of the transversal steps taken in *transversal Differential Evolution*. These steps are taken in addition to the regular neoteric DE. This is Figure 6.3 from [Feoktistov, 2006].

DE is very transparent, easy to implement, and requires very few operations per iteration. Despite this simplicity, it is quite powerful. For example, in Feoktistov [2006, chapter 10] a comparison is made between DE and an SQP method (see section 6.2), using a *classical identification problem* as their benchmark problem. This problem has several nonlinear constraints, which are handled directly by SQP, and via penalty functions by DE. It was found that DE outperformed SQP on the account of the amount of constraint violations, minimum function value found and number of FE's – basically *all* of the important optimization criteria.

My personal experience with DE did however bring some disadvantages to light. The transversal operation usually is more demanding on the number of FE's before the population converges, albeit slightly and at an improved quality of the final solution. However, transversal also seems to run slower than the neoteric version. Especially for expensive objective functions, this is certainly something to keep in mind. Also, for the more challenging (and realistic) many-dimensional and discontinuous objective functions, DE was found to be somewhat too greedy. Although various remedies against this behavior have been discussed in the literature, their improvements are usually only marginal, so they are not considered here. To mitigate these effects, DE's three control parameters Cr, F and n must be optimized to give the best performance on the most problems. This is done in section 7.8.

7.4. Particle Swarm Optimization

Another heuristic method for global optimization is the so-called PSO method. Unlike GA or DE, it has its roots in *swarm intelligence*, exploiting the power of social interaction between individuals in a population. The PSO-method was first described in a conference on neural networks (see [Kennedy and Eberhart, 1995]), but has greatly evolved since then, and has found many more applications. The current section refers most frequently to a Master's thesis written on the subject, [Wilke, 2005]. PSO may be defined with the following steps

(see Figure 7.4) [Wilke, 2005]:

Step 0.1 – Initialize an N-dimensional population of trial vectors, with NP "particles" $\boldsymbol{x}_i$, where $i = 1, \ldots, NP$. These particles are all randomly selected from the interval $[lb, ub]$, where lb and ub are also N-dimensional vectors.

Step 0.2 – Assign a velocity $\boldsymbol{\nu}_i$ to each particle i, with random orientation and speed.

Step 0.3 – For each particle, define a small social network by selecting n unique numbers $j \in [0, NP]$; the particle's *neighbors* or *friends*.

Step 1 – Evaluate the objective function for all individuals, and save the *personal* optimum $lbest_i$ found by each particle, the *global* optimum $gbest$ found by the whole population, and the *neighborhood* optimum $nbest_i$ found by the neighbors of each particle.

Step 2 – Update the velocities, according to

$$\boldsymbol{\nu}_i \leftarrow \omega\boldsymbol{\nu}_i + c_l\left(lbest_i - \boldsymbol{x}_i\right) + c_g\left(gbest - \boldsymbol{x}_i\right) + c_n\left(nbest_i - \boldsymbol{x}_i\right)$$

Step 3 – Update the position of each particle i, by the procedure

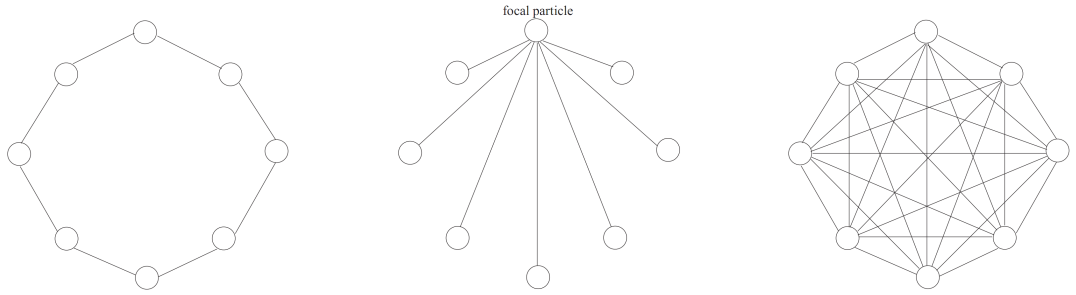
$$\boldsymbol{x}_{i+1} = \boldsymbol{\nu}_i + \boldsymbol{x}_i,$$

Step 4 – If any $\boldsymbol{x}_{i+1}$ exceeds the boundaries set by lb or ub, the particle will "bounce" back from the boundary, and thus always be inside the allowed search space.

Step 5 – Repeat from **Step 1** until convergence is attained.

It should also be noted that **Step 0.3** in the algorithm above has one more point to consider – the *network topology*. That is, the precise way in which the particles are connected to their neighbors. Several different topologies have been studied, most notably the *ring* topology, *star* topology and *fully-connected* topology [Reyes-Sierra and Coello-Coello, 2006]. These topologies are shown graphically in Figure 7.3. Various authors seem to agree that the fully-connected topology causes the swarm to converge very quickly, because the experiences of every particle are communicated to every other particle. However, because of this, it also has a much higher probability to converge to a local optimum, instead of the global one. This is contrasted by the ring topology, which generally takes much longer to converge and requires many more FE's, but is usually more robust. The trade-off between these extremes is the star topology, in which there is one particle (the *leader*) that is connected to some other particles and vice versa, but those other particles not to each other. Since the star topology has been established as being the better trade-off between fast convergence and robustness, this topology will be used throughout this work.

Aside from the population size NP, PSO requires no less than *five* control parameters – the *cognitive learning factor* c_l, the *cooperative factor* c_g, the *social learning factor* c_n, the *inertial constant* ω, and the *number of neighbors* n [Wilke, 2005, chapter 3]. All these factors combined aim to give a good balance between social and selfish behavior of the particles.



(a) Ring topology (b) Star topology (c) Fully-connected topology

Figure 7.3 The three most common network topologies used by the Particle Swarm Optimization community. The fully-connected topology (Figure 7.3(c)) usually has fast convergence rates but higher probability of converging to a local minimum, while the ring-topology (Figure 7.3(a)) behaves the opposite way. The best trade-off between these extremes seems to be the star-topology (Figure 7.3(b)). These are Figures 3, 4 and 5 from [Reyes-Sierra and Coello-Coello, 2006].

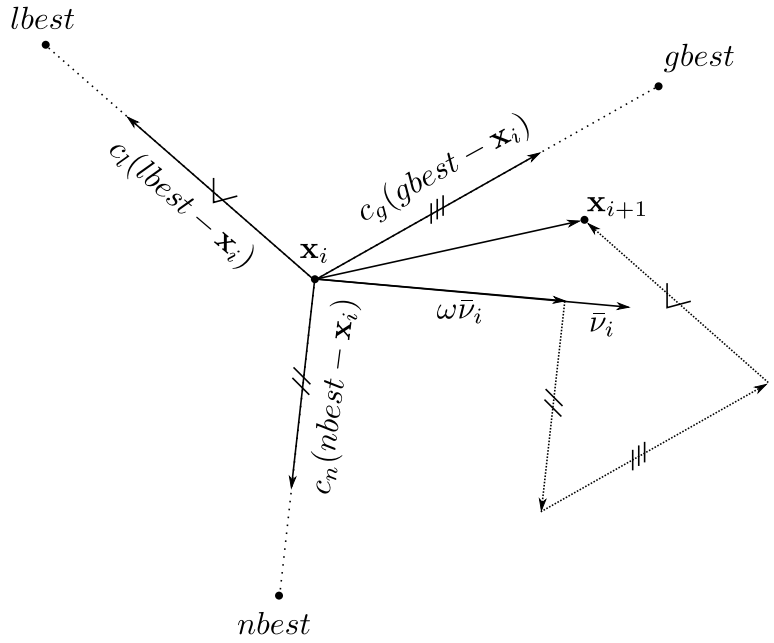


Figure 7.4 This shows how the velocity of the individual is directed selectively towards the global optimum $gbest$, the personal optimum $lbest$ found by the particle itself, and the best solution found by all of its neighbors, $nbest$. With carefully chosen parameters c_l, c_g and c_n, this gives a good balance between social and selfish behavior.

In [Wilke, 2005, chapter 3] it was also strongly suggested that some modifications to this standard formulation are necessary, particularly to each particle's speed. The position of each particle at the next iteration should not deviate *too* much, and also not *too* little, otherwise the search either stagnates, collapses into a line-search or is essentially reduced to a Monte-Carlo search strategy. This translates into defining an upper and lower boundary $|\boldsymbol{\nu}_{\max}|$ and $|\boldsymbol{\nu}_{\min}|$ for the speed. If these values are exceeded in one of the directions, the *unit vector* $\hat{\boldsymbol{\nu}}_i$

should be re-scaled to the minimum or maximum length.

However, [Wilke, 2005] concluded this based on objective functions that had more or less symmetric intervals for each dimension on which they are defined. This does not make the conclusion applicable *generally*; for problems where the range in each dimension differs greatly, taking the same minimum and maximum speed seems rather strange. For that reason (and to cut down on the number of control parameters for PSO), a minimum and maximum speed of respectively one-millionth and one-half of the interval length for each dimension will be adopted. Also, when either of these limits is exceeded, not the *entire* vector will be adjusted, but only the speed in that particular dimension. As this is a choice based mainly on practical considerations, finding the actual optimal values will be left as a recommendation. All other control parameters must be optimized to give the best performance on most problems. This is the subject of section 7.8.

7.5. Simulated Annealing

Another method is called Simulated Annealing (SA). Invented by Kirkpatrick et al. [1983], this method has its roots in statistical mechanics, particularly from the thermodynamic theory underlying liquids that solidify into crystalline form upon freezing. In such liquids, each atom is originally free to move randomly in the confining space (canister, bowl, can, etc.). When the positions of all atoms are measured at two different times a short interval apart from each other, the distance a single atom has moved depends on the temperature of the liquid; the higher the temperature, the higher the average speed per atom and thus the farther it can travel in any given amount of time. However, when the liquid cools, this motion is more and more inhibited and thus each atom is more and more likely to move to a lower energy state. When the temperature has dropped to a little over the freezing point of the liquid, random motion throughout the liquid is no longer possible and all the atoms freeze into the lowest energy state that they have managed to find.

This can very well be translated into an optimization method. Let the confining space represent the N-dimensional search space, atoms the individual trial solutions, and the energy state per atom its corresponding objective function value. Then there is a globally decreasing variable (the temperature T) that determines the probability of an atom performing a step to a higher energy state (higher function value). The lower the temperature, the less likely such a move will be and the more likely it is the atom will move to a lower energy state.

Although again many flavors of the SA algorithm exist, only the most basic one from [Kirkpatrick et al., 1983] is discussed here. In individual steps, this algorithm can be defined as

Step 0 Initialize a population of NP-trial solutions $\boldsymbol{x}_i$, $i = 1, \ldots, NP$, each taken randomly from the search space confined by $[lb, ub]$. Set the temperature equal to $T = T_0$, and evaluate the objective function for each of the atoms.

Step 1 Generate a new population, each member a small and random step away from the members of the original population:

$$pop_i = pop_i + \sqrt{T} \cdot randn(N) \text{ for all } i,$$

where $randn(N)$ is a vector of N random numbers, all drawn from the standard normal distribution. This is called the *Boltzmann generating scheme*. Evaluate the objective function for all the new positions of the atoms.

Step 2 Insert each new atom into the corresponding position in the original population, when its function value is either smaller than before, or when

$$rnd \leq \exp\left(\frac{E_{\text{prev}} - E_{\text{new}}}{T}\right),$$

whenever $E_{\text{prev}} - E_{\text{new}} < 0$. Here, E_{prev} and E_{new} are the atom's energies (function values) at the original and new positions, respectively, rnd is a random number from $\{0, 1\}$, and T is the current temperature.

Step 3 Apply the *cooling schedule:*

$$T \leftarrow cT, \quad 0 < c < 1.$$

Step 4 Repeat everything from **Step 1**, until the convergence criteria are met.

This SA has two control parameters: the cooling schedule constant c and the initial temperature T_0, respectively. The complete SA algorithm is represented schematically in Figure 7.5.

Strictly speaking, the original SA algorithm was not a population-based method; it only performed the above steps on a *single* atom. However, transforming the original SA into a population-based algorithm is a trivial task and can be expected to improve the algorithm's chances of finding the global minimum. Naturally, its two control parameters must be optimized to give the best performance on most problems. This is the subject of section 7.8.

7.6. Combining All Methods - The GODLIKE Algorithm

A meta-heuristic optimizer usually converges to the global optimum *with high probability*. This means that in some cases (for some problems, many cases) the global minimum will be missed, and the algorithm will converge to a local, sub-optimal point. This is a serious problem, which haunts every meta-heuristic optimization algorithm; *premature convergence*.

In the literature, several methods are applied to try and increase an algorithm's chances of finding the global optimum. The most commonly used "method" is to simply run the same optimization scheme several times. Although this indeed increases the chances of finding the optimum, it also implies having to cope with several times the (already quite high) computational cost. Another method is to make the optimization algorithm more sophisticated. Examples of this are using a variable population size in a GA to increase population diversity (see for example, [Lanzarini et al., 2000]). Or, running the same algorithm multiple times in

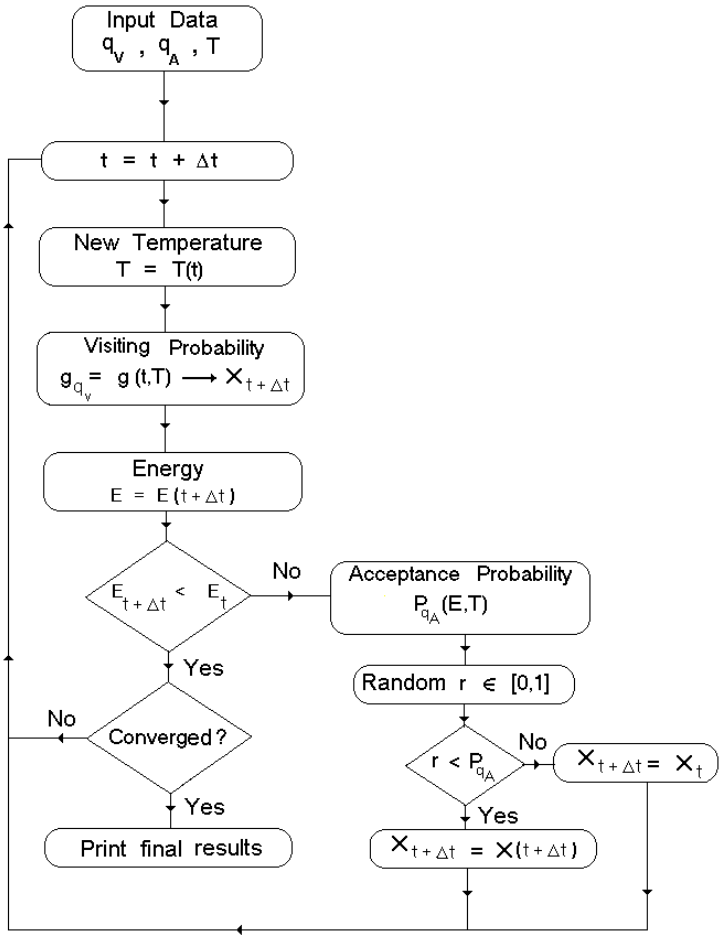


Figure 7.5 Schematic representation of the SA algorithm. Note that in this thesis, these steps will be applied to NP distinct trial vectors $\boldsymbol{x}_i$, $i = 0 \ldots N$, thus rendering it into a population-based method. From [Mundim, 2009].

parallel, with each algorithm operating on a fraction of the total population, while exchanging information between different instances. This again improves population diversity and uses some degree of specialization through isolation (see for example, [Blackwell and Branke, 2004]).

Many of these attempts to improve an algorithm seemed to complicate the existing algorithms much more than the actual improvement in the quality of its solutions would justify. One of the more successful attempts in this respect however, is the combination of several *different* meta-heuristic optimizers into one algorithm. Such an algorithm would split the base population up into smaller fractions, and perform a few iterations with *any* of the above algorithms on each fraction. The somewhat optimized population would then be given to some *other* heuristic optimizer, and the process is repeated until convergence. This was suggested in [Vinkó et al., 2007, and references therein] for example, and was indeed found to be more effective than a single algorithm for comparable computational cost.

In [Vinkó et al., 2007], only a combination of DE and PSO was used. For this thesis research,

a new algorithm is developed that is based on this idea. This algorithm is to *generalize* the idea of using multiple solvers simultaneously and collaboratively. It is thus a Global Optimum Determination by Linking and Interchanging Kindred Evaluators (GODLIKE).

7.6.1 Potential Advantages

GODLIKE uses a *random* meta-heuristic algorithm for a *random* amount of iterations on a *random* fraction of the base population, and passes this population to another, *random* meta-heuristic optimization algorithm (*linking*). After all optimizers have worked on all fractions of the base population (referred to as optimization *streams*), all populations are recombined and shuffled (*interchanged*), and everything is repeated until any of the termination conditions are met.

The basics of its operation are shown graphically in Figure 7.6. Thus, the GODLIKE algorithm is an attempt to improve the robustness of the meta-heuristic algorithms, and to do away with the need to fine-tune each algorithm for each new optimization problem, without having to resort to self-adaptation methods. Implemented as above, it is a sort of "umbrella" algorithm; able to quickly include (or exclude) other meta-heuristic, population-based methods in their simplest form.

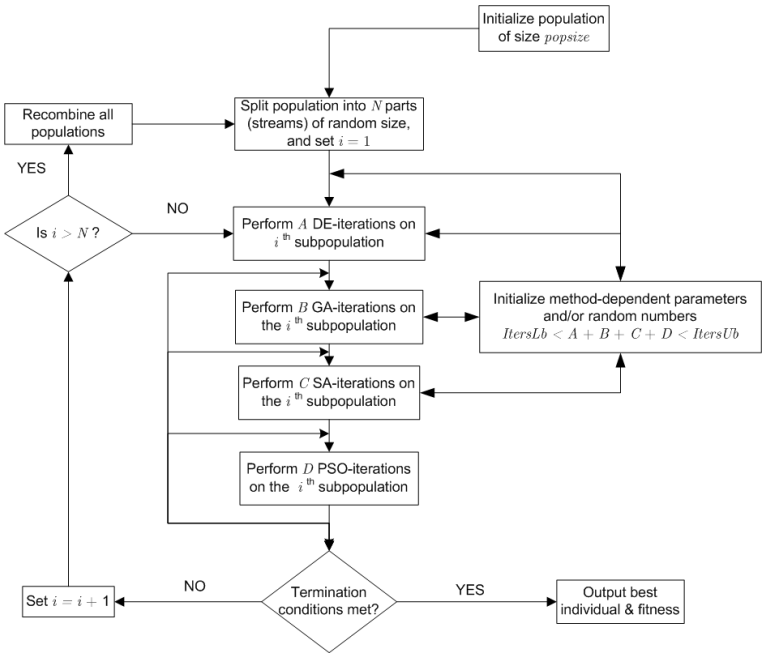


Figure 7.6 Schematic representation of the principles of the GODLIKE algorithm.

By using N separate optimization streams, and four different global optimizers sequentially, it is essentially equal to performing multiple consecutive optimizations automatically, which already improves the chances of finding the global optimum. Hopefully, also the weaknesses associated with each algorithm are negated by the strengths of another, while the strengths of all algorithms simply add up; note for example that when a GA is run in combination with one of DE, PSO or SA, the mutation operator in the GA is now also applied to individuals of

the other evaluator. Most importantly, the use of N separate and independent optimization streams makes *parallelization* straightforward to implement, greatly reducing the required computation time.

Note that the *interchange*-operator potentially destroys part of the convergence properties of either of the algorithms it uses, but that is exactly the intention – the convergence one of the algorithms is experiencing might be to a *local* optimum, while the others might be converging to the global solution or other local minima. By interchanging individuals between populations, GODLIKE automatically introduces *immigrants* into other populations that can provide alternative good solutions to the ones already being explored. These immigrants can steer the population into other, unexplored areas of the search space. By keeping the populations separate, also the principle of *isolation* is exploited automatically – portions of the search space will be thoroughly explored by one of the populations, while not affecting the other populations.

In summary, GODLIKE does not aim to make either of the algorithms more efficient in terms of the number of required FE's before convergence; rather, it will most likely require *more* FE's. Instead, *robustness* is aimed for, which remains to be tested. This testing is the subject of section 7.8.

7.6.2 Implementation and Control Parameters

The meta-heuristic optimizers discussed earlier this Chapter use a variety of schemes to improve the quality of their population at each iteration. All of these schemes have the inherent risk that some of the newly generated individuals fall outside the allowable region delimited by the vectors LB and UB. Without exception, all of the aforementioned meta-heuristic optimizers *re-initialize* those individuals until the whole population falls entirely inside the given search space. Although it is a simple and practical solution to address the problem of boundary violation, this particular method becomes increasingly inefficient when the feasible regions in the search space are either very small, have a large degree of fragmentation and/or are discontinuous; in those situations, the optimizer effectively changes into a Monte-Carlo scheme. Moreover, re-initializing individuals when they fall outside of the search space makes it harder for the optimizer to locate minima that lie on or close to the boundaries LB and UB. So, instead of re-initializing individuals that violate the bound constraints, GODLIKE will use the coordinate transformation defined in Equation 5.12 (and its inverse) to make the search space appear *boundless*. This way, it is inherently impossible for any individual to ever violate the bound constraints, no matter how it is generated.

How individuals are to be *linked* between a GA and a DE optimizer is straightforward, as these optimizers only require the population itself as their input. However, some "tricks" need to be used to be able to include the PSO and SA algorithms as well, as these optimizers require more information (the temperature for SA, and ν_i, $lbest_i$, $gbest$, $nbest_i$ for PSO). These "tricks" are kept simple for this thesis research for reasons of expediency, although a more elaborate (and likely, more robust) implementation is likely to be achievable. Any further improvements will be left as a recommendation.

Simulated Annealing

If a sub-population is optimized by the SA algorithm for some iterations, the temperature has dropped to a very low value; this is why the SA algorithm converges in the first place. However, the simplest way to give a new sub-population to the SA optimizer is to re-initialize the temperature for this new population to the value for T_0, defined by the user. This has the property that overall convergence of the GODLIKE algorithm becomes rather difficult, as individuals are first confined to a very small region (due to the low temperature), and at a next iteration are again free to fly across the whole search space. On the other hand, if the temperature for the new sub-population is kept equal to the low value at the end of the previous optimization, the effect is the opposite – the new individuals hardly have any freedom to explore regions they haven't already explored by virtue of one of the previous optimizers. The best solution to these problems seems to be to apply a certain degree of *re-heating*. That is, the temperature is re-initialized to a higher value than at the end of the previous optimization, but a value lower than the initial temperature T_0. As such, the amount of re-heating should depend on the amount of *interchanges* performed by the GODLIKE algorithm; in each consecutive *interchange* operation, the new initial temperature should be lower than the initial temperature used for the previous *interchange*. Thus, the amount of re-heating should be

$$T_0^{n+1} = \frac{T_0}{C_{\text{reheat}}^n}, \tag{7.1}$$

where C_{reheat} is a control parameter and n is the current *interchange*.

Particle Swarm

The *interchange*-operator is considerably more complicated for the PSO algorithm, as it requires much more information about the particles, which can not be provided (or kept track of somehow) by one of the other algorithms. However, although it seems simplistic, it can be argued that completely re-initializing all the velocities and parameters PSO requires after an *interchange* operation, is in fact the most beneficial. Because, when one of the other optimizers has brought a large portion of the population to a sub-optimal point, re-initializing the particle's velocities will automatically pull them out of that local optimum and prevent premature convergence. But this tactic also works when that point is the *global* optimum already, if also all other parameters are re-initialized (the local, neighbor and global best locations) – PSO's particles will automatically be led back to that same point after only a small amount of iterations (the value for *gbest* will then always stay the same). Therefore, a complete re-initialization of all PSO's parameters will allow PSO to be included in GODLIKE, and make GODLIKE more likely to converge to the global minimum.

There is one more issue to consider – the minimum and maximum amount of iterations that can be spent in each of the optimizers per iteration in GODLIKE – $ItersLb$ and $ItersUb$ respectively. These parameters determine the random number of iterations that will be executed per optimizer. It is better to define $ItersUb$ not as the maximum on the interval from which values will be taken, but rather the *total* amount of iterations that will be spent in all of the optimizers used in each stream combined. This allows better control over the maximum amount of *operations* allowable per stream, since this particular definition *guarantees* that exactly

$ItersUb$ will be executed in each stream – the only unknown is which algorithms will perform these iterations. Thus, at the start of each new GODLIKE-iteration and each new optimization stream, four random integers $A \geq ItersLb$, $B \geq ItersLb$, $C \geq ItersLb$, $D \geq ItersLb$ will be chosen, for which it holds that $A + B + C + D = ItersUb$.

Naturally it is quite hard to find those values for C_{reheat}, $ItersLb$, and $ItersUb$, that will give GODLIKE the best performance; these control parameters will each have to be optimized. This is the subject of section 7.8.

7.7. Convergence Criteria For Heuristic Optimizers

A natural question to ask when dealing with these meta-heuristic global optimization methods is when to stop the algorithm and determine that the algorithm has converged to a solution. Such *convergence criteria* are only intuitive with the MS (maximum FE's) or SA (temperature has fallen below the threshold) methods, but for DE, PSO and GA these criteria are much less obvious.

With these algorithms, no information is generally available about the neighborhood of the best solution found so far, and thus it is unknown whether that solution is a stationary point of the objective function $F(\boldsymbol{x})$ (the KKT-conditions do not apply). There are however many other convergence criteria that are used in practice. The most common criteria are

Maximum Iterations The heuristic optimizer is allowed to do its operations on the population for a fixed maximum amount of iterations. This is not really a "convergence" criterium, but more like a safeguard to prevent the algorithm from entering an infinite loop.

Maximum Function Evaluations The algorithm is allowed a certain limited budget for the amount of evaluations of the objective function. Essentially, this is more or less the same as the maximum iterations criterion for a given population size NP, however it is a more convenient way to express the computational budget.

Minimum Descent The function value of the best individual found so far is stored. If the algorithm is unable to find a solution that improves the stored best function value by some specified amount for some maximum amount of iterations, the algorithm has converged. That is, if the global minimum found so far is not improved by more than d_{min} for i_{max} iterations, convergence is said to have been achieved.

Achieve Function Value If additional information about the objective function is available, particularly on its value $F(\hat{\boldsymbol{x}})$ at the unknown global minimizer $\hat{\boldsymbol{x}}$, the algorithm can be forced to perform iterations until a solution is found with a function value a small tolerance away from the (known) minimal value $F(\hat{\boldsymbol{x}})$. Only when this is the case, will the counters for the other criteria be started. As the amount of FE's or iterations is essentially unbounded with this method, the algorithm may get stuck in an infinite loop in case the global minimizer is particularly hard to find.

As always, there is not *one* criterium that works best in *all* situations. In every published work on the subject, one of these criteria is selected by the researcher that seems to best fit his/her needs at that point. However, it seems quite reasonable to include *all* of these criteria in each of the algorithms. Especially setting default upper limits on the number of FE's and iterations will always prevent the algorithm from stagnating in some non-converging sequence. The minimum-descent and Achieve-Function-Value criteria can of course be combined easily, simply by including both criteria, but setting the default function-value-to-achieve to ∞; that way, the value-to-reach criterium can still be used, while the default criterium effectively is the minimum-descent one.

Also, for the GODLIKE algorithm, a convergence state is most effectively detected by looking at the descent in the function value of the global minimum found after two consecutive *interchange* operations. If that value has decreased less than some pre-defined tolerance ($TolFun$) for at least a certain amount of *interchanges* ($MaxInterchange$), it can be concluded that no further progress can be made by performing more *interchange* operations, and thus convergence has been achieved. Thus, for each individual optimizer, *and* for GODLIKE itself, the minimum-descent criterion seems to be the most effective way to detect convergence.

In an attempt to reduce the number of FE's, the same conditions will be checked in each of the algorithms used inside each optimization stream. That is, if the GA algorithm is allowed to perform 100 iterations, but it is unable to improve its global best solution more than $TolFun$ in more than $i_{\max}$ iterations, its population is passed to the next optimizer in the stream. Naturally, when the global minimum found by *any* optimizer *anywhere* inside the algorithm satisfies the Value-To-Reach, *and* that is the only requirement for a particular optimization, the GODLIKE algorithm will recognize this as convergence and return that particular individual as its solution.

7.8. Tuning The Control Parameters

7.8.1 Test Functions

When developing a new heuristic optimization algorithm, it is very convenient to have a simple function to test it on. Such a function must be relatively simple in order to evaluate it quickly, and yet be complicated enough so that it is a challenge to find its global minimum. In the history of the development of heuristic methods, many such functions have been devised and used throughout the optimization community.

Six such functions will be used to tune and test the different optimization algorithms. The first three are "classical" test functions. They are

- the Himmelblau Function
- the the Six-hump Camel Back Function.
- Rosenbrück's Banana Function

The last three can be found in [Mishra and Sudhanshu, 2006]:

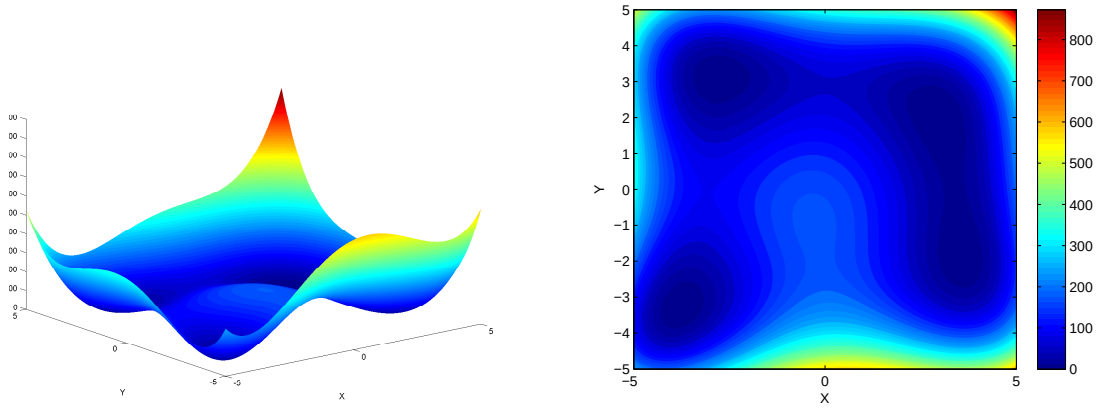
- the Test-Tube Holder Function
- the Sine-Envelope-Sine function
- The 6$^{\text{th}}$ Bukin Function.

These six functions are sorted according to their difficulty to find the global optimum; the minimum of the Himmelblau-function is hardly any challenge to find, while the global minimum of the 6$^{\text{th}}$ Bukin-function is next-to-impossible to locate. A short overview for each of these functions is presented below.

Himmelblau Function

The Himmelblau-function is defined as

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2. \tag{7.2}$$



(a) Function Overview (b) Filled contour plot

Figure 7.7 The Himmelblau function. The four global minima are located at $f(x, y) = f(+3.00, +2.00) = f(-2.81, +3.13) = f(-3.78, -3.2832 = f(+3.58, -1.85) = 0$.

A graph of it is given in Figure 7.7. As can be seen, this function is multimodal, but its four minima are all global:

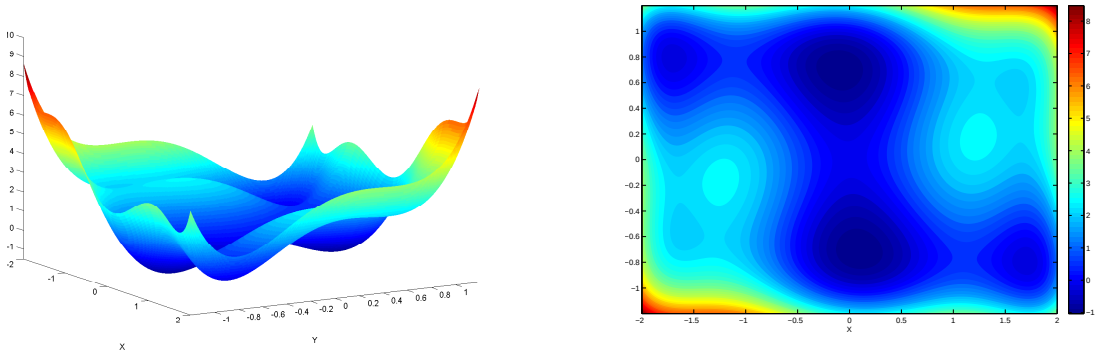
$$\begin{aligned} f(+3.0000\ldots, +2.0000\ldots) &= 0, \\ f(-2.8051\ldots, +3.1313\ldots) &= 0, \\ f(-3.7793\ldots, -3.2832\ldots) &= 0, \\ f(+3.5844\ldots, -1.8481\ldots) &= 0. \end{aligned}$$

Usually, function values are kept in the interval $[-5, 5]$.

"Six-Hump Camel Back" Function

The "Six-Hump Camel Back" Function gets its odd name from its six "humps" (local extrema). It is defined as

$$f(x, y) = (4 - 2.1x^2 + x^4/3)x^2 + xy + (4y^2 - 4)y^2. \tag{7.3}$$



(a) Function Overview

(b) Filled contour plot

Figure 7.8 The Six-Hump Camel function. Function has two global minima, $f(x, y) = f(\pm 0.09, \mp 0.72) = -1.03$.

A graph of it is given in Figure 7.8. This function is multimodal, and has four minima, of which only *two* are global. The global minima are

$$\begin{aligned} f(-0.0898\ldots, +0.7217\ldots) &= -1.0316\ldots \\ f(+0.0898\ldots, -0.7217\ldots) &= -1.0316\ldots \end{aligned} \tag{7.4}$$

Rosenbrück's "Banana" Function

The nickname "banana" in Rosenbrück's function comes from the way the function's isolines look when displayed on an old black-yellow CRT-computer display; like a bunch of bananas. The simple two-dimensional version is defined as

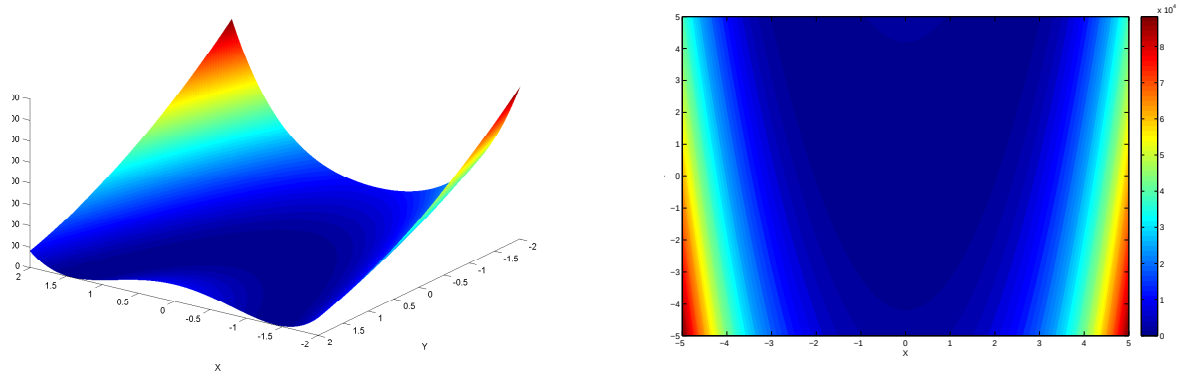
$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2. \tag{7.5}$$

A graph of it is given in Figure 7.9. This function has its single global minimum at

$$f(1, 1) = 0.$$

The main difficulty in finding the global optimum comes from the fact that it is located in a long, low, flat and narrow "valley", around which the function values reach very high values. Finding the valley usually is no problem for any optimizer, but finding the exact point is more difficult.

Usually, function values are kept in the interval $[-100, 100]$.



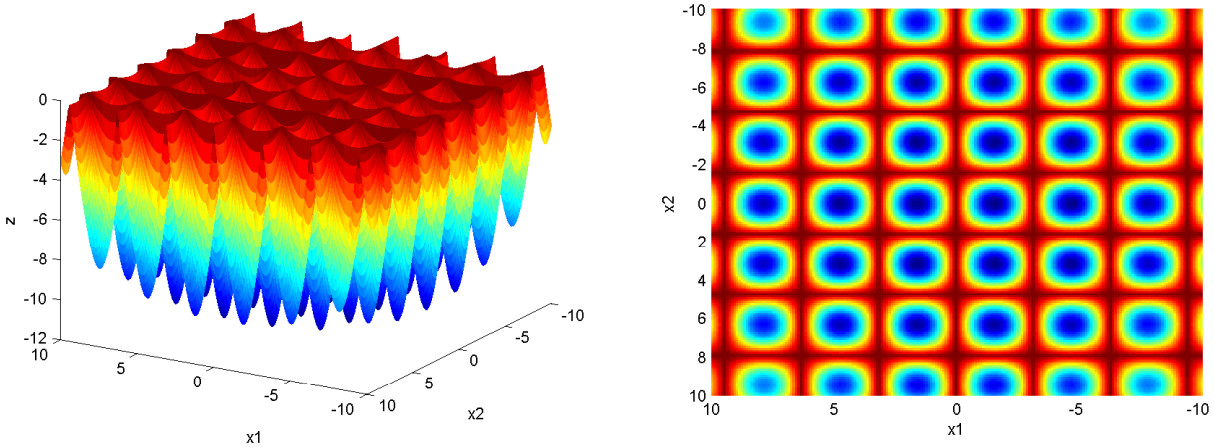
(a) Function overview (b) Filled contour plot

Figure 7.9 Rosenbrück's Banana function. Global minimum is at $f(x, y) = f(1, 1) = 0$.

Test-Tube Holder function

The somewhat odd name for this test-function is self explanatory when looking at its graph (Figure 7.10). It is defined as

$$f(x, x_2) = -4\left|\sin(x_1)\cos(x_2)e^{\left|\cos((x_1^2+x_2^2)/200)\right|}\right| \tag{7.6}$$



(a) Function Overview (b) Filled contour plot

Figure 7.10 The Test-tube holder function has its single global minimum at $f(x, y) = f(-\pi/2, 0) = -10.8723$.

On $[-\infty, \infty]$ it has an *infinite* amount of local minima, but only *one* global minimum:

$$f(x_1, x_2) = f(-\pi/2, 0) = -10.8723 \tag{7.7}$$

Usually, function values are kept in the interval $[-10, 10]$.

Sine-Envelope-Sine function

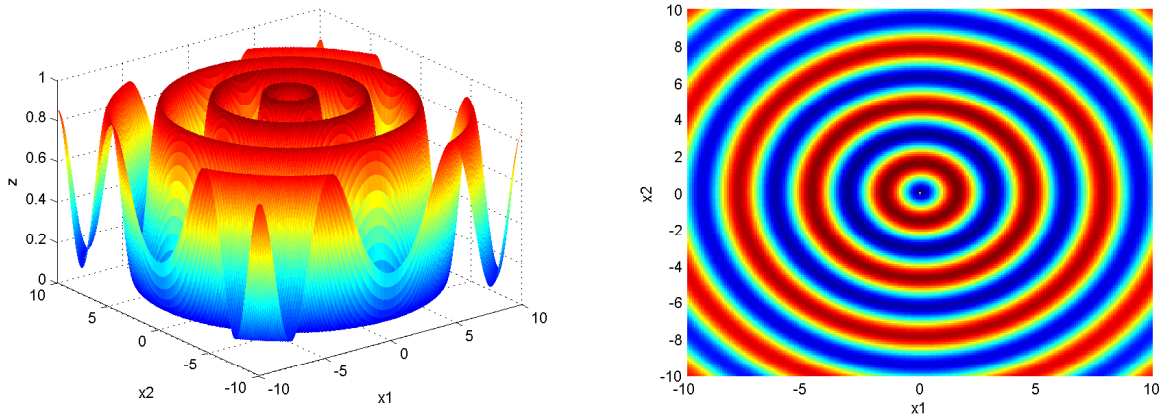
This function is a squared-sine function, but then in all directions – a squared-sine function circled around the origin. Also, the further away from the origin, the smaller the amplitude of the sine function and the higher the average function value. The single global minimum lies at the origin and thus is surrounded by relatively high function values, which makes this function more difficult to optimize. It is defined as

$$f(x_1,\, x_2) = \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - \frac{1}{2}}{\left(1 + 0.001\sqrt{x_1^2 + x_2^2}\right)^2} + \frac{1}{2} \tag{7.8}$$

A graph of it is given in Figure 7.11. It has its single global minimum at

$$f(x_1,\, x_2) = f(0,\, 0) = 0. \tag{7.9}$$

Usually, function values are kept in the interval $[-100,\, 100]$.



(a) Function Overview (b) Filled contour plot

Figure 7.11 The Sine-Envelope-Sine function has $f(x_1,\, x_2) = f(0,\, 0) = 0$.

6th Bukin Function

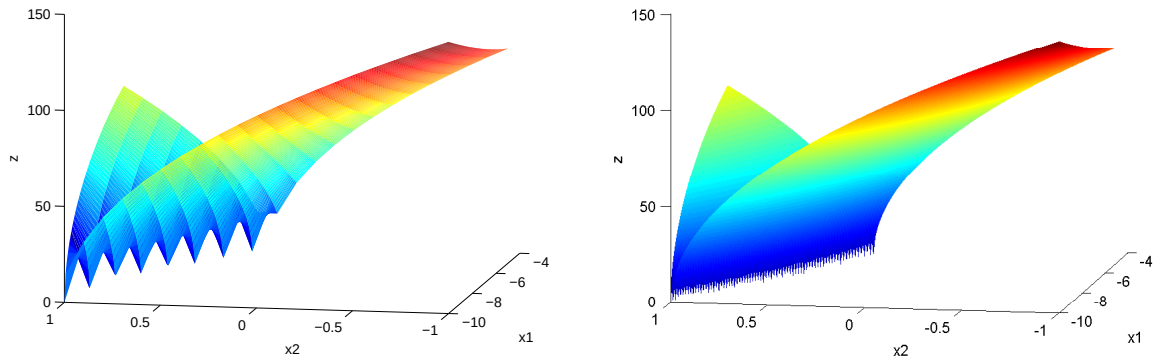
This function is by far the most difficult function to optimize. Its global minimum is surrounded by an enormous amount of ever decreasing local minima; so much in fact that the function is nearly fractal. It is defined as

$$f(x_1,\, x_2) = 100\sqrt{|x_2 - 0.01x_1^2|} + 0.01\,|x_1 + 10|\,. \tag{7.10}$$

A graph of it is given in Figure 7.12. Note that the local minima can not even be displayed properly. This is still true when the plot is drawn in a space confined to a square with sides $1{\times}10^{-16}$ symmetrically around the global minimum. Its single *global* minimum is located at

$$f(x_1,\, x_2) = f(-10,\, 1) = 0. \tag{7.11}$$

Function values are to be kept in the interval $x_1 \in [-15,\, -5]$, $x_2 \in [-3,\, 3]$.



(a) 1,000 plotting points

(b) 100,000 plotting points

Figure 7.12 The 6$^{\text{th}}$ Bukin function. Note that the amount of local minima displayed increases when the amount of plotting points is increased – the function cannot be properly displayed due to the extremely large amount of points required to do so. This is a coarse indication of the near-fractal behavior of this function. It has its single global minimum at $f(x_1, x_2) = f(-10, 1) = 0$, surrounded by a (countably-infinite) amount of local minima.

7.8.2 Algorithm Performance

To see how the different algorithms perform on these functions for different values of their control parameters, a whole range for each control parameter is taken, and the algorithm is run with those control parameters on all six test functions, 20 times. When the algorithm converges, the global minimum it returns is compared to the real and known value of the global minimum, and if it lies within $1{\times}10^{-3}$ of said value, the score for those particular parameters is increased by 1. Thus, a maximum score of $6 \times 20 = 120$ is possible. Then, the whole procedure is done all over again, for slightly different values of the control parameters, and this is continued until all values in all ranges of all control parameters have been scored.

All scores thus found are then multiplied by 100/120 to express the *success-rate* of the algorithm as a percentage. Thus, if the success-rate is for example 97.23, the algorithm can be expected to successfully find the global minimum of some function about 97.23% of the time.

The parameters used in this small investigation are

- Population size is $NP = 100$,
- Max. FE's fixed to 50,000 (total of maximum one million evaluations per test),
- Min. descent $d_{\min} = 0$ for at least $i_{\max} = 100$ iterations,
- No maximum on the total amount of iterations.

Multi-start

Since the total number of FE's, $T_{\max}$, is fixed for all algorithms at 5×10^4, the only tuning parameter for this algorithm is the number of maximum FE's per individual per iteration $t_{\max}$. The number $t_{\max}$ was varied from 10 to 1000, taking steps of 30. The resulting data from this experiment can conveniently be plotted in a simple two-dimensional graph, as is shown in Figure 7.13.

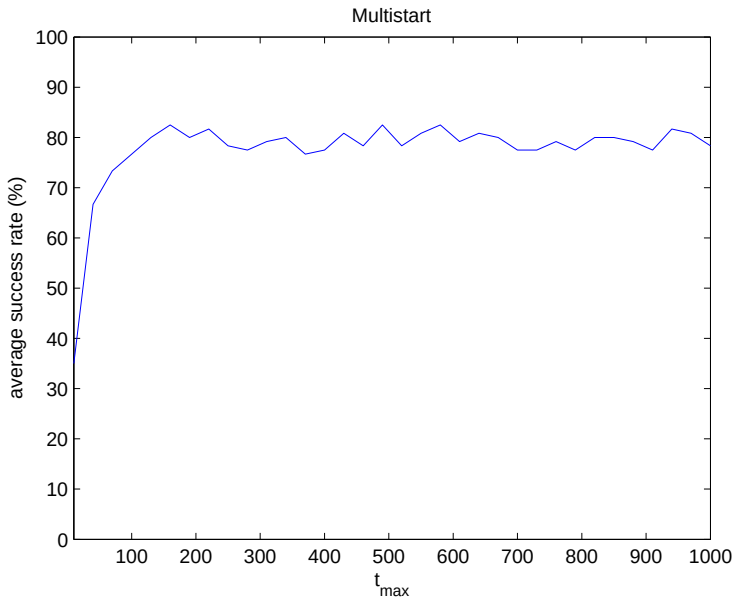


Figure 7.13 Optimizing the only control parameter for the MS-method. Note that the performance of this algorithm is more or less constant, irrespective of the value of the control parameter. Note also that this algorithm outperforms GA and SA for the tested functions.

Note that its performance is surprisingly good; the best overall performance ($\sim$83%) indicates that the optima of *all* test functions (except the 6$^{\text{th}}$ Bukin function, $100 - 100/6 \approx 83$) were *always* found by the multi-start algorithm. This makes it into quite a powerful alternative method to be seriously considered for real-word optimization problems. Keep in mind however, that as the problem's dimensions grow, the performance of this algorithm will most likely decrease to unacceptable levels. But it can certainly be used to optimize problems of lower dimensions. This makes it quite relevant for the optimization of the current mission, because the number of decision variables involved for MGA problems (the start date, and the transfer times between the planets) are usually not so numerous.

Also note that its performance is relatively insensitive to the amount of restarts used; at least for the test functions used, almost *any* value of $t_{\max}$ gives the same performance, save for very low values. This might be due to its initialization procedure – a constrained version of MATLAB®'s `fminsearch` was used for the above test, which uses Equation 6.6 to initialize its first simplex). Using Equation 6.7 instead (which results in a larger initial simplex) might give different results. Because these results are so unexpected, the above test was re-done with the different initial values from Equation 6.7. The results are shown in Figure 7.14.

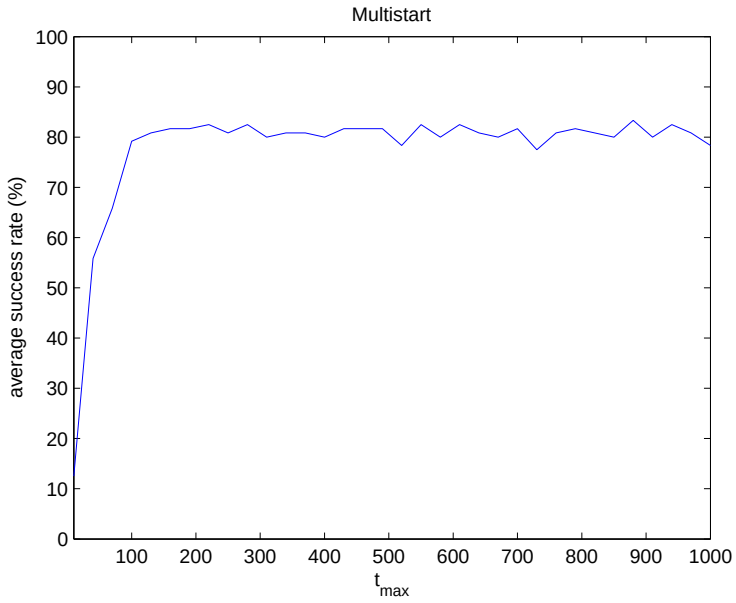


Figure 7.14 Optimizing the only control parameter for the multi-start method, this time with the different starting simplex as defined in Equation 6.7. Note that the performance of this algorithm is more or less constant, irrespective of the value of the control parameter. Again, this algorithm outperforms GA and SA for the tested functions.

Comparing this second figure to the first one, it can be concluded that the globalized simplex method performs equally well for both starting values, and that performance remains fairly insensitive to the number of simplex restarts. To maintain some degree of control over the number of FE's per individual, and to prevent single individuals from consuming large fractions of $T_{\max}$, using a constant value of $t_{\max} = 200$ seems like a well-balanced choice.

Genetic Algorithm

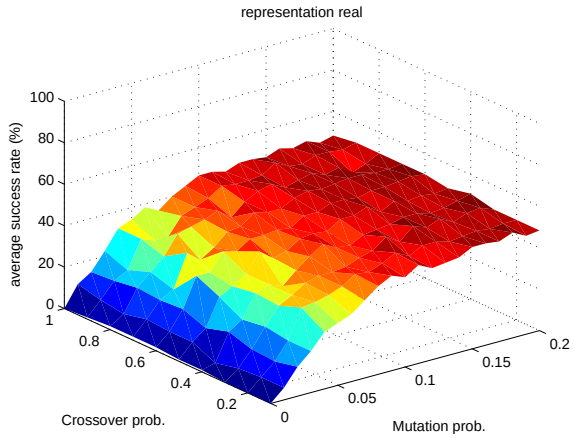
For the GA there are two control parameters; p_{mutate} and p_{cross}. The value of p_{mutate} was taken from $p_{mutate} = [0.00, 0.01, \ldots, 0.2]$, and the value for p_{cross} was taken from $p_{cross} = [0.1, 0.2, \ldots, 1]$. Further, both real-valued coding and binary-coding have been tested.

For the binary coding, initially, 52 bits bits per decision variable have been used to make sure the maximum resolution was maintained throughout the test. Later, 100 bits were used to make absolutely sure the resolution of the solutions was not impaired by the associated fixed-bits cut-off.

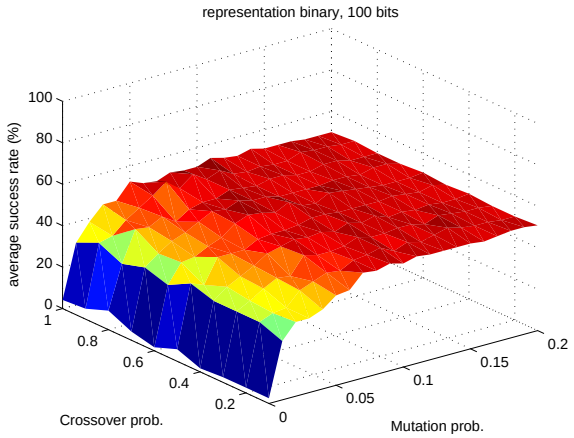
Also, initially, the real-representation was applied as described previously. In a second test, the decision vectors were crossed over and mutated without first "stringing" the decision variables together, i.e.,

$$
\begin{aligned}
parent_1 &= [2.20371, 5.70434, \ldots, 3.0172] \\
parent_2 &= [3.05718, -19.2758, \ldots, 0.00248] \\
\text{crossover} &\rightarrow \\
child_1 &= [2.20371, -19.2758, \ldots, 0.00248] \\
child_2 &= [3.05718, 5.70434, \ldots, 3.0172],
\end{aligned}
$$

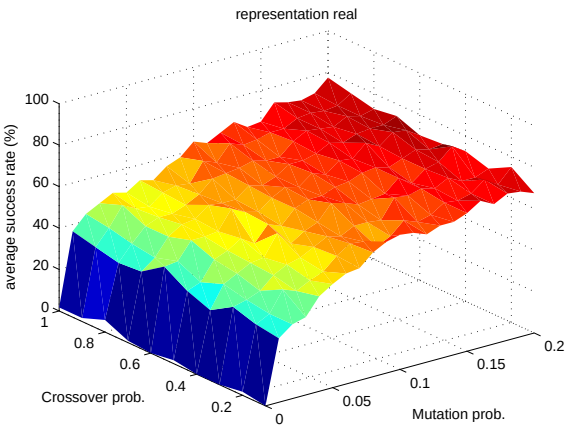
instead of first converting them into a single long string. The results of all these tests are shown in Figure 7.15.



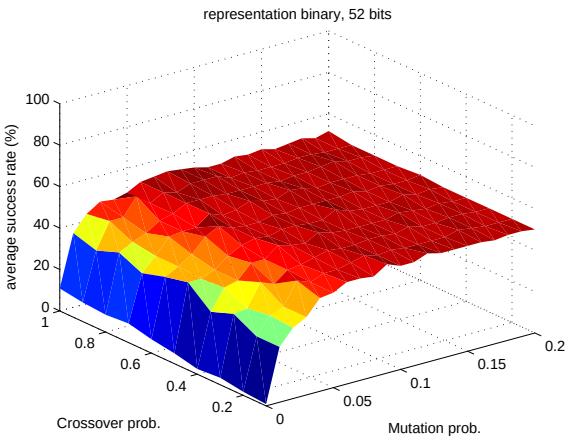
(a) GA (Real Coding, un-stringed)



(b) GA (Binary Coding, 100 bits.)



(c) GA (Real Coding, stringed)



(d) GA (Binary Coding, 52 bits.)

Figure 7.15 Tuning the GA's control parameters. Both real-coding and binary-coding have been tested. Note the increasing influence of larger values of the mutation probability for the real-coded test. This is indicative the problem's dimensionality (all test-functions are 2-dimensional); this is rather low and does not really allow accurate solutions with real-coding. All points forming the surfaces are the averaged success rates of 20 individual runs on all 6 test functions, using the indicated values of the control parameters.

It can readily be concluded from these figures that the GA has a serious problem when it comes to converging to accurate solutions. It seems to be able to find *some* of the global minima, but at least for the test functions used, it misses the global minima about 55% of the time. Some further testing reveals that in accordance with the conclusions of many researchers, GA brings the solutions *close* to the global minimum, but getting them *closer* is quite difficult. In this test, a seemingly reasonable closeness of 1×10^{-3} was used, which already proved too demanding for the GA. Some further experimentation showed that the first three test functions pose no problem for the GA, but the optima of the latter three functions (the test-tube holder, sine-envelope-sine and the 6^{th} Bukin functions) could almost never be found using a GA. This fact seems to be independent from the specific coding being used (binary or real) and from the GA's control parameters.

Based on these results, it appears best to use a relatively high value for the mutation probability, a high probability for crossover and a stringed real-coding. To test the GODLIKE algorithm, the values $p_{mutate} = 0.2$, $p_{cross} = 0.95$ and stringed real-coding will be maintained.

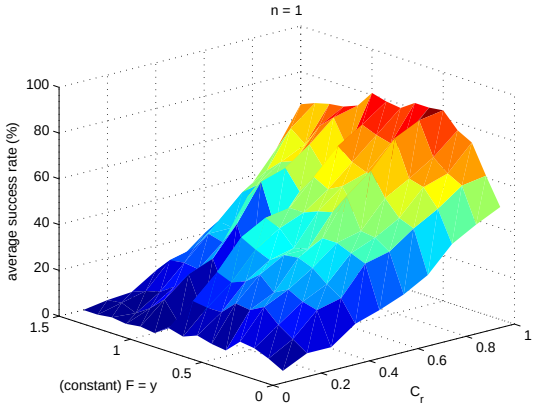
Differential Evolution

For the DE algorithm there are 3 control parameters: F, n and C_r. Feoktistov [2006] also suggests to randomly select the parameter F from an interval symmetric about 0, instead of just keeping it constant. To investigate this claim (the author never verifies it), both options have been tested; F is fixed or F is randomly selected from $[-ub, ub]$, with ub some upper boundary. For both cases, F was tested by taking $ub = [0.1, 0.2, 0.3, \ldots, 1.5]$. The other parameters were taken simply from $n = [1, 3, 10]$ and $C_r = [0.1, 0.2, \ldots, 1.0]$. Doing so automatically includes both neoteric and transversal DE ($n = 1$ is essentially neoteric DE). The results are shown in Figure 7.16.

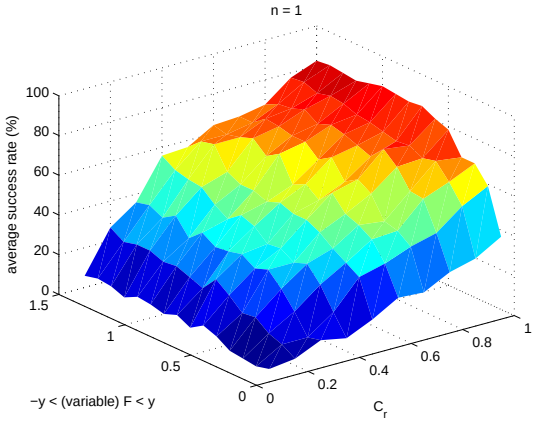
It can readily be concluded that DE performs better overall than the GA. The best performance found for the GA ($\sim$78%) is primarily due to a very high value for p_{mutate}, so that this result comes from a GA with a strong Monte Carlo "flavor". DE's performance is quite consistent for all experiments; high values for C_r and values close to 1 for F appear to give the best results, independent from whether F is constant or randomized, and whether the DE is neoteric or not. As can be concluded from Figures 7.16, DE has the most robust performance for $C_r = 1$ and $-1 \leq F \leq +1$, which validates the claim from [Feoktistov, 2006]. However, the performance appears to be independent from the value of n; it doesn't seem to matter if the DE algorithm is used in neoteric or transversal mode. Considering the fact that a transversal DE adds a layer of complexity to a neoteric DE which is apparently useless, a simple neoteric DE will be used in all further optimizations; with the aforementioned values for the control parameters, of course.

Simulated Annealing

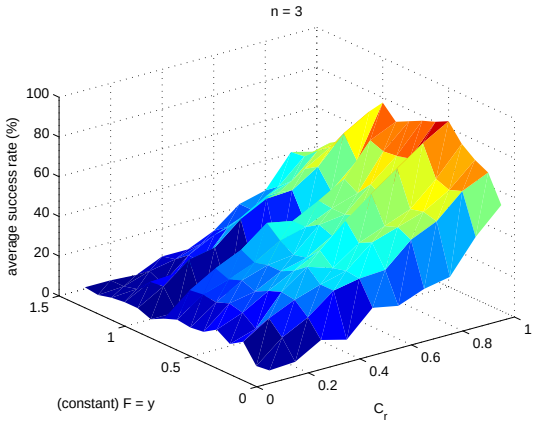
The SA algorithm has only two tunable control parameters; the initial temperature T_0 and the cooling-schedule constant c. Values for T_0 were taken from the range $T_0 = [1, 5, 10, \ldots, 100]$, and the values for c were taken from $c = [0.80, 0.81, \ldots, 0.99]$. The results of its test are



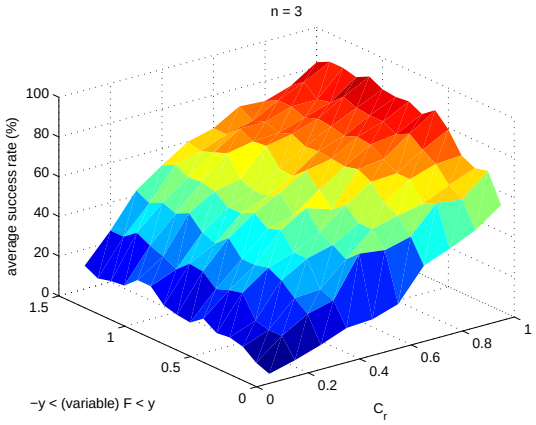
(a) $n = 1$ (Neoteric DE), constant F



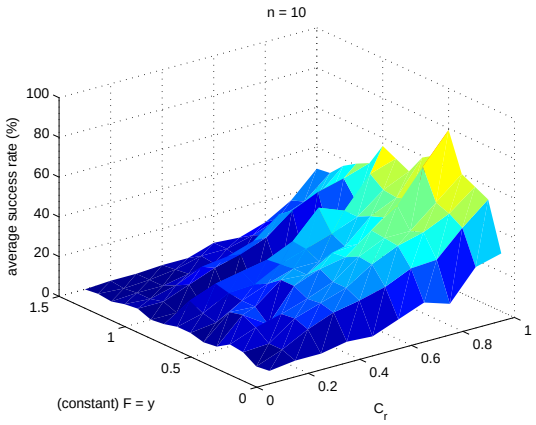
(b) $n = 1$ (Neoteric DE), variable F



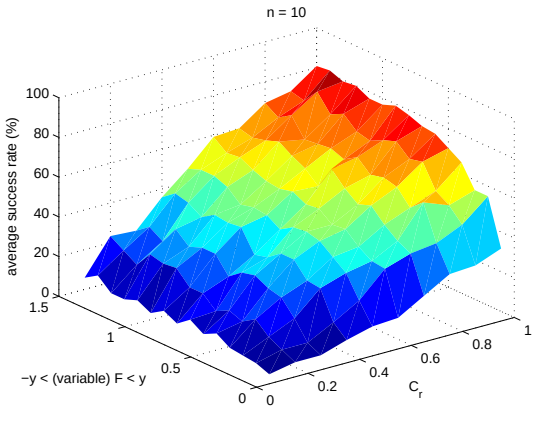
(c) $n = 3$ (Transversal DE), constant F



(d) $n = 3$ (Transversal DE), variable F



(e) $n = 10$ (Transversal DE), constant F



(f) $n = 10$ (Transversal DE), variable F

Figure 7.16 Tuning DE's control parameters. All points forming the surfaces are the averaged success rate of 20 individual runs on all 6 test functions, using the indicated values of the control parameters.

shown in Figure 7.17.

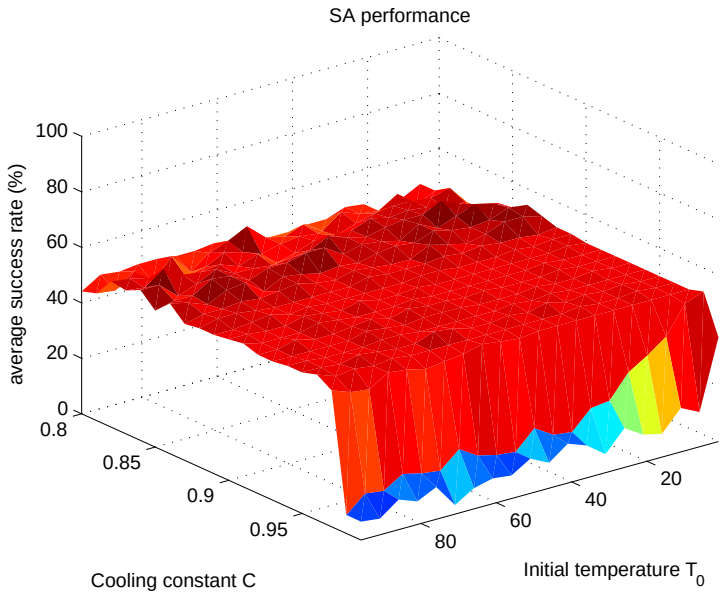


Figure 7.17 Tuning SA's control parameters. All points forming the surfaces are the averaged success rate of 20 individual runs on all 6 test functions, using the indicated values of the control parameters.

It can readily be concluded that the overall performance of SA is rather poor. Some further experimentation showed that similar to the GA, the first three test functions pose no problem for SA, but the latter three (the test-tube holder, sine-envelope-sine and the 6^{th} Bukin functions) could almost never be optimized with SA. Note also that neither of the control parameters has much influence on its performance; it is relatively insensitive to choices of the initial temperature T_0 or the cooling constant c. There is a maximum performance on $c \approx 0.87$, although it hardly shows up in the figure. There is no optimal value for the initial temperature; in this test, SA appears insensitive to different values for T_0. To test the GODLIKE algorithm, a value of 100 will be used.

Particle Swarm Optimization

Since the PSO algorithm has no less than 5 control parameters[1], finding that specific combination of values that optimizes the algorithm's performance necessitates making some compromises. It is known that the parameters that most strongly influence the overall behavior of the swarm are ω and c_l.

The inertial constant ω has a large influence on the spread of the swarm. Since low values will quickly steer the swarm to one of the other attractors it will result in very fast convergence. High values on the other hand will maintain more of the original velocity at each iteration, and therefore a better spread of the solutions and thus a slower convergence rate. The cognitive learning factor c_l is a measure of the particle's ability to learn from its

[1] Actually, 6 if the network topology is included.

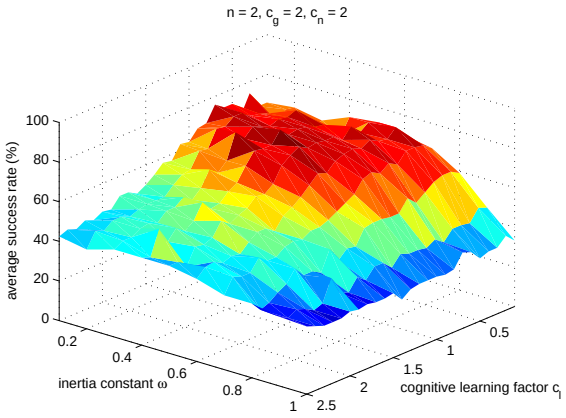
own mistakes and successes which, for low values, will result in a higher degree of social dependence (only the swarm as a whole is "smart") and usually a slower rate of convergence. A high value also makes the individual particles "smarter" and more independent from its neighbors, usually resulting in better chances of finding the global optimum.

Therefore, these two parameters will be tuned, while only varying the other control parameters (c_g, c_n and n) roughly to see what their influence is. The parameter c_l will be taken from $c_l = [0, 0.1, \ldots, 2.5]$, the parameter ω will be taken from the range $\omega = [0.0, 0.1, \ldots, 1]$, and the others will simply be varied according to $n = [2, 10]$, $c_n = [1, 2, 3]$ and $c_g = [1, 2, 3]$. Also, to better visualize the influence of c_g on c_n, these parameters will first be taken equal ($c_n = c_g = 2$) (referred to as *egalitarianism*), and then unequal ($c_n = 1$, $c_g = 3$ and $c_n = 3$, $c_g = 1$). The first of these inequalities emphasizes the "common goal" of the swarm more, and will therefore be referred to as *communism*. The other one emphasizes a particle's social environment more strongly, and will therefore be referred to as *socialism*. For all tests, the particle's social network topology was star-shaped. The results of this test are shown in Figure 7.18.

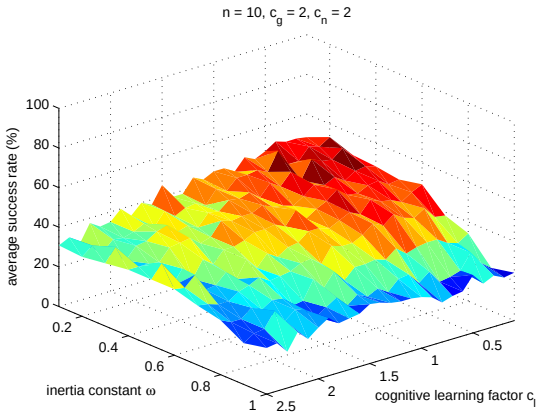
From these figures it can be concluded that a communist population ($c_g = 3$, $c_n = 1$) in which all individuals are easily persuaded to change their direction (ω small), are strongly interconnected (n large) but fairly egoistic (c_l small) works best; Figure 7.18(f) has the largest "dark red" region, showing that its performance is the most robust. To test the GODLIKE algorithm, the values $c_g = 3$, $c_n = 1$, $\omega = 0.4$, $c_l = 0.7$ and $n = 10$ will be used. Note that the surprisingly disappointing performance for the socialists (Figure 7.18(c)) is not a numerical artifact of some sort or due to erroneous implementation; all tests have been done several times for various different cases, only to find that this is a real phenomenon – the cause of this remains unknown.

GODLIKE

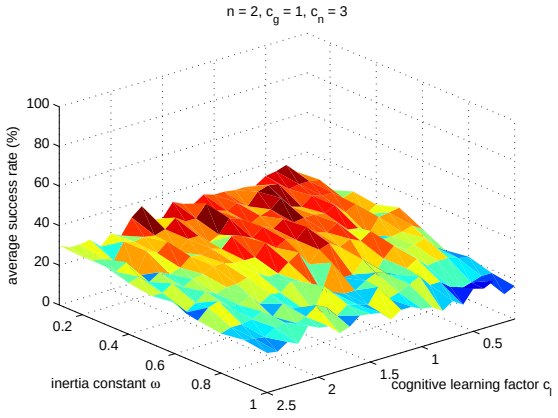
When using all of the algorithms discussed in this chapter, GODLIKE has four tunable control parameters: the number of optimization streams N, the minimum and maximum amount of iterations $ItersLb$ and $ItersUb$ respectively, and the re-heating constant C_{reheat} used to re-initialize the SA-algorithm. It can be expected that the influence of C_{reheat} on the whole algorithm is minimal, as most time is actually spent in one of the other optimizers per iteration in GODLIKE. Therefore, only two different values will be tested to see its effect: $C_{\text{reheat}} = [1, 10]$. Note that the first of these values implies re-heating to the full initial temperature, while the second value implies only a small amount of re-heating. It can be expected that the number of individual optimization streams has more influence on the performance, since a higher value of N implies more interaction between the streams. Two different settings will be tested, $N = 1$ (no *interchange*) and $N = 5$ (considerable amounts of *interchange* operations per iteration). The remaining parameters, $ItersLb$ and $ItersUb$, will be chosen from the intervals $ItersLb = [5, 10, \ldots, 100]$ and $ItersUb = [20, 25, \ldots, 400]$. Note that a prerequisite of GODLIKE is $ItersUb \geq 4 \cdot ItersLb$, since $ItersUb$ defines the *total* amount of iterations spent in the four different optimizers per iteration in GODLIKE. For many combinations of values in these intervals this condition can not be met, hence these combinations are simply skipped. The intervals should however be maintained, since they



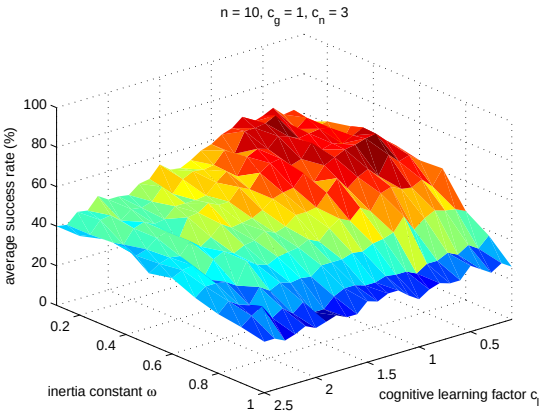
(a) $n = 2$, egalitarian swarm



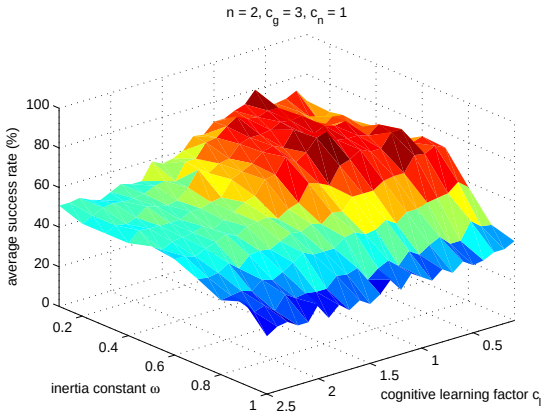
(b) $n = 10$, egalitarian swarm



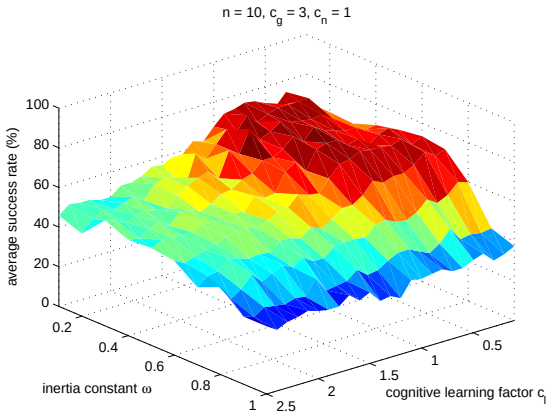
(c) $n = 2$, socialist swarm



(d) $n = 10$, socialist swarm



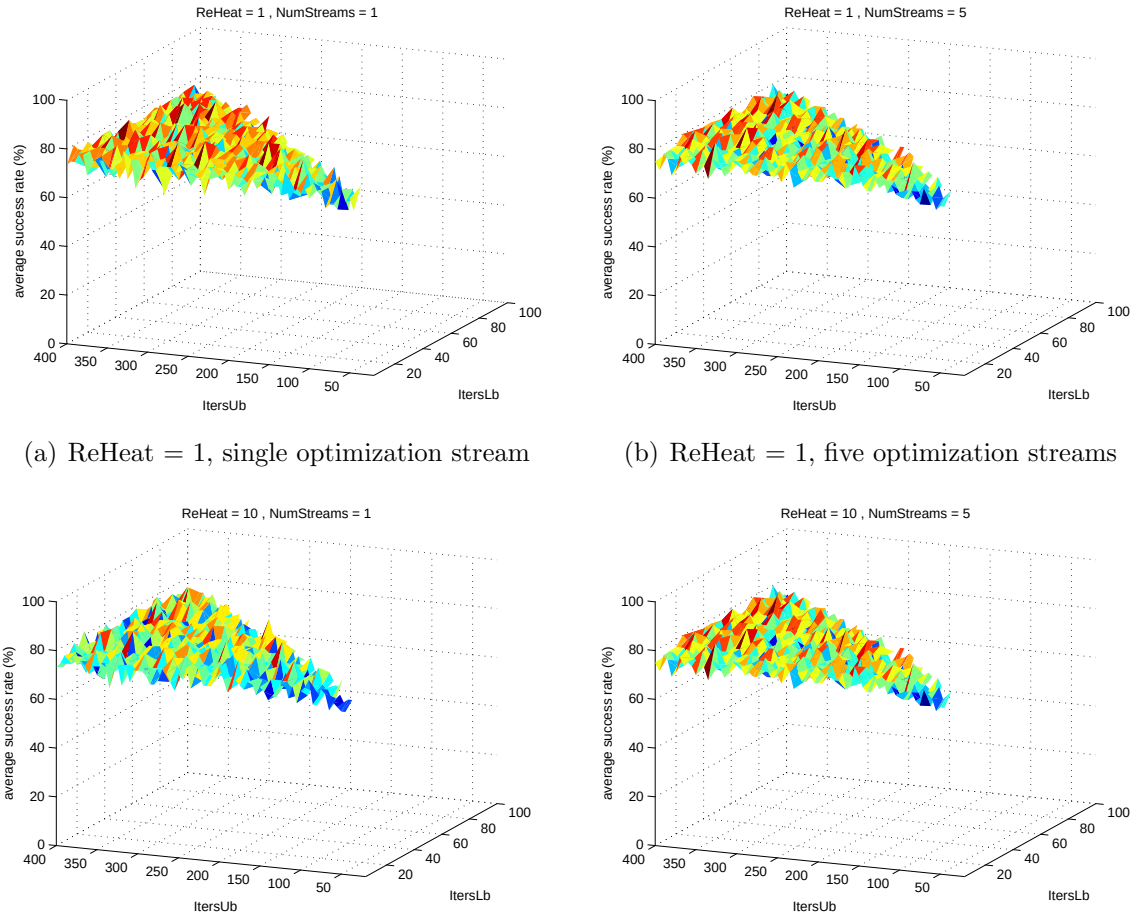
(e) $n = 2$, communist swarm



(f) $n = 10$, communist swarm

Figure 7.18 Tuning PSO's control parameters. The analogies egalitarianism, socialism and communism refer to the values of the social and global control parameters; $c_n = c_g = 2$ = egalitarian, $c_g = 3$, $c_n = 1$ = communist, $c_g = 1$, $c_n = 3$ = socialist. All points forming the surfaces are the averaged success rate of 20 individual runs on all 6 test functions, using the indicated values of the control parameters.

will also illustrate GODLIKE's behavior at very low settings for both, or very high settings for both $ItersLb$ and $ItersUb$.



(a) ReHeat = 1, single optimization stream

(b) ReHeat = 1, five optimization streams

(c) ReHeat = 10, single optimization stream

(d) ReHeat = 10, five optimization streams

Figure 7.19 Tuning GODLIKE's control parameters. All points forming the surfaces are the averaged success rate of 10 individual runs on all 6 test functions, using the indicated values of the control parameters.

The results of this test are shown in Figures 7.19(a) through 7.19(d). All values of the control parameters for each individual optimizer that were found to give it its best performance were also applied for this test. As can be concluded from these Figures, GODLIKE's performance is quite good for *any* value of its control parameters. This is indeed partly the intention; do away with the need to hand-tweak the global optimizer to a specific problem. There does seem to be a slight preponderance for higher values of $ItersUb$, so by default a value of 250 will be used. Judging from Figures 7.19(a) through 7.19(d) this value is high enough to yield good results while it is low enough to enable the algorithm to perform a few *interchange* operations when allowing a low number of FE in combination with large population sizes. Finally, as expected, the influence of C_{reheat} is hardly noticeable, so a default value of $C_{\text{reheat}} = 5$ will be used for all optimizations.

Unfortunately this test does not suffice to show the influence of the *interchange* operator on GODLIKE's performance; for almost all test functions (save for the 6^{th} Bukin function), the optimum was found before any significant amount of *interchange* operations could be carried out. The influence of this operation will have to be found via another test, which will be left as a recommendation.

8

Multi-Objective Optimization

All methods discussed so far have assumed that any trial solution has a one-to-one mapping with the value of some objective function. This is generally referred to as *single-objective* optimization, since there is only one objective to be optimized, e.g., ΔV, profit, minimum error, etc.

In most practical problems however, it is desirable that multiple objectives are optimized simultaneously. For instance, when setting up the production line for some new product, it is not desirable that the initial investment is very high. On the other hand, a larger initial investment generally means more production capacity and thus better commercial viability of the product, leading to more profit in the end. The production line is optimal when some optimum trade-off is found between a minimum initial investment and a maximum profit. These sorts of problems are referred to as *multiple-objective* or MOO problems. MOO problems are relevant to the current mission, because it inherently has multiple objectives to optimize. For mission scenarios 1 and 2, the time of flight needs to be minimized while maximizing the spacecraft's dry mass. For mission scenarios 3 and 4, the situation is even more complex – the amount and/or quality of all minor planets visited in the meantime must be maximized, while optimizing both previous objectives as well.

As can be expected, multiple objectives usually have a strong influence on each other. In most situations, one objective can not be improved without making another objective worse, giving rise to a whole new level of complexity in the optimization, which is the subject of this chapter.

8.1. Optimality with Multiple Objectives

A MOO-problem with N objectives can be formulated as

$$\begin{aligned} \min F(\boldsymbol{x}) &= \begin{bmatrix} f_1(\boldsymbol{x}) & f_2(\boldsymbol{x}) & \dots & f_N(\boldsymbol{x}) \end{bmatrix}^T && (8.1) \\ &\text{s.t.} \\ G(\boldsymbol{x}) &\leq 0 \\ H(\boldsymbol{x}) &= 0 && (8.2) \\ \boldsymbol{a} \leq \boldsymbol{x} &\leq \boldsymbol{b} \end{aligned}$$

It is far from trivial to formulate exactly when $F(\boldsymbol{x})$ has an optimum. The usual KKT-conditions do not apply, since the optima of either of the functions $f_i(\boldsymbol{x})$, $i = 1, \dots, N$ generally have nothing to do with that of all other functions.

8.1.1 Aggregate Objective Function

The most intuitive solution to the problem 8.1–8.2 would be to use one of the aforementioned methods (local optimizer, or global optimizer with penalty functions imposed on the constraints) on the Aggregate Objective Function (AOF)

$$\hat{F}(\boldsymbol{x}) = \lambda_1 f_1(\boldsymbol{x}) + \lambda_2 f_2(\boldsymbol{x}) + \ldots + \lambda_N f_N(\boldsymbol{x}), \tag{8.3}$$

subject to the same constraints as before (or with penalty-terms added). Here, $\Lambda = (\lambda_1, \lambda_2, \dots, \lambda_N)$ with $\lambda_i \geq 0$, $i = 1, \dots, N$ and $\sum_{i=1}^{N} \lambda_i = 1$ is a vector of some pre-defined weights.

The problem with this method is that its final solution strongly depends on the chosen weights Λ. There is no set of weights that results in a completely unbiased optimization. As an extreme example: in a problem with two objectives, if $\Lambda = (1, 0)$ is chosen, only the first objective is optimized while neglecting the second, while $\Lambda = (0, 1)$ will accomplish the opposite.

8.1.2 Pareto Optimality

As there is no obvious completely unbiased set of weights Λ, any choice of λ will result in a different optimal solution. It can be shown that when *all possible* choices of λ are optimized individually, the resulting optima will lie on the same hypersurface in the associated *bi-loss map*[1]. As an example, for the two-objective problem

$$F(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n-1} \left(-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2} \right) \right) \\ \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5 \sin x_i^3 \right) \end{bmatrix} \tag{8.4}$$

the hypersurface takes the shape of a "wavy line", which is plotted in Figure 8.1. This hypersurface is generally referred to as the *Pareto front*, after the Italian sociologist, economist and philosopher Vilfredo Pareto (1848-1923).

Solutions belonging to the Pareto front can not be reduced further in one objective, without negatively affecting another objective. Therefore, solutions that lie on the Pareto front are

[1]The bi-loss map is a classical name for a plot where all the individual function values for a particular solution are plotted against each other. Figure 8.1 is an example of this.

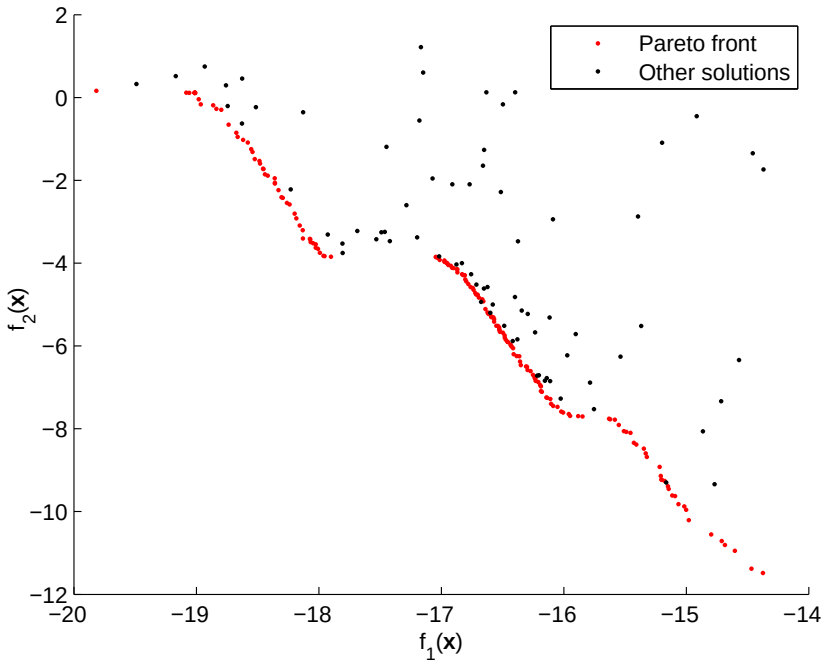


Figure 8.1 The Pareto front for a simple two-objective optimization problem. Those points above the "solid" line indicate other, less optimal, solutions.

optimal in *all* objectives *simultaneously*. Therefore, these solutions are said to be *Pareto efficient* or *Pareto optimal* solutions.

Thus, the result of an MOO problem is not a single best individual solution, but a whole spectrum of possible choices that together optimize the problem. Although only *one* of these solutions will eventually be used, the Pareto front is actually the preferred solution since it greatly facilitates a decision maker's job. Any single solution does not allow much flexibility, nor does it provide much insight into the behavior of the objectives around it. For example, when a project's initial budgetary constraints are suddenly tightened half-way through completion (which happens quite often in large scale projects such as the intended mission), the optimization does not have to be performed all over again – a different Pareto solution that can meet the new demands is simply selected[2].

8.2. Finding the Pareto Front

Naturally, it is even more challenging to find the complete Pareto front of globally optimal solutions than a single optimal solution. A large variety of methods has been studied in the past, two of which generally stand out from the rest – *Normal Boundary Intersection* and *Non-dominated Sorting Genetic Algorithms*.

[2]Naturally, with the Pareto solutions, a much more elaborate financial model can be constructed already at the start of the project.

8.2.1 Normal Boundary Intersection

Probably the most intuitive way to optimize an MOO problem is to optimize the AOF M times, for M differently chosen weights Λ. However, there are several drawbacks to this approach. As shown in [I. Das, 2005], the M solutions will *only* converge to the Pareto front if that front is convex. Although for many problems this is indeed the case, it certainly does not hold for all functions encountered in practice. Also, even if the weights Λ are chosen from an equally distributed sample space, the resulting solutions will not be equally distributed over the Pareto front; there will be regions where the solutions tend to cluster together, and other, much harder to locate regions, will be virtually devoid of solutions.

The Normal Boundary Intersection (NBI) method, first described by Das and Dennis [1998], aims to prevent these problems. The technique is based on the observation that points on the Pareto front (the curve ACB in the (oversimplified) bi-loss map shown in Figure 8.2) always lie in between the points A and B, and always lie on the normal to the line AB towards the origin.

The complete NBI method can be stated as follows. Let the *utopia point* or *shadow minimum* be given by

$$F^* = \begin{bmatrix} f_1^* & f_2^* & \dots & f_N^* \end{bmatrix}^T,$$

where f_i^* are the minima of the individual objective functions. In Figure 8.2, this would be the point

$$F^* = \begin{bmatrix} A & B \end{bmatrix}^T.$$

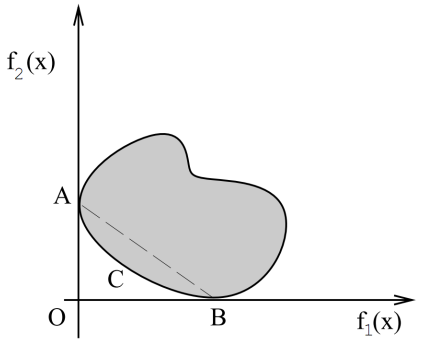


Figure 8.2 A typical bi-loss map with a simple convex Pareto front (curve ACB). This is Figure 1 from [Das and Dennis, 1998].

Redefine the function $F(\boldsymbol{x})$ to have the utopia point at its origin, e.g.,

$$\hat{F}(\boldsymbol{x}) \leftarrow F(\boldsymbol{x}) - F^*.$$

Then, using the central idea outlined above, the optimization problem can be subdivided into M NLP-subproblems of the form:

$$\max_{\boldsymbol{x},t} \quad t \tag{8.5}$$

$$\begin{aligned} \text{s.t. } \Phi\Lambda_n + t\hat{n} &= \hat{F}(\boldsymbol{x}) \\ H(\boldsymbol{x}) &= 0 \\ G(\boldsymbol{x}) &\leq 0 \\ \boldsymbol{a} \leq \ \boldsymbol{x} &\leq \boldsymbol{b}, \end{aligned} \tag{8.6}$$

where the constraint functions $G(\boldsymbol{x})$ and $H(\boldsymbol{x})$ and the lower and upper boundaries $\boldsymbol{a}$ and $\boldsymbol{b}$ are defined as in Equation 8.2, $\hat{n}$ is the normal vector to the surface defined by the individual minima, Λ_n is the n^{th} set of weights from the total of M chosen weight vectors, and the i^{th} column of the matrix Φ is defined as

$$\Phi(:,\, i) = F(\boldsymbol{x}_i^*) - F^*,$$

in which $\boldsymbol{x}_i^*$ is the location of the minimizer for the i^{th} component function f_i.

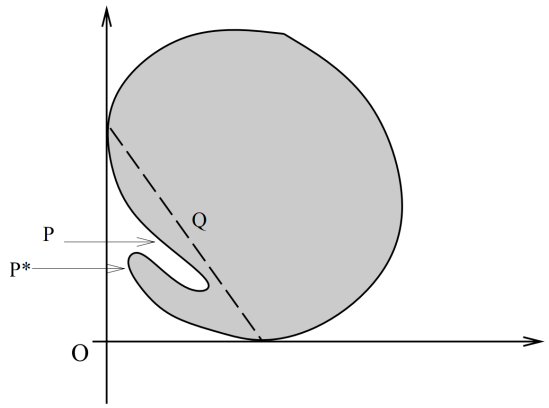


Figure 8.3 A complicated and hard to find Pareto Front. Although NBI will not find the solution P, it *will* find the globally optimal Pareto point P^* (provided a proper starting point is used). This is Figure 4 from [Das and Dennis, 1998].

The formulation of the subproblem above ensures the solution found will be inside the domain of $F(\boldsymbol{x})$, as follows from the first constraint. Moreover, since the *maximum* value of the parameter t is aimed for, it will ensure that the particular solution indeed lies on the *global* Pareto front (farthest from the line AB in the example), even if that front has a very complex and not globally convex shape (see Figure 8.3).

Solving M of these NLP-subproblems will construct the complete global Pareto front. The homogeneity of the distribution of the solutions found only depends on the (preset) value of M – the higher the value of M, the more points on the global Pareto front will be found. Moreover, when implemented carefully, the NBI method does *not* require one of the global optimizers discussed in chapter 7, but could use one of the local optimizers discussed in chapter 6. These methods, as discussed previously, generally require far less FE's to converge (provided analytical derivatives can be provided), and are thus potentially much faster.

One obvious drawback to the NBI method is that the utopia point F^* should be known beforehand, which would require N optimizations. But the particular way the matrix Φ is defined can remedy this drawback. Obviously,

$$\begin{aligned} \Phi(i,\, i) &= 0, \text{ and} \\ \Phi(j,\, i) &\geq 0,\, j \neq i. \end{aligned}$$

Using these relations, a very simple way to check the correctness of the initial estimate for F^* can be implemented. If a negative value appears in Φ at some iteration j, the current test point $\boldsymbol{x}_j$ apparently *improves* the i^{th} column of Φ, indicating the original F^* was incorrect. The matrix Φ (and F^*) can then be simply be updated. Therefore, this particular way to define Φ allows one to use only a very rough approximation to F^*.

Another drawback is that NBI in the above form depends on the particular scaling of the $f_i(\boldsymbol{x})$. That is, if all the objectives $f_i(\boldsymbol{x})$ are multiplied by some arbitrary factor, the points on the Pareto front found with the NBI method will be different from those found for the problem without these factors. However, the Pareto front found can be made independent of scaling altogether, by redefining the normal vector $\hat{n}$ to a *quasi-normal* vector $\tilde{n}$. This vector is not exactly normal to the convex hull of the points $f_i(\boldsymbol{x})$, but has at least *some* negative

components. It can be shown that if the quasi-normal vector is chosen as an equally weighted combination of the columns of the matrix Φ, e.g.,

$$\tilde{n}_n = -\Phi e_n,$$

with e_n a standard basis vector in dimension n, the NBI method becomes completely *independent* of the scaling of the functions $f_i(\boldsymbol{x})$.

In summary, NBI is a powerful method to find the Pareto front for a particular problem. The amount of solutions returned can be controlled, and problems that could previously only be solved with global optimizers (which require a large amount of FE's) can now be solved with (more efficient) local optimizers. There are some practical drawbacks, mostly regarding a generally efficient implementation. Although many of the aspects regarding this issue have been studied (and remedied) by Lim et al. [2001], correct and efficient implementation of the NBI method for the problems encountered in the current mission remains a daunting task.

8.3. Non-dominated Sorting Genetic Algorithm

The most popular method for MOO problems by far, is the Non-dominated Sorting Genetic Algorithm (NSGA). NSGA was introduced by Srinivas and Deb [1994], which was later improved to the slightly more efficient Non-dominated Sorting Genetic Algorithm, second version (NSGA-II) by Deb et al. [2000]. Its popularity can be explained by the fact that NSGA simply *extends* the already popular GA used for single objective optimization with an additional set of operators. These operators enable the GA to directly find the complete Pareto front in a *single* run, unlike many other MOO-optimizers.

The principle behind NSGA-II is surprisingly simple. In a single-objective GA, the final solution returned is the individual trial solution that was found to have the best value of its fitness (lowest function value). This particular solution is found by applying the principles of evolution to some initial, randomly distributed, population of trial solutions. The better the function value (or "fitness") of a particular trial solution, the higher the probability that some elements from that trial solution will appear in the next generation.

In MOO, a solution $\boldsymbol{x}_k$ is said to *dominate* another solution $\boldsymbol{x}_m$,

$$\boldsymbol{x}_k \prec \boldsymbol{x}_m$$

if the following conditions apply:

$$\begin{array}{rcll} f_i(\boldsymbol{x}_k) & \leq & f_i(\boldsymbol{x}_m) & \text{for all } 1 \leq i \leq N, \text{and} \\ f_i(\boldsymbol{x}_k) & < & f_i(\boldsymbol{x}_m) & \text{for at least one } i \in \{1,\, N\}\,. \end{array} \tag{8.7}$$

In NSGA-II, the fitness of individual trial solutions is no longer the function value alone, but the *ranking* of the solution – the amount of *other* individuals that dominate it. Also, the new population is not only generated by the principles of evolution, but also by a tournament selection based on both the ranking and the *crowding distance* of the trial solution – trial

solutions that appear in the next generation will be both low in rank, *and* have little other trial solutions "close" to it.

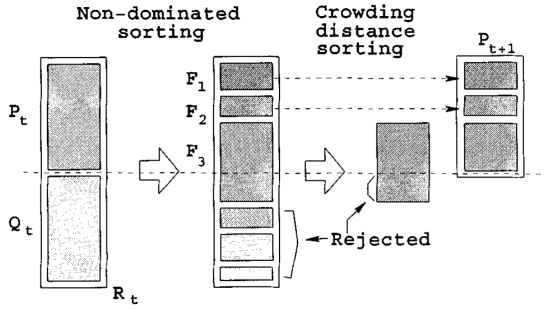


Figure 8.4 The principle of using non-dominated sorting and the crowding distance to create a new generation in the NSGA-II method. This is Figure 2 from [Deb et al., 2000].

One complete iteration of NSGA-II works as follows[3] (see also Figure 8.4): consider a population P_t of Z individuals. The crowding distance operator, which is best defined in algorithmic form:

$$
\begin{array}{ll}
\mathbf{I}[i] = 0,\ i = 1 \ldots \ell & \text{initialize for } \ell \text{ individuals} \\
\text{for each objective } m & \\
\quad \mathbf{I} = \text{sort}(\mathbf{I}, m) & \text{sort using each objective} \\
\quad \mathbf{I}[1] = \mathbf{I}[\ell] = \infty & \text{boundary points are always selected} \\
\quad \text{for } i = 2 \text{ to } \ell - 1 & \text{for all other points} \\
\qquad \mathbf{I}[i] = \mathbf{I}[i] + \frac{\mathbf{I}[i+1] - \mathbf{I}[i-1]}{f_m^{\max} - f_m^{\min}} & \\
\quad \text{end} & \\
\text{end} &
\end{array}
\tag{8.8}
$$

is used for a binary[4] tournament selection that selects suitable parents to use for crossover. This binary tournament selection works as follows: two random individuals $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are selected from population P_t, and the one that is "better" than the other in the following sense

$$
\begin{array}{lll}
\boldsymbol{x}_i & \prec_n & \boldsymbol{x}_j \\
& \text{if} & (\boldsymbol{x}_i^{\text{rank}} < \boldsymbol{x}_j^{\text{rank}}) \\
& \text{or} & ((\boldsymbol{x}_i^{\text{rank}} = \boldsymbol{x}_j^{\text{rank}}) \text{ and } (\boldsymbol{x}_i^{\text{distance}} > \boldsymbol{x}_j^{\text{distance}}))
\end{array}
\tag{8.9}
$$

is selected as a suitable parent and inserted into the mating pool of size R. This selection procedure is repeated until this mating pool is filled. It is common practice to have the size of the mating pool equal to half the population size, e.g., $R = N/2$. The parents from the mating pool are then used by a common GA to generate the offspring population, Q_t.

[3]This is an iteration *after* the first iteration. The first iteration is better regarded as an initialization of the NSGA-II algorithm, and thus very different from all following iterations – it will not be discussed any further here.

[4]The term "binary" in this context does not refer to the binary number system, but to the fact that 2 individuals are selected to compete.

The population Q_t is appended to the parent population P_t, to apply all operations to follow to *both* populations simultaneously – call this population $P_t \cup Q_t$ the *augmented* population – the size of the augmented population is thus $2Z$. For each individual in the augmented population $\boldsymbol{x}_j$, the values of the N objective functions $f_i(\boldsymbol{x}_j)$, $i = 1, \ldots, N$ are calculated. Next, the function values of *every* individual are compared with those of *every* other individual, and Equation 8.7 is applied to see if one dominates the other. The purpose of this procedure is to count the number of individuals that dominate every other individual, and the output of this procedure is indeed a vector containing these numbers. The values contained in this vector are equal to the individual's *ranking*, or *front number* F. For some individuals, the ranking will be equal to zero, meaning that *no* individuals dominate it – that individual is thus part of the Pareto front of the current generation. If an individual has a ranking of one, it means that it is part of the current generation's Pareto front, if the individuals with $F = 0$ had not been present in the population. This reasoning holds for every value of $F > 0$, thus an individual's F-value is indicative of its quality in the current population.

A new population P_{t+1} is then constructed, by inserting the individuals with the lowest F number into P_{t+1}. Since the new population P_{t+1} should contain only Z individuals, the individuals from the last front F that partially fits in P_{t+1} are sorted according to their crowding distance. The individuals from this last front that have the largest crowding distance, are then selected as the remaining individuals to fill the population P_{t+1} (see the rightmost part in Figure 8.4). Because both the parent and offspring populations are used and re-sorted in every iteration, NSGA-II automatically preserves the best individuals found thus far – it inherently employs the principle of elitism, making the NSGA-II inherently very robust. Also, in contrast to a single-objective GA, convergence in the NSGA-II algorithm is much more well-defined. The whole population converges towards the Pareto front, meaning that if *all* members are non-dominated, it can safely be assumed that the true Pareto front has been found and the algorithm has converged.

This amounts to the strongest advantage of the NSGA-II algorithm: at least Z equally distributed points will be found on the Pareto-front, and these points are found in only *one* run of the algorithm. This makes it quite versatile and very easy to use. Also, compared with the NBI method, the implementation is lengthy but quite straightforward, and usually does not pose any major difficulties. The strongest drawback of NSGA-II is the computational cost for the non-dominated sorting operator: since $2Z$ individuals need to be compared against $2Z$ other individuals M times, the complexity of this part of the algorithm is a staggering $O(M(2Z)^2)$. As shown in [Deb et al., 2000], careful bookkeeping and efficient use of memory can reduce this to $O(MZ^2)$, but still it remains the dominant part of the complexity of the algorithm.

8.4. Trade-Off

NBI and NSGA-II were both used in a comparative study by Sendin et al. [2006], who applied the methods to a real-world problem in the design of a biochemical reactor. To solve the individual NLP-problems globally for NBI, three methods were applied: SQP, SRES and

GLOBAL (two other global optimizers, see [Sendin et al., 2006, and references therein]). Their findings indicate that even in the relatively simple test problem considered, the NLP-problems resulting from the NBI method can be quite difficult to solve with SQP or SRES. The completeness and spread of points on the resulting Pareto front seemed to depend quite strongly on the initial individual optimum the algorithm was started with. On the other hand, the other method used (GLOBAL) *was* able to solve the problems satisfactorily and without much difficulty. The NSGA-II method produced the complete Pareto front without any difficulty, and points were spread more or less homogeneously across it. Several optimality measures also indicated the fronts found by both methods were about equal in quality.

However, although the NBI method tested required more FE's than the NSGA-II method (the associated derivatives were computed numerically), the NSGA-II method required much more computation time for similar run-time conditions. This is indeed indicative of the relatively large computational effort required by NSGA-II: the overall complexity is $O\left(M(N^2 + \log N)\right)$, which grows linearly with the number of objectives but *super-quadratically* with the population size. Such complications are non-existent with the NBI method – its complexity depends mainly on the efficiency of the global or local optimizer used, which usually has a much lower complexity.

The optimizations required for the current mission are generally much more difficult than the biochemical reactor example used by Sendin et al. [2006]. This is partly because the individual objective functions are locally non-convex, might not be continuous and, in case the amount of minor planets visited itself is an objective function (which is the case for mission scenario 4), is integer-valued. But it is primarily due to the fact that the required search spaces are generally quite large[5], which necessitates relatively large population sizes. Considering the computational complexity required for NSGA-II, its quite likely that the internal operations of the NSGA-II algorithm itself will greatly exceed the time required for the actual FE's.

However, considering the difficulty of a correct and efficient implementation of NBI (see [Lim et al., 2001]) *and* the difficulties encountered by Sendin et al. [2006] to achieve a homogeneous spread of solutions along the Pareto front, *plus* the difficulties to achieve convergence of the associated NLP-problems, using and further testing of the NBI method will be left here as a recommendation. Instead, the relatively simple NSGA-II algorithm will be employed and the additional computation time required will be taken for granted.

8.5. Using NSGA-II on Constrained Functions

It is not effective to use penalty-function methods on constrained multi-objective problems. This is because solutions that violate the constraints severely can still be non-dominated, and thus be returned as a member of the final Pareto front by NSGA-II. Also, re-initializing population members when they violate the constraints will indeed work, but is very inefficient when the constraints become more demanding. The algorithm will *loose* all information

[5] As will be described in section 11, for almost every case, no well-founded upper or lower limits can be set on the individual transfer times between the planets – this necessitates making the associated search spaces quite large.

on the non-domination history of each member that is re-initialized, which will occur more frequently with more demanding constraints. This destroys the power of the embedded elitist approach used in NSGA-II.

Instead of using one of the aforementioned methods, [Deb et al., 2000] proposed to use an additional quality measure in both the non-dominated sorting and the tournament selection routines. The quality measure q they used was simply the sum of all constraint violations:

$$q(\boldsymbol{x}) = \sum_{k=1}^{N} |H_k(\boldsymbol{x})| + \sum_{j=1}^{M} \begin{cases} G_j(\boldsymbol{x}) & \text{if } G_j(\boldsymbol{x}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8.10}$$

with $G(\boldsymbol{x})$ and $H(\boldsymbol{x})$ as in Equation 8.2. The quality measure q is then used in much the same manner as the crowding distance d in both the tournament selection and non-dominated sorting routines, to force those solutions that are inside the feasible domain to be preferable over those that are not. Thus, the measure q advances the Pareto front primarily with solutions that are "more feasible" than others.

Although this method is rather simplistic, it was shown to the surprise of the authors to be superior to a much more advanced method. Since it is rather easy to include the quality measure q in the complete NSGA-II algorithm (and skip it if no constraints are present), it will certainly be used during the optimization of the MGA problems associated with the current mission.

8.6. Non-dominated Sorting in Conjunction with DE, PSO & ASA

The NSGA-II method uses non-dominated sorting and a tournament selection based on the crowding-distance between the individuals, in conjunction with a (largely unmodified) GA, which is used only to generate the new population. It should be obvious that the non-dominated sorting algorithm is in essence *completely separate* from the GA. In other words, the non-dominated sorting and calculation of the crowding distance can be done *separately*, while generating the new population can be carried out by *any* optimizer; as well as one (or all) of the other global optimizers discussed in chapter 7.

To put this idea to the test, all global optimizers discussed in chapter 7 were modified to incorporate the necessary changes. It should be obvious that GA and DE pose no problem, as GA was the original algorithm used, and DE can generate a new population based solely on the previous one. The PSO and SA algorithms are significantly more difficult to reshape into a multi-objective form, as they both require globally changing variables (*lbest*, etc. for PSO, and T for SA), and both require more information about the function values of the offspring to decide whether to accept them or not. The trouble however does not lie with the SA algorithm; only the temperature has to be decreased at every iteration, and the acceptance criterion is simply replaced by the one carried out by the non-dominated sorting algorithm described above. The real difficulty lies with translating the PSO algorithm into a

multi-objective one.

Although much research has been done on this in the past decade, it is only in the past 3 to 4 years that Multi-objective Particle Swarm Optimizers (MOPSO's) have been developed that were somewhat successful. For example, an early work by Xiaohui and Eberhart [2002] used the notion of a *dynamic neighborhood*; they assign new neighbors to the particles in each iteration. The n particles that are closest to each particle in function-value space form that particle's new social environment. The value for *nbest* is replaced, if one of the new neighbors dominates the previous *nbest*. If more than one new neighbor dominates, the one with the largest crowding distance is selected. The authors continued by optimizing each objective function separately while trying to keep the value of the other function constant. Unfortunately, they experienced many difficulties to obtain convergence even for simple problems. This is indicative of the problematic nature of using the PSO scheme for multi-objective problems.

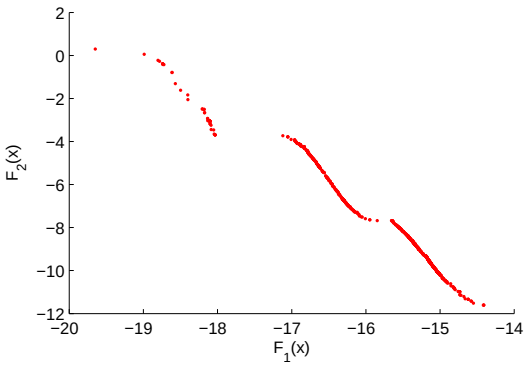
An excellent survey written more recently by Reyes-Sierra and Coello-Coello [2006] focusses on the state-of-the-art on MOPSO's. In this review, the authors also refer to a version that is based on the NSGA-II algorithm ([Reyes-Sierra and Coello-Coello, 2006, ref. 37]. This particular version uses the non-dominated sorting procedure described above, but instead of comparing the old population with the newly created one, they compare all the *local best solutions* to the newly created population. This ensures that non-dominated solutions are always retained, and that the *lbests* are only replaced when they dominate the previous *lbests*. They also use a *fully-connected* social network topology, that is, every particle has every other particle as its neighbor. The global best *gbest* and neighborhood best *nbest* are updated by those particles that are part of the current Pareto front, and have the largest crowding distance. Since this version requires the least amount of modifications to the NSGA-II algorithm, this particular version will be used and tested.

Only a simple test will be carried out to validate these different versions of the NSGA-II, and see whether it is worth to pursue further. For this simple test, the problem given in Equation 8.4 will be optimized by all 5 algorithms (GA, DE, PSO, SA and GODLIKE). This problem is described in [Deb et al., 2000, ref. 1], and is quite hard to optimize because its Pareto front is disconnected. Particularly the top-leftmost branch is challenging to find.

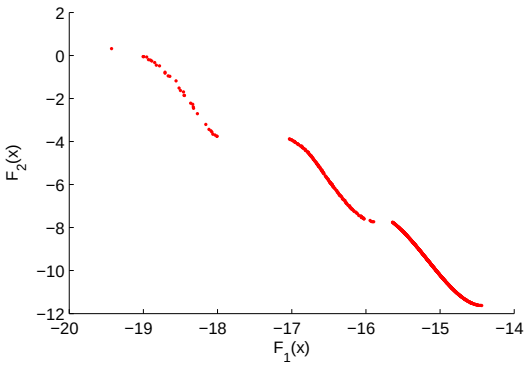
For all tests, a population size of 1000 individuals will be used to give good statistics on the final spread of the solutions along the Pareto front, and a maximum amount of FE's of 100,000 will be maintained. The results of this simple test are shown in Figure 8.5.

From these figures it can be readily concluded that the PSO-algorithm, despite its changes for MOO, is not very robust. It does seem to locate the Pareto front, but it has severe difficulty in achieving convergence. Also the SA algorithm, by itself, performs rather poorly in MOO[6]. The other algorithms however can indeed be used without difficulty. As expected, all of the algorithms experience difficulty finding the top-leftmost branch of the Pareto front.

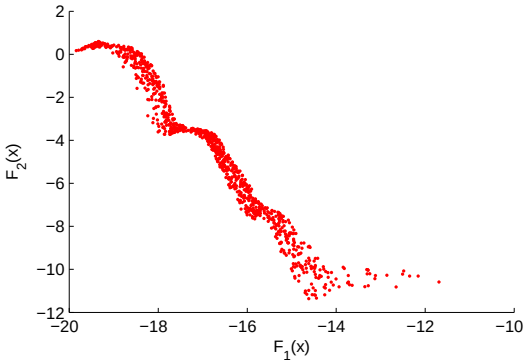
[6]Some further testing showed that simply *not* reducing the temperature gave much better results. Investigating how or why that is, will be left as a recommendation.



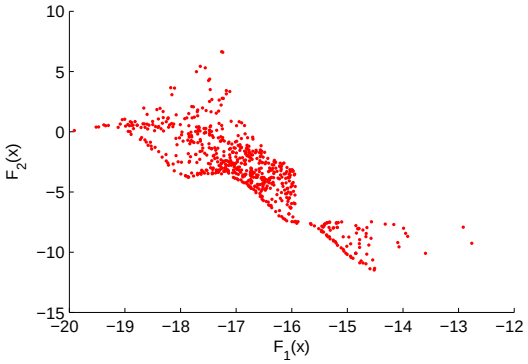
(a) Solution generated by GA. All solutions lie on the Pareto front.



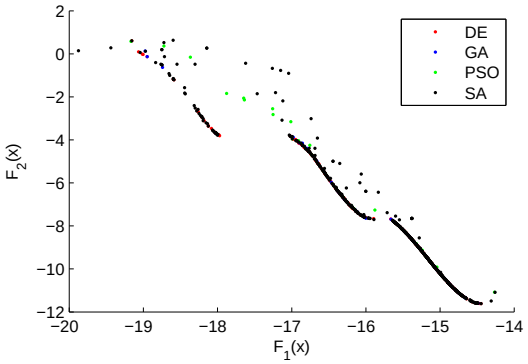
(b) Solution generated by DE. All solutions lie on the Pareto front.



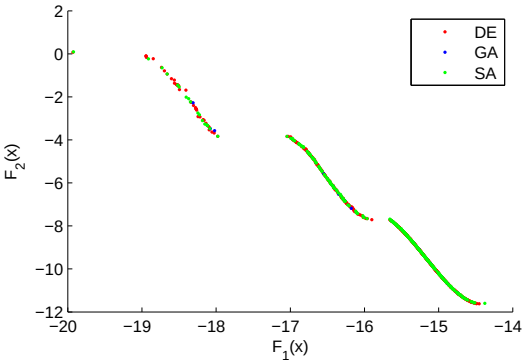
(c) Solution generated by SA. After exhausting the maximum number of function evaluations,SA proved unable to put all solutions on the Pareto front.



(d) Solution generated by PSO. After exhausting the maximum number of function evaluations,PSO proved unable to put all solutions on the Pareto front.



(e) Solution generated by GODLIKE. When PSO was *in*cluded, GODLIKE proved unable to put all solutions on the Pareto front within the given computational budget.



(f) Solution generated by GODLIKE. When PSO was *ex*cluded, GODLIKE generated the highest quality Pareto front of all algorithms tried.

Figure 8.5 Testing & validating NSGA-II in conjunction with optimizers different than GA, using the test-functions from Equation 8.4. For each optimization the population size was 1000, and the maximum number of function evaluations allowed was fixed to 100.000. Note that the absolute worst performing algorithm is the MOPSO, while DE outperforms the original GA.

However, GODLIKE (with PSO excluded) seems to have little problems with this; indeed, in all subsequent trials with the GODLIKE algorithm, it seemed to always be able to find a good spread of solutions that lie on the top-leftmost branch. Of all algorithms, GODLIKE seems to be the most stable for MOO (provided PSO is excluded).

In conclusion, using PSO or ASA individually for MOO is not advisable at this stage. GODLIKE does seem to perform well, even with SA included, provided that the PSO is *not* included; PSO seems to constantly lead solutions a small distance *away* from the front. This is likely due to a slight implementation flaw; upon re-initializing the swarm in GODLIKE, it is quite difficult to also keep track of the locations and values of the local bests *lbest* of the previous swarm, as these values do not exist in the DE/GA/SA populations. This basically restarts the swarm completely. Although this flaw could be improved, when looking at the performance of PSO by itself (which does not share the same problem) it hardly seems worth the effort. Therefore, the GODLIKE algorithm using GA, SA and DE optimizers will be used for all multi-objective problems encountered in the design of the current mission.

Part III

High-Thrust Mission

9

The High-thrust MGA-Problem

NASA's Mariner 10 spacecraft, launched in 1974, was the first spacecraft ever to explore Mercury. The planet Mercury is a notoriously difficult target for space mission designers, not merely because it is so close to the Sun, but primarily because orbit acquisition around Mercury requires more propellant than most other interplanetary missions. As a remedy for this, Mariner 10 employed a previously untried method. Mariner 10 was not only the first spacecraft to visit Mercury, but also the first spacecraft to use the "gravitational slingshot effect" to reach another planet [Dunne and Burgess, 1978].

This method basically opened up the outer Solar system for exploration by spacecraft. Before its invention, it was widely believed to be impossible to ever reach any planet or asteroid further removed than Mars, because using the most energy-efficient way known at the time (a Hohmann transfer) to reach any of those targets already required more than the world's most powerful rocket could provide. But since Mariner 10, nearly every interplanetary mission to the outer planets has employed the gravitational slingshot effect[1].

The energy requirements for the current mission are so high that the use of this technique seems unavoidable. Indeed, missions such as these employ multiple GAM's in sequence, to increase the total energy in a step-wise manner. Logically, the more GAM's are used, the more energy is gained at the cost of an increased mission time. Designing trajectories in this framework is generally referred to as the MGA problem. Finding the *optimal* MGA trajectory is quite complicated, so the next two Parts of this thesis will be devoted entirely to this subject. This chapter will treat the MGA problem for high-thrust propulsion systems.

[1] Except for the missions *Ulysses* and *New Horizons*, which indeed hold the records for the first spacecraft to have ever accomplished a direct Hohmann transfer to one of the outer planets.

9.1. High-thrust Systems

The oldest and still most frequently used type of space propulsion, is the *chemical thruster* or *chemical rocket*. As the name implies, chemical rockets convert the energy stored in chemicals into thrust, and finally kinetic energy of the spacecraft. The term "high thrust" is used primarily in the field of trajectory optimization, to indicate that a spacecraft is equipped with a chemical rocket and to imply what type of optimization techniques can be used; it is only used to discriminate it from low-thrust propulsion.

Three types of chemical rockets can be distinguished. These three types are distinguished by the type of phase their propellants can be in. Chemical rockets can use either *liquid* propellants, *solid* propellants, or a *hybrid* of the two. Rockets that use liquid propellants generally have the highest specific impulse (300–400 s), followed by hybrid (275–300 s) and the solid propellants (200–270 s) [Zandbergen, 2004]. This difference in specific impulse stems forth from the higher combustion temperatures that are possible for liquid propellants, which is accomplished by combining two (or more) highly reactive vapors in the rocket's nozzle (see Figure 9.1). More importantly, liquid–propellant rockets can be throttled or even switched off mid-flight, which is not possible for solid–propellant engines. On the other hand, solid–propellant engines are much less complicated and expensive to manufacture. Therefore, solid–propellant engines are used almost exclusively for "boosters" that assist launch vehicles in their ascent from ground to orbit.

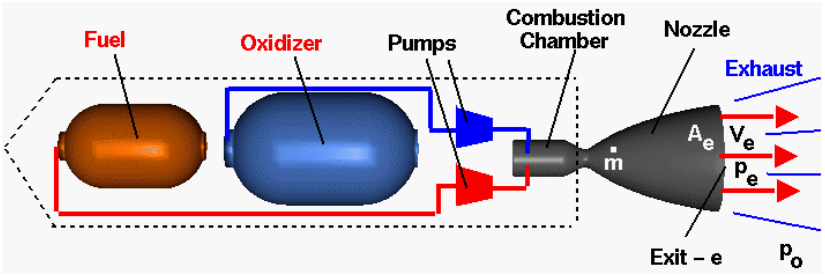


Figure 9.1 A schematic of a basic liquid-propellant rocket. From [Benson, 2009].

Liquid–propellant engines have been the *de-facto* on board propulsion system for decades. A small representative list of different kinds of liquid–propellant engines is given in Table 9.1. Their I_{sp} averages at ∼300 s – the value assumed in section 3.4.4 to be used in the optimizations. Also note that the nominal thrust levels are greater than 400 N for most engines. These two observations justify both the assumed value for I_{sp} of 300 s, and the assumption of the impulsive-shot, to be explained in the course of this chapter.

9.2. Method of Patched Conics

The standard first-order method to solve high-thrust MGA problems, is the so-called Method of Patched Conics (MPC), since it simply "patches" together parts of different conic sections.

Table 9.1 Different types of on-board liquid rocket engines, and their masses and specific impulses. Adapted from [Melman, 2002, Table 5.1].

Engine	Thrust [N]	I_{sp} [s]	mass [kg]
RS-21	1330	294	8.39
R-42	890	305	4.54
R-4D	489	310	3.76
DM/LAE	445	315	4.54
MMBPS	445	302	5.22
ADLAE	445	330	4.50
HPLAM	445	325	4.60
S400/1	400	303	2.8
RS-25	111	285	0.96
R-6C	22.0	290	0.67
S10/1	10.0	287	0.35
RS-45	4.5	300	0.73

The MPC already provides quite accurate estimates of the final trajectory, while being relatively simple, straightforward and computationally cheap.

In the MPC, the motion of a spacecraft is initially assumed to be entirely dominated by the Sun. That is, all gravitational interactions between the spacecraft and (minor) planets is neglected (even when encountering such a planet), and no other forces besides the Sun's gravity are taken into consideration. With this simplification, the spacecraft's trajectory through the Solar system can be treated as a two-body problem and can thus be designed by using conic sections, and the methods developed in section A.1 may be applied.

To design an interplanetary mission using the MPC, a sequence `seq` of υ planets is selected and the interplanetary trajectory is divided up into so-called *legs*. Each leg (here: part of a conic section) describes a portion of the whole trajectory; it starts at one planet, and ends at the next planet in `seq`. Usually, two subsequent planets from `seq` are different, but this is not a necessity – it is perfectly possible to use the same planet for both the starting and endpoint of one leg.

The shape of each leg depends on the desired t_f between the two planets, and the positions of the departure and target planet at these times. If the spacecraft is propelled by engines that are designed to give relatively short but powerful bursts, the sub-problem of determining the shape of a leg is said to be a *high thrust* problem (see section 9.1), and its solution must be found by solving the corresponding *high thrust orbital boundary value* problem.

Generally, the shapes of two subsequent legs are different, so that the velocity at the end of one leg is also in a different direction than the initial velocity of the next. This is accomplished by using a GAM (see section A.4) at the target/departure planet where this discontinuity occurs. When performing such a GAM, the planet in question is assumed to be a point mass, and the two legs are assumed to be connected via a hyperbolic trajectory around the swingby-planet, starting and ending at the edge of the planet's Sphere of Influence

(SOI) (see section A.2.2).

9.3. The High-thrust Orbital Boundary-value Problem

The *High Thrust Orbital Boundary-Value Problem*, better known as *Lambert's problem*, is defined as follows (see also Figure 9.2):

> find the conic section that connects the points in space P_1 and P_2, in a given transfer time t_f.

It turns out that this is quite a challenging problem to solve *generally*, that is, making no *a priori* assumptions on the shape of the orbit (just $e \geq 0$), and no assumptions on the multiplicity of the solutions ($\Delta\theta + 2m\pi$, $m = 0, 1, 2, ...$).

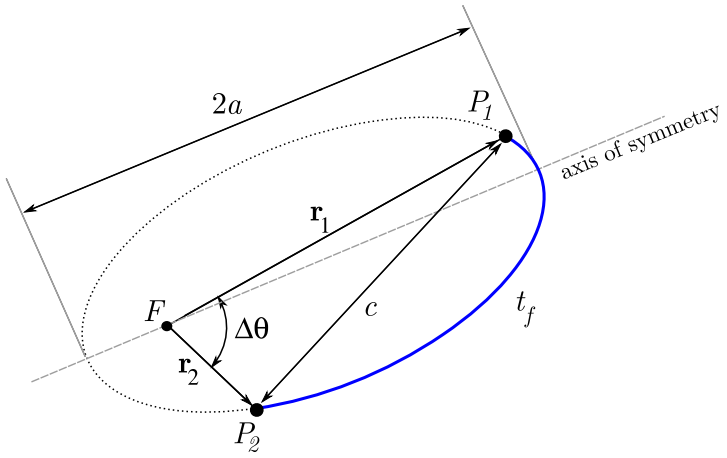


Figure 9.2 The geometry of Lambert's problem. The solution for given transfer-time t_f does not depend on the eccentricity, but only on the semi-major axis a, the radii r_1 and r_2, the chord length c and the covered angle $\Delta\theta$.

As shown in [Prussing and Conway, 1993], as was originally derived by Johann Heinrich Lambert (1728–1777), the solution for elliptic cases depends only on the semi-major axis a, the turn angle $\Delta\theta$ spanned by the vectors r_1 and r_2, and the chord length $c^2 = r_1^2 + r_2^2 - 2r_1 r_2 \cos\Delta\theta$ (see Figure 9.2). This statement is better known as *Lambert's Theorem*. It may be proved by a transformation of the equation for the transfer time (see also section F.1)[Prussing and Conway, 1993]

$$t_f = \sqrt{\frac{a^3}{\mu}} \left(E_2 - E_1 + \sin E_1 - \sin E_2\right) \tag{9.1}$$

into

$$t_f \sqrt{\mu} = a^{3/2} \left(\alpha - \beta - (\sin\alpha - \sin\beta)\right) \tag{9.2}$$

where

$$\begin{aligned} \sin\left(\frac{\alpha}{2}\right) &= \sqrt{\frac{s}{2a}} \\ \sin\left(\frac{\beta}{2}\right) &= \sqrt{\frac{s-c}{2a}} \\ s &= \frac{r_1 + r_2 + c}{2} \end{aligned}$$

It turns out that Lambert's theorem may be extended to include multi-revolution cases ($\Delta\theta \leftarrow \Delta\theta + 2m\pi, m = 1, 2, ... \neq 0$) and parabolic and hyperbolic orbits ($e = 1$ and $e > 1$ respectively). Finding solutions to this *general* problem is a daunting task. Fortunately, many different algorithms that solve this problem have been developed over the years, and a particularly efficient one is the algorithm developed by Lancaster and Blanchard [1969]. The efficiency and elegance of this algorithm make it stand out from most others, so it is indeed the default Lambert targeting routine used by Skipping Stone. Unfortunately, it is too lengthy to discuss in full here. For the interested reader: a complete treatise of the algorithm is included in appendix C.

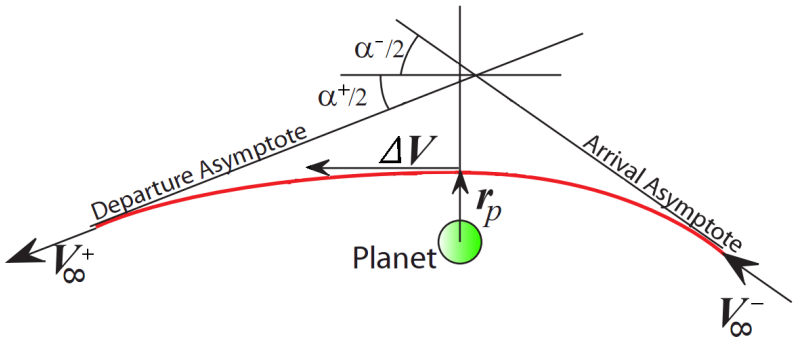
9.4. Powered Gravity-assist Manoeuvres

The most general high-thrust GAM does not only use the rotation of the V_∞-vector by the planet's gravity (see section A.4), but also makes use of the fact that a ΔV given in close proximity of a gravitating body is more effective than anywhere else. This may be proved by considering the increase in kinetic energy for any ΔV:

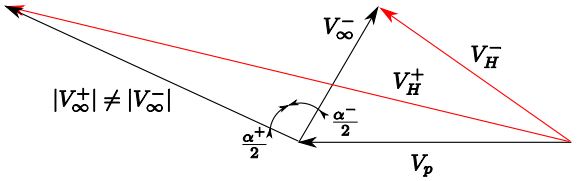
$$\Delta\text{KE} = \frac{(V + \Delta V)^2}{2} - \frac{V^2}{2} = \frac{\Delta V^2 + 2V\Delta V}{2},$$

which shows that the increase of kinetic energy is quadratic in ΔV, but more importantly, depends also on the velocity the spacecraft has at the time this ΔV is applied. Since an object in orbit always moves fastest at its pericenter (see section A.1), applying any ΔV near the pericenter will result in the largest change in the magnitude of V_∞ that can be achieved with this ΔV. Thus, when the spacecraft is performing a GAM, applying a (small) ΔV in the *pericenter* of the fly-by trajectory (see Figure A.7(a)) will make the GAM much more effective. This principle is known as the *Oberth-effect*, an effect that is also exploited in chapter 10 which explains how the effect can be used to gain considerable amounts of orbital energy by flying in close proximity to the Sun.

The geometry for a powered GAM is shown in Figure 9.3. Note that the turn angle α still is the only parameter determining the effectiveness of the GAM. However, because the velocity is changed at pericenter, the two hyperbolic legs now have different energies, and thus different shapes. The approaching V_∞ is referred to as V_∞^- and the departing V_∞ as V_∞^+. The turn angle α is now a sum of two *unequal* parts, which can be formulated as [Bernelli-Zazzera et al., 2007]



(a) The general geometry of a powered Gravity-assist Manoeuver. Adopted from [Melman, 2002, Figure 6.5].



(b) Vector diagram for the powered GAM. Note that the magnitude of V_H^+ has increased considerably compared to that in Figure A.7(b).

Figure 9.3 A schematic of the powered GAM. Because a ΔV is applied at pericenter, the two hyperbolic legs are not part of the same hyperbola, and the turn angle is composed of two unequal parts. But as shown in the lower figure, the gain in speed possible with this manoeuver is considerably greater than without the ΔV.

$$\alpha = \frac{\alpha^+}{2} + \frac{\alpha^-}{2} \tag{9.3}$$

$$= \arcsin\left(\frac{a^+}{a^+ + r_p}\right) + \arcsin\left(\frac{a^-}{a^- + r_p}\right) \tag{9.4}$$

$$= f(r_p), \tag{9.5}$$

where $a^+ = \mu/\left(\boldsymbol{V}_\infty^+ \cdot \boldsymbol{V}_\infty^+\right)$ and $a^- = \mu/\left(\boldsymbol{V}_\infty^- \cdot \boldsymbol{V}_\infty^-\right)$. From this it may be concluded that the ΔV applied at pericenter not only increases the magnitude of V_∞^+ considerably, but also allows the turn angle α to assume significantly different values than is possible without the application of this ΔV. This fact can again increase the effectiveness of a GAM, as is shown in Equation A.32.

In the patched-conics approach, the turn angle α is prescribed by the corresponding two consecutive solutions to the Lambert-problem (see section 9.3). This is also true for the approaching- and departing V_∞-vectors. Therefore, all quantities in Equation 9.3 are *fixed* by the positions of the planets as they are at the different prescribed travel times t_f. This means that for a certain configuration, the pericenter distance r_p is defined by Equation 9.3.

Finding its value is however somewhat of a challenge. It is given *implicitly*, so that some root-finding algorithm must be applied. Although the vast majority of the MGA computation-time

will be spent in the Lambert targeting routine, calculating r_p from Equation 9.3 must also be done *very often* and thus a fast convergence rate and stability of the solution are of primary concern. The Newton-Raphson iteration scheme is the obvious choice, since the function is monotonously decreasing, and the derivative is well-behaved.

However, a large number of numerical experiments (each with different values for a^+ and a^-) showed that the Newton-Raphson scheme with an initial value halfway between the pericenters defined by both hyperbolae,

$$\begin{aligned} r_p^0 &= \frac{r_p^+ + r_p^-}{2} \\ &= \frac{\frac{a^+}{\sin(\alpha/2)} - a^+ + \frac{a^-}{\sin(\alpha/2)} - a^-}{2}, \end{aligned} \tag{9.6}$$

(see [Bernelli-Zazzera et al., 2007] for details) can converge rather slowly, and might even fail sometimes. This is primarily due to the fact that the limit of the derivative of Equation 9.3,

$$\begin{aligned} \lim_{r_p\to\infty} \frac{df(r_p)}{dr_p} &= \lim_{r_p\to\infty} -\frac{a^+}{(a^+ + r_p)^2\sqrt{1 - \frac{a^+}{(a^+ + r_p)^2}}} - \frac{a^-}{(a^- + r_p)^2\sqrt{1 - \frac{a^-}{(a^- + r_p)^2}}} \\ &= 0, \end{aligned} \tag{9.7}$$

so that for poorly chosen r_p^0 the first Newton-Raphson step will result in a severely underestimated new estimate for r_p. This behavior can result in poor convergence (or complex-valued end results, in case the next iterate is negative). Since the solution is known to lie in between r_p^+ and r_p^-, taking these values as the initial values for the Regula-Falsi scheme is an easy alternative. However, as [Bernelli-Zazzera et al., 2007] hinted at, a much better solution would be to find the root of

$$\begin{aligned} g(r_p) &= \frac{1}{f(r_p)} - \frac{1}{\alpha} \\ &= \frac{1}{\arcsin\left(\frac{a^+}{a^+ + r_p}\right) + \arcsin\left(\frac{a^-}{a^- + r_p}\right)} - \frac{1}{\alpha} \end{aligned} \tag{9.8}$$

for which

$$\begin{aligned} \lim_{r_p\to\infty} \frac{dg}{dr_p} &= \lim_{r_p\to\infty} \left(\frac{\frac{a^+}{(a^+ + r_p)^2\sqrt{1 - \frac{a^+}{(a^+ + r_p)^2}}} + \frac{a^-}{(a^- + r_p)^2\sqrt{1 - \frac{a^-}{(a^- + r_p)^2}}}}{\left(\arcsin\left(\frac{a^+}{a^+ + r_p}\right) + \arcsin\left(\frac{a^-}{a^- + r_p}\right)\right)^2} \right) \\ &= \lim_{r_p\to\infty} \left(\frac{f'(r_p)}{f^2(r_p)} \right) \\ &= \frac{1}{a^+ + a^-}. \end{aligned} \tag{9.9}$$

Since r_p is generally quite large (the smallest permissible value is ∼2400 km for Mercury), the "∞" appearing in the limit above is already well approximated for most values of r_p. This is illustrated by the numerical example given in Figure 9.4. This figure shows both functions $g(r_p)$ and $f(r_p)$ for a GAM around the Earth, with

$$\begin{aligned} 0 \leq \; r_p \; &\leq 10,000 \text{ km}, \\ V_\infty^+ \; &= \; 10 \text{ km/s}, \\ V_\infty^- \; &= \; 6 \text{ km/s, and} \\ \alpha \; &= \; 75°. \end{aligned}$$

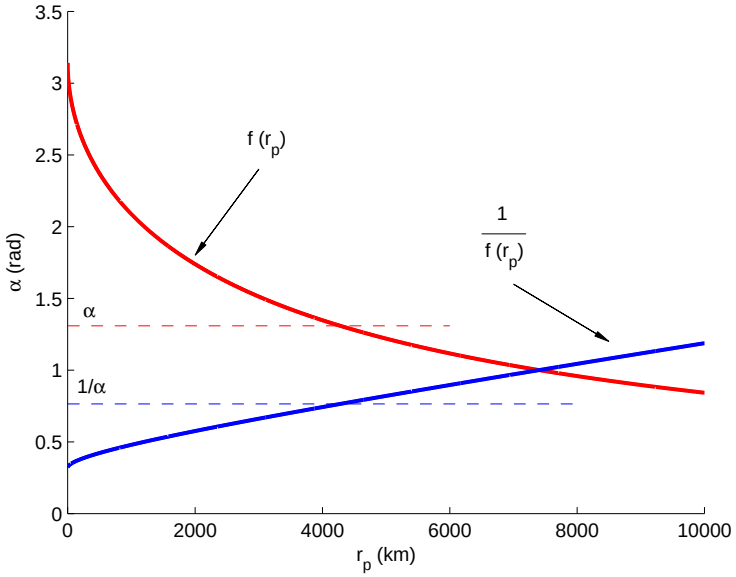


Figure 9.4 A quick way to find the pericenter radius r_p.

As can be seen in this figure, the function $g(r_p)$ is quite well approximated by a straight line with slope $1/(a^+ + a^-)$. Therefore, the pericenter r_p is most easily found by setting

$$\begin{aligned} \hat{r}_p^0 \; &= \; \frac{r_p^+ + r_p^-}{2}, \\ r_p^0 \; &= \; \hat{r}_p^0 - \frac{1/f(\hat{r}_p^0) - 1/\alpha}{f'(\hat{r}_p^0)/f^2(\hat{r}_p^0)} \qquad (9.10) \\ &= \; \hat{r}_p^0 - \frac{\left(1 - f(\hat{r}_p^0)/\alpha\right) f(\hat{r}_p^0)}{f'(\hat{r}_p^0)} \qquad (9.11) \end{aligned}$$

with $f(r_p)$ and $f'(r_p)$ as in Equations 9.3 and 9.7, respectively.

In many cases, the value for r_p thus found will be much *lower* than the minimum allowed value $r_{\min}$. In such cases, the maximum allowed turn angle $\alpha_{\max}$ must be used, which follows from

$$\alpha_{\max} = \arcsin\left(\frac{a^+}{a^+ + r_{\min}}\right) + \arcsin\left(\frac{a^-}{a^- + r_{\min}}\right). \tag{9.12}$$

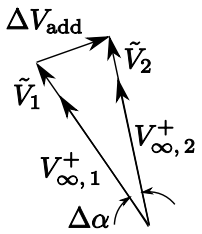


Figure 9.5 Geometry for the application of an additional corrective ΔV.

The turn angle that is required by the configuration of the planets, α_{req}, must be then met by an additional rotation of the departing V_∞-vector over an angle

$$\Delta\alpha = \alpha_{\text{req}} - \alpha_{\max}.$$

This may be achieved by applying an additional ΔV_{add} at some other point than pericenter in the hyperbolic trajectory. From the geometry shown in Figure 9.4, this additional ΔV_{add} has magnitude

$$\begin{aligned} \Delta V_{\text{add}} &= 2|\tilde{\boldsymbol{V}}|\sin(\Delta\alpha/2) \\ &\approx 2|\boldsymbol{V}^+_\infty|\sin(\Delta\alpha/2) \end{aligned} \tag{9.13}$$

where the vector $\tilde{\boldsymbol{V}}$ indicates the instantaneous velocity. The magnitude of ΔV_{add} thus depends on the instantaneous speed $\tilde{V} = |\tilde{\boldsymbol{V}}|$. Theoretically, this means it should be applied *infinitely* far from the planet to be at its smallest value, since in a hyperbolic trajectory in a one-body model, $\tilde{V} \downarrow V_\infty$ if $r \uparrow \infty$, with r the instantaneous radius. In the MPC-framework, this implies ΔV_{add} must be applied at the *edge* of the SOI of the flyby body, so that

$$\tilde{V}^2 = \left(V^+_\infty\right)^2 + \left(V^2_{\text{esc}}\right)_{\text{SOI}} = \left(V^+_\infty\right)^2 + \frac{2\mu}{R_{\text{SOI}}}. \tag{9.14}$$

The last term in Equation 9.14 is generally small compared to $(V^+_\infty)^2$, because the radii of the SOI's are so large (see Table A.1). However, taking the Earth as an example,

$$\frac{2\mu_E}{R_{\text{SOI}}} = \frac{2 \cdot 398600.4}{9.2412179 \times 10^5} \approx \frac{8 \times 10^5}{1 \times 10^6} = 0.8\,\text{km}^2/\text{s}^2,$$

which shows it is indeed small, but certainly not negligible in cases where V^+_∞ is relatively small. Therefore, since it is only a little extra effort, this term will be taken into account. This (constant) term has been calculated for all the planets, the results can be found in Table 9.2.

9.4.1 Effectiveness of a powered GAM

To get an idea of how effective a powered GAM is compared to an un-powered one, rough estimates similar to the ones made in section A.4.1 can be made here. Note that the equation derived for the energy gain in an un-powered GAM (Equation A.33) has to be modified to incorporate a ΔV at pericenter. To that end, observe that

Table 9.2 $(V_{\text{esc}}^2)_{\text{SOI}}$ for all planets.

Planet	$(V_{\text{esc}}^2)_{\text{SOI}}$ (km^2/s^2)
Mercury	0.3920
Venus	1.054
Earth	0.8627
Mars	0.1484
Jupiter	5.253
Saturn	1.389
Uranus	0.2241
Neptune	0.1582

$$V_{\text{peri}}^{+} = \sqrt{(V_{\text{esc}}^{R})^2 + (V_{\infty}^{-})^2} + \Delta V \tag{9.15}$$

$$V_{\infty}^{+} = \sqrt{\left(V_{\text{peri}}^{+}\right)^2 - (V_{\text{esc}}^{R})^2} \tag{9.16}$$

for the departure hyperbola. Then, using the expression for $\sin(\alpha/2)$ in Figure 9.3 as before, the energy gain for a powered GAM can be written as

$$\Delta\epsilon = \left(V_{\infty}^{+}\right)^2 - \left(V_{\infty}^{-}\right)^2 + V_p \left(\frac{a^{+}V_{\infty}^{+}}{a^{+} + r_p} + \frac{a^{-}V_{\infty}^{-}}{a^{-} + r_p} \right) \tag{9.17}$$

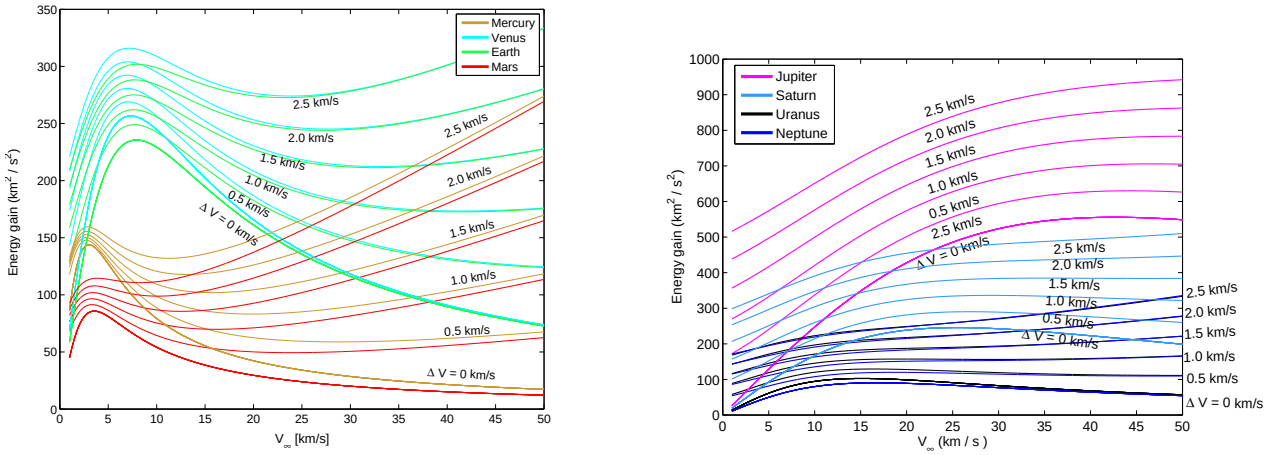
When setting $r_p = R_p$, plots similar to Figure A.8 can be obtained, for different values of ΔV (see Figure 9.6). From these Figures it can be concluded that for all planets, the GAM is more effective when the applied ΔV and/or V_∞ is higher. Note especially the changes in Mars and Mercury – when an un-powered GAM was assumed, these planets were hardly attractive targets (see Figure A.8), but the powered GAM makes them well worth considering. Note also the drastic change in Jupiter – its maximum energy gain more than *doubles* when a relatively modest ΔV of 2.5 km/s is applied.

9.5. Deep-Space Manoeuvres

A very interesting and promising technique that can be used in MGA problems, is the Deep Space Manoeuver (DSM). Including such a manoeuver in the MGA procedure greatly enlarges the accessible configurations, potentially decreasing the overall ΔV-budget by several km/s.

The idea behind a DSM is to not only give a ΔV at the pericenters of planets encountered, but also at one or more other points along the interplanetary transfer trajectories. This is illustrated in Figure 9.7. A relatively small ΔV, when given somewhere in the middle of a transfer trajectory, can have a large influence on the position of the spacecraft at later epochs, which means that targets which were hard to reach with powered GAM's alone, might now be reached with a much smaller ΔV-budget.

However, including such DSM's considerably complicates the MGA search space. According



(a) Inner planets

(b) Outer planets

Figure 9.6 The maximum possible energy gain from a powered GAM, for both the inner and outer planets.

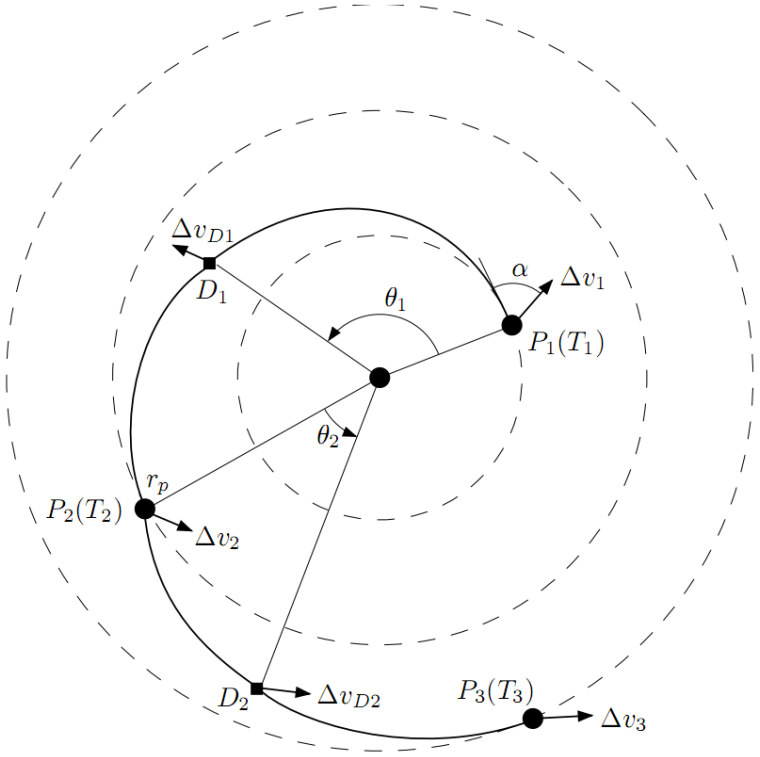


Figure 9.7 The basic principle of DSM's in the MGA framework. The ΔV budget required for the transfer from one planet to the next may be significantly reduced by using DSM's. Copied from [Bernelli-Zazzera et al., 2007, Figure 4.3].

to [Bernelli-Zazzera et al., 2007], each DSM introduces at least *four* additional dimensions to the search space: the *time* the DSM is executed (1 dimension), and the magnitude and direction of the thrust applied (3 dimensions). In much of the literature on the subject, the

third spatial dimension is normally ignored, that is, the entire trajectory from one planet to the next (including DSM's) is assumed to take place in a single plane. This assumption seems reasonable from an energetic standpoint (plane changes generally consume much energy) and reduces the required extra dimensions to "only" three[2]. Therefore, when considering DSM's, the search space complexity would be increased from $O(\mho^2)$ to $O((\mho + 3 \cdot M)^2)$, where M is the amount of DSM's applied over the whole trajectory.

For the current mission, high-thrust solutions to the MGA problem will be extensively studied to try and reach the SBS in a timely fashion. However, the mission requirements almost dictate that the spacecraft is equipped with a very high specific impulse propulsion system, so that the emphasis of this research will be on low-thrust propulsion. Therefore, because of the added search-space complexity, the difficulty of efficiently implementing a generalized DSM in the optimizations, and the fact that high-thrust propulsion will likely not result in the best candidate trajectory as the final result, using DSM's will be left as a recommendation.

[2]In some extreme examples, the DSM is always assumed to lie tangential to the instantaneous velocity, so that the search space complexity is increased by only 2 dimensions.

10 Solar Flyby

Strictly speaking, a GAM can only change the spacecraft's Heliocentric orbital energy if the spacecraft "steals" some *orbital energy* from a planet. In this light, it does not seem beneficial to perform a similar manoeuver using the Sun as the swingby body, for it obviously has *zero* orbital energy with respect to itself. However, the Oberth effect discussed in section 9.4 *can* be used to impose a large change in Heliocentric orbital energy; by applying a small ΔV in close vicinity to the Sun. Since the Sun is by far the most massive body in the Solar system, this somewhat "drastic" manoeuver is well worth considering when looking at Figure 10.2 – the energy potentially gained from such a manoeuver is the largest energy gain possible for any body in the Solar system for any given amount of ΔV (compare this figure with Figure 9.6; the corresponding figure for the powered planetary GAM).

10.1. General Considerations

10.1.1 Allowable Approach Distance

Although an Solar Flyby (SF) has the potential to increase the orbital energy considerably, it also has several strong disadvantages; the most obvious of which is the scorching intensity of the Solar radiation at small distances. A close proximity flyby of the Sun necessarily means an increase in the spacecraft's mass, because its instruments (which need to operate at peak efficiency $\sim$ 200 AU from the Sun) will have to be well-protected by a heat shield and some means of insulation. These protections against overheating would have to be carried along during the whole interplanetary phase, while they are only useful for an exceedingly small portion of the mission.

A back-of-the-envelope version of the conditions NASA's MESSENGER spacecraft is confronted with is examined here, in order to get a rough idea on what would be a realistic value to uphold for the minimum allowable distance to the Sun and what the mass of these

protective systems would add up to. The maximum temperature MESSENGER's heat shield reaches can be found on [NASA/JPL, 2009c]; its temperature peaks at 370° C when Mercury is at its pericenter, about 0.308 AU from the Sun. The spacecraft's thermal system's design allows the instruments on the interior to maintain a "comfortable" 20° C, even at these soaring temperatures just outside the heat shield. From the energy balance based on these data it follows that

$$\frac{\alpha P_0}{r^2} = \epsilon \sigma T^4 \tag{10.1}$$

$$\rightarrow \frac{\alpha}{\epsilon} = \frac{r^2 \sigma T^4}{P_0} = \frac{(0.3075)^2 \cdot 5.6704 \times 10^{-8} \cdot (370 + 273.15)^4}{1366} \approx 0.672 \simeq 0.7. \tag{10.2}$$

where $\sigma = 5.6704 \times 10^{-8}$ W m^{-2} K^{-4} is Stefan-Boltzmann's constant, $\alpha = (1 - \rho)$ is the heat shield's absorption coefficient (one minus the reflectivity, averaged over all wavelengths) and ϵ its emissivity coefficient, $P_0 = 1366$ W m^{-2} the Solar constant, r the distance in AU and T the temperature in K. The α/ϵ ratio can be used in the inverse formula to find r:

$$r = \sqrt{\frac{\alpha}{\epsilon} \frac{P_0}{\sigma T^4}}. \tag{10.3}$$

The somewhat high value for the α/ϵ ratio for MESSENGER's heat shield might be explained by the fact that this heat shield has several more functions than just shielding heat; one of the other primary functions is to provide protection against micro-meteorite impacts. This necessitates the use of multiple layers of strong, non-metallic materials that unfortunately increase the α/ϵ ratio. Moreover, the thermal system is designed to withstand this extreme environment continuously for more than a whole year (the nominal mission lifetime). Such demanding requirements are not present when performing a single SF during the current mission to the SBS; the heat shield is only used once and for a relatively short duration. It can therefore be assumed that the heat shield does *not* serve as the spacecraft's primary protection mechanism for micro-meteorite impacts and is entirely expendable – it can be discarded after the SF phase. For these reasons it is assumed that other materials can be used that have a lower overall α/ϵ ratio *and* that the thermal system will use means of carrying away the heat at the expense of the shield's structural integrity, which is likely to increase the efficiency of that process. In that light, it is reasonable to assume

$$\left(\frac{\alpha}{\epsilon}\right)_{\text{SBS mission}} = \frac{1}{1.25} \left(\frac{\alpha}{\epsilon}\right)_{\text{MESSENGER}} \tag{10.4}$$

$$T_{\text{max, SBS mission}} = 450°\text{C}. \tag{10.5}$$

Inserting these values into Equation 10.3 gives

$$r_{\min} \approx 0.218 \text{ AU} \tag{10.6}$$

or roughly 2/3 of Mercury's perihelion.

10.1.2 Novelty of the Solar Flyby

Another important consideration is that a high-thrust SF as described in this Chapter seems to be quite a novelty, as nothing could be found on it in literature. Therefore, the algorithm developed below is completely new, and no literature has been used during its development. The SF can therefore not be validated against any published results, so no data can be used to check if the final implementation is indeed correct. This makes any results produced with an SF less reliable and thus subject to a longer validation process in case the best result found for this mission will indeed use an SF.

10.2. Outline

An SF is fundamentally different from a normal GAM, and if the Sun is to be included at an arbitrary position in the trial-sequence `seq`, the MGA procedure needs to be adjusted to incorporate it. The Solar flyby problem can be stated as

> Find two trajectories Q_1 and Q_2 that connect the points P_1 and P_2 in a transfer time t_f. Q_1 and Q_2 are connected in the point $P_{\Delta V}$ located at $\boldsymbol{r}^{\odot}$, which is the point of application of an impulsive ΔV manoeuver that transforms Q_1 into Q_2.

This is shown graphically in Figure 10.1. The simplest approach to this problem would seem to be to solve the two corresponding Lambert-problems, and find the required ΔV from

$$\Delta V^2 = \boldsymbol{V}_{\text{end}}^{Q_1} \cdot \boldsymbol{V}_{\text{end}}^{Q_1} + \boldsymbol{V}_{\text{start}}^{Q_2} \cdot \boldsymbol{V}_{\text{start}}^{Q_2} - 2 \left| \boldsymbol{V}_{\text{end}}^{Q_1} \right| \left| \boldsymbol{V}_{\text{start}}^{Q_2} \right| \cos\lambda,$$

where $\boldsymbol{V}_{\text{end}}^{Q_1}$ is the terminal velocity from the solution to the first Lambert problem, $\boldsymbol{V}_{\text{start}}^{Q_2}$ the initial velocity of the second, and λ the angle between these velocities. The obvious problem is that the given parameters P_1, P_2 and t_f *do not fix* the point $\boldsymbol{r}^{\odot}$, so that this point has to be assumed and/or result from an independent optimization.

Such an optimization should at least maximize the effectiveness of the SF. With a bit of thought, it may be concluded that the maximum effect is achieved by minimizing the required ΔV by setting

$$\boldsymbol{V}_{\text{end}}^{Q_1} \times \boldsymbol{V}_{\text{start}}^{Q_2} = \boldsymbol{0}$$

so that $\cos\lambda = \pm 1$. In addition,

$$\boldsymbol{V}_{\text{end}}^{Q_1} \cdot \boldsymbol{r}^{\odot} = \boldsymbol{V}_{\text{start}}^{Q_2} \cdot \boldsymbol{r}^{\odot} = 0,$$

so that there are no gravity-losses. In other words, both terminal velocities are parallel to each other and exactly perpendicular to $\boldsymbol{r}^{\odot}$, which is only possible if $\boldsymbol{r}^{\odot}$ is the *pericenter* of both orbits/trajectories. This is indeed the same in a powered GAM, where the ΔV is applied at pericenter for the same reason (see section 9.4).

There is however a fundamental difference, which gives rise to some complications in the

analysis – in a powered GAM, both the approaching and departing legs are part of a *hyperbolic* trajectory, in which case the (known) turn angle may be used to find the pericenter distance. In the case of an SF, this fact is not known beforehand – both legs may be either *elliptic* or *hyperbolic*.

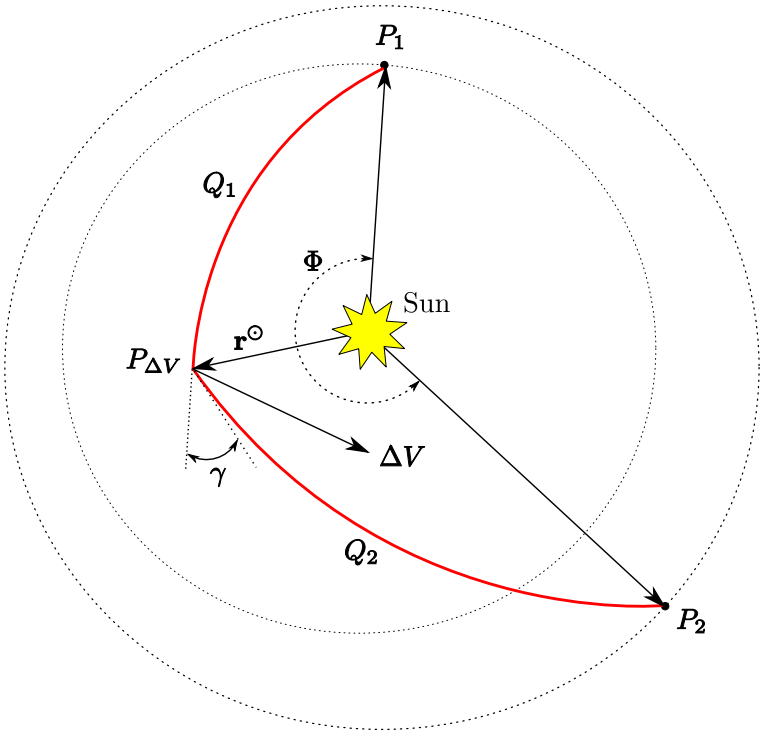


Figure 10.1 Overview of the geometry for the Solar flyby.

Denote the travel time from P_1 to $\boldsymbol{r}^\odot$ by t_1^f, and from $\boldsymbol{r}^\odot$ to P_2 by t_2^f. Normally, the points P_1 and P_2 are determined by the positions of the corresponding planets in the fly-by sequence (or the SBS), and thus these quantities are known. The quantities t_1^f and t_2^f are not known individually, but their sum $t_f = t_1^f + t_2^f$ is given and thus known. Let the semi-major axis of Q_1 be given by a_1, its eccentricity by e_1, and those of Q_2 by a_2 and e_2, respectively. Then the following equations apply:

$$r^\odot = a_1(1-e_1) = a_2(1-e_2), \tag{10.7}$$

$$r_{1,2} = \frac{a_{1,2}(1-e_{1,2}^2)}{1+e_{1,2}\cos\theta_{1,2}}, \tag{10.8}$$

$$\theta_1 + \theta_2 = \Phi, \tag{10.9}$$

$$\begin{aligned} t_f\sqrt{\mu_S} &= |a_1|^{3/2} \cdot \begin{cases} E_1 - e_1\sin E_1 & \text{if } e_1 < 1 \\ e_1\sinh F_1 - F_1 & \text{if } e_1 > 1 \end{cases} \\ &+ |a_2|^{3/2} \cdot \begin{cases} E_2 - e_2\sin E_2 & \text{if } e_2 < 1 \\ e_2\sinh F_2 - F_2 & \text{if } e_2 > 1 \end{cases} \end{aligned} \tag{10.10}$$

where the subscript $_{1,2}$ means either the first or the second leg, r_1 and r_2 are the magnitudes of the corresponding vectors, and Φ is the total turn angle between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ (see Figure 10.1).

The quantities $E_{1,2}$ and $F_{1,2}$ are related in the usual way to $\theta_{1,2}$ (see Equation F.2)[1]. Thus, for each of the two subproblems, there are 4 unknowns ($a_{1,2}$, $e_{1,2}$, $\theta_{1,2}$, and $r^{\odot}$) and 4 equations – This implies the system is fully determined, and the problem is thus solvable. As always, these governing equations are transcendental and must thus be solved iteratively.

Unfortunately there is a problem. Equation 10.9 shows that, since the total turn angle Φ can be found from the known quantity $\boldsymbol{r}_1 \cdot \boldsymbol{r}_2$, the quantities $\theta_{1,2}$ actually describe only *one* unknown. If θ_1 is taken to be the variable, every occurrence of θ_2 can then be written as

$$\theta_2 = \Phi - \theta_1.$$

This fact reduces the set of equations by 1, which renders the system under-determined. Also, Equations 10.7 and 10.8 can be combined to eliminate the eccentricities:

$$e_{1,2} = e_{1,2}(r^{\odot}, \theta_1) = 1 - \frac{r^{\odot}}{a_{1,2}} \tag{10.11}$$

$$a_{1,2} = a_{1,2}(r^{\odot}, \theta_1) = \frac{r_{1,2} \cdot r^{\odot} \cdot \cos\theta_{1,2} - (r^{\odot})^2}{r_{1,2}(1 + \cos\theta_{1,2}) - 2r^{\odot}}, \tag{10.12}$$

which again reduces the set of equations by 1. Note that by rewriting this last equation two free variables were implicitly created – θ_1 and $r^{\odot}$. Using these variables it can be concluded that the SF problem can be solved by finding a root of the bivariate function

$$\begin{aligned} f(r^{\odot}, \theta_1) = \; -t_f\sqrt{\mu_S} \;&+\; |\hat{a}_1|^{3/2} \cdot \begin{cases} \hat{E}_1 - \hat{e}_1 \sin\hat{E}_1 & \text{if } \hat{e}_1 < 1 \\ \hat{e}_1 \sinh\hat{F}_1 - \hat{F}_1 & \text{if } \hat{e}_1 > 1 \end{cases} \\ &+\; |\hat{a}_2|^{3/2} \cdot \begin{cases} \hat{E}_2 - \hat{e}_2 \sin\hat{E}_2 & \text{if } \hat{e}_2 < 1 \\ \hat{e}_2 \sinh\hat{F}_2 - \hat{F}_2 & \text{if } \hat{e}_2 > 1 \end{cases} \end{aligned} \tag{10.13}$$

Here, $\hat{e}_{1,2} = e_{1,2}(r^{\odot}, \theta_1)$, $\hat{a}_{1,2} = a_{1,2}(r^{\odot}, \theta_1)$ as in Equations 10.11 and 10.12, substituted with the relation $\theta_2 = \Phi - \theta_1$, and the functions $\hat{E}_{1,2} = E_{1,2}(r^{\odot}, \theta_1)$ and $\hat{F}_{1,2} = F_{1,2}(r^{\odot}, \theta_1)$ follow from substituting these functions into Equations F.2 in appendix F.

10.2.1 Potential Energy Gain

A plot similar to Figure 9.6 can be made to get an impression on the potential energy gain possible with a Solar flyby; see Figure 10.2. The point of application of the ΔV-manoeuver is again assumed to be at zero altitude $R = R_S$, taken equal to 700,000 km. The total energy gain depends on both the incoming and outgoing energies, which translates into the change of semi-major axis:

$$\Delta\epsilon = \frac{\mu_S}{2}\left|\frac{1}{|a_2|} - \frac{1}{|a_1|}\right| \tag{10.14}$$

[1]Lambert's equation (Equation 9.2) might seem more convenient to use because it is not troubled by the additional unknown e. However, Lambert's equation would require a long analysis on how the proper signs for α and β should be taken, since it can not determine these parameters uniquely from the problem statement alone. And, as will be shown shortly, the eccentricity will automatically drop out of the given equations anyway.

The semi-major axes of the arrival and departure trajectory may be found by using the vis-viva relation:

$$a_1 = \frac{-\mu_S}{V_1^2 - 2\mu_S/R_S}, \tag{10.15}$$

$$a_2 = \frac{-\mu_S}{(V_1 + \Delta V)^2 - 2\mu_S/R_S}. \tag{10.16}$$

or expressed differently,

$$\Delta\epsilon = \frac{\mu_S}{2}\left(\frac{\Delta V^2 + 2V_1\Delta V}{\mu_S}\right) \tag{10.17}$$

In Figure 10.2, V_1 was taken from the range

$$V_{R_S} \leq V_1 \leq V_{R_S} + 500 \qquad \text{(km/s)}$$

where $V_{R_S} = \sqrt{\mu_S/R_S} \approx 430$ km/s is the circular speed at zero altitude (this quantity gives rise to the gradually increasing slope). The applied ΔV was varied from 0 to 10 km/s, which is the Figure's x-axis. Note that in the bottom region (the green-to-blue gradient), both the arrival and departure trajectories are elliptic, whereas in the topmost region (the blue-to-red gradient) both are hyperbolic. Note also that the potential energy gain $\Delta\epsilon$ is considerably higher than a GAM at any planet can accomplish.

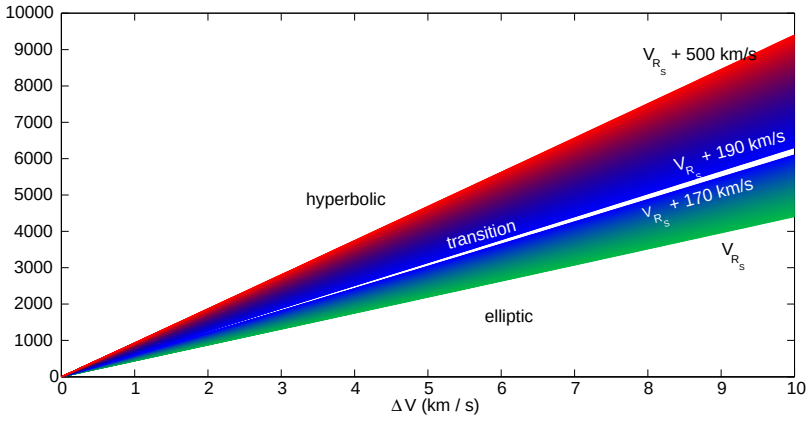


Figure 10.2 Potential gain in energy for a Solar GAM. The point where the ΔV was applied was assumed at $R = R_S \approx 700,000$ km. The total energy gain depends on the arrival and departure trajectories, which are both elliptic in the bottom half of the figure (labelled "elliptic"), and both hyperbolic in the top half (labelled "hyperbolic"). Note that the potential total energy gain is much larger than with planetary GAM's. The linearity of the figure is only an artifact of the short ΔV-scale: only if this is extended does the non-linearity of the equations become apparent.

10.3. Algorithm

What remains to be found is an efficient algorithm that finds roots for the bivariate Equation 10.13. This section presents such an algorithm, and discusses how to find suitable initial

estimates and what constraints are applicable to the problem. In addition it analyzes some of the algorithm's failure modes. In all of the following, the subscripts $_{1,2}$ are omitted (where applicable) for brevity.

10.3.1 Finding Solutions

Generally speaking, for any bivariate function, there can be either 0, 1, *or more* roots – indeed, there can be an infinite number of solutions to the SF problem. However, there is only *one* feasible solution that results in the *minimum* required ΔV at pericenter. Finding that single solution is therefore an optimization in itself.

As mentioned earlier in this chapter, the points P_1 and P_2 are usually the positions of planets directly prior and after the SF, respectively. Because these planets are used as swingby bodies, the optimization of an SF problem will not only have to minimize the required ΔV at $\boldsymbol{r}^{\odot}$, but also take into account three different sets of constraints:

$$\begin{aligned} \text{first set:} \quad & \text{where } Q_1 \text{ and } Q_2 \text{ meet:} \\ t_f - t_1^f - t_2^f &= 0 \end{aligned} \tag{10.18}$$

$$\begin{aligned} \text{second set:} \quad & \text{where } Q_1 \text{ departs, and similarly,} \\ \text{third set:} \quad & \text{where } Q_2 \text{ arrives:} \\ \Delta V_{\text{GAM}} - \Delta V_{\text{GAM}}^{\max} &\leq 0 \\ r_p - (R_{\text{mean}} + h_{\min}) &\geq 0. \end{aligned} \tag{10.19}$$

The latter four constraints were discussed at length in chapter 9, i.e., they constitute the set of constraints that apply to any normal swingby. Note that the ΔVs used during the powered swingby's are *not* included in this optimization; those ΔVs are already handled by the patched conics routine. In conclusion: solving an SF problem is equal to a *constrained* optimization problem, that is to find the minimum ΔV at pericenter, while satisfying all constraints at both swingby planets and those constraints that make the SF-problem physical.

To get a rough idea of the shape of its search space, a generic SF-search space is given in Figure 10.3. This figure also shows the rather large differences between the ΔVs for the different feasible solutions. The sequence used to generate this figure was [Earth, Venus, Sun, Earth, Mercury, Mars], and the total travel time was fixed to 50 days. The high values for the ΔV at pericenter near the edges of the feasible region can be explained by recognizing that the first orbit is free to use up a large fraction of the total travel time (a "lingering" elliptic orbit), necessitating an extremely hyperbolic trajectory towards the second point. The transition of the slow elliptic orbit to the extremely hyperbolic trajectory requires a very high ΔV, but this scenario is still a feasible solution in theory. Naturally, in case such a situation is found to be the best solution, it can safely be discarded. To enforce this demand, a maximum allowable $\Delta V_{\max}$ is adopted just as in the MPC-approach, which gives the algorithm a constraint to obey at all times:

$$\Delta V - \Delta V_{\max} \leq 0. \tag{10.20}$$

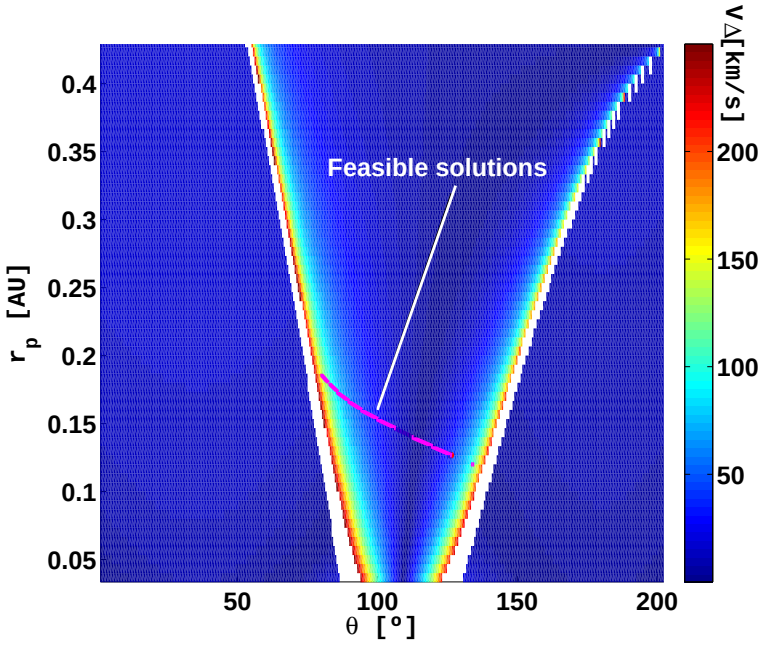


Figure 10.3 General shape of the search space for an SF problem. For nearly all such problems, the search space is quite similar to this one; the main differences are the location of the feasible region and the scaling (magnitudes of all ΔVs).

10.3.2 Preliminary Analysis

Naturally, to keep any SF technically feasible, a lower limit on the pericenter radius $r^{\odot}_{\min}$ should be upheld at all times; a suitable value has already been calculated in Equation 10.6. An important additional demand is that the eccentricities for both Q_1 and Q_2 should always remain *positive*. As can be seen in Equation 10.11, this implies

$$|a| \geq r^{\odot},$$

when a is positive, which translates into the fairly trivial

$$r^{\odot}_{\max} \leq \min\left(r_1,\, r_2\right).$$

The eccentricities are also positive when the corresponding a is *negative*, irrespective of its magnitude. Solving $a < 0$ for both Q_1 and Q_2 in Equation 10.12 fixes an interval of allowable values for θ_1:

$$\begin{aligned} \theta_1^{\min} &= \arccos\left(\tfrac{r^{\odot}_{\min}}{r_1}\right) \\ \theta_1^{\max} &= \Phi - \arccos\left(\tfrac{r^{\odot}_{\min}}{r_2}\right). \end{aligned} \tag{10.21}$$

For some geometries these limits can very well be inconsistent, i.e., $\theta_1^{\min} > \theta_1^{\max}$. For those cases, the initial lower limit $r^{\odot}_{\min}$ has to be shifted upward such that $\theta_1^{\min} = \theta_1^{\max}$. Equating

these quantities gives

$$r_{\min}^{\odot} = \frac{r_2 r_1 \sin\Phi}{\sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos\Phi}}. \tag{10.22}$$

This higher value for $r_{\min}^{\odot}$ might again give the inconsistent condition $r_{\min}^{\odot} > r_{\max}^{\odot}$. In those cases the posed SF problem has no solution, and the procedure can be terminated.

If however the condition $r_{\min}^{\odot} < r_{\max}^{\odot}$ still holds, there *might* be solutions; some further checks are necessary. On the interior of the interval for θ_1 as defined by Equations 10.21, Equation 10.12 might contain singularities. These are the locations where the sign changes in a occur – from hyperbolic ($a < 0$) via parabolic ($a \to \pm\infty$) to elliptic ($a > 0$), and/or back. The point(s) where this occurs follow from equating the denominator in Equation 10.12 to zero:

$$r\left(1 + \cos\theta_1\right) - 2r^{\odot} = 0 \tag{10.23}$$

$$\Rightarrow \theta_1 = \pm \arccos\left(\frac{2r^{\odot}}{r} - 1\right). \tag{10.24}$$

For clarity, θ_1 will be taken from the interval $0 \leq \theta_1 \leq 2\pi$, so that these points are to be computed as

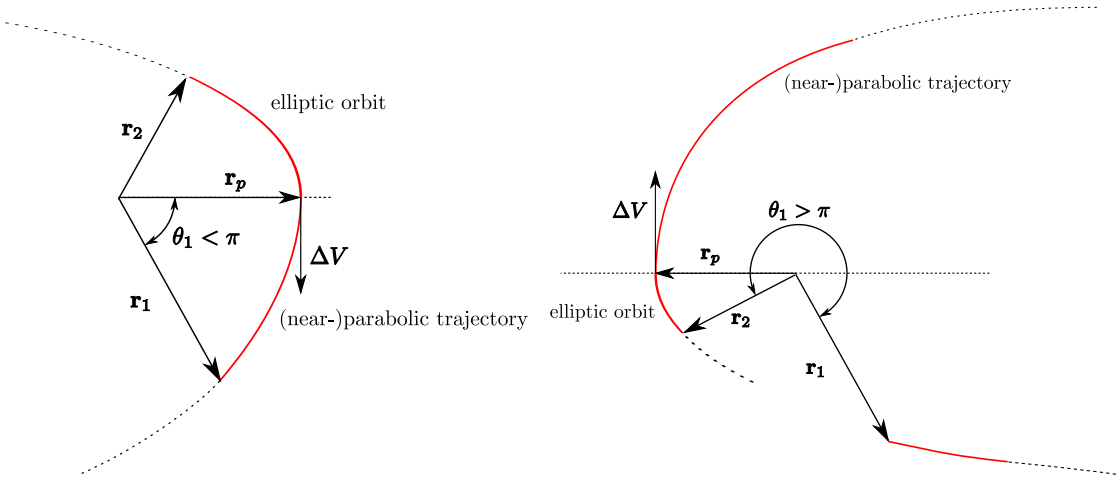
$$\theta_1 = \begin{cases} \arccos\left(\frac{2r^{\odot}}{r} - 1\right), \\ 2\pi - \arccos\left(\frac{2r^{\odot}}{r} - 1\right). \end{cases}.$$

This translates into four distinct values for θ_1 where singularities might occur; two for Q_1 and two for Q_2. However, only *two* of these four singularities will give rise to problems – those with $\theta_1 > \pi$. This can be understood by looking at Figure 10.4; only in case θ_1 is near a singularity *and* obeys $\theta_1 > \pi$ will the orbit also have to pass its *apocenter*, which, for near-parabolic orbits, requires a near-infinite amount of time. The times-of-flight at the other two singularities will remain finite (and their derivatives also continuous); these simply correspond to the transition from hyperbolic solutions to elliptic solutions, or vice versa.

The algorithm will therefore have to check if the θ_1-interval defined by Equation 10.21 contains any singularities in $a(r^{\odot}, \theta_1)$ that are also larger than π. This gives rise to either 1, 2 or 3 *sub*-intervals inside the original interval – 1 subinterval if none of these singularities falls inside the interval (the sub-interval simply spans the entire interval), 2 if it only contains one of the singularities, and 3 if it contains both.

10.3.3 Generating Initial Values

Generating initial values is necessarily an iterative process, since it is not known beforehand what values for θ_1 and $r^{\odot}$ are close to the minimum ΔV while also satisfying constraint 10.18. First it is important to point out that this iterative process should only give a small overhead to the actual optimization, so the amount of allowable iterations should be kept quite low. In this research a maximum of 5 iterations was allowed before the SF routine was forced to



(a) Type I singularities: If $\theta_1 < \pi$, t_f remains finite.

(b) Type II singularities: If $\theta_1 > \pi$, $t_f \to \infty$

Figure 10.4 The two types of singularities in the SF-problem. The first type arises from a "normal" transition from an elliptic orbit to a hyperbolic trajectory. The second type arises from the same transition, but now the trajectory must first pass its apocenter. Only the second type will remain present in the equation for the time-of-flight.

give up and return its result-on-failure. However, during all tests conducted, the completed SF routine *never* returned this specific error, proving that the routine to find the initial value discussed here is quite effective. Because the routine is somewhat involved, the following description is also illustrated in Figure 10.5.

It is very reasonable to assume that the minimum ΔV at pericenter occurs at low values of the pericenter distance $r^{\odot}$; the Oberth-effect is more prominent at small distances to the body in question. Therefore the initial estimate on the pericenter distance $r^{\odot}$ should be taken equal to $r_0^{\odot} = r_{\min}^{\odot}$, which follows from either Equation 10.6 or Equation 10.21. Then for each of the 1, 2 or 3 sub-intervals in the interval given by Equation 10.21, the corresponding value for θ_1 should be found that solves Equation 10.18. There may be 0, 1, or more of such roots, so the Chebyshev-root finding method (see appendix E) is the most cost effective way to find all of these (possible) roots at once.

In case one or more roots are found, the corresponding value for ΔV can be calculated from

$$\Delta V = \left| \sqrt{\frac{2}{r^{\odot}} - \frac{1}{a_1}} - \sqrt{\frac{2}{r^{\odot}} - \frac{1}{a_2}} \right|, \tag{10.25}$$

which follows from rearranging the two vis-viva equations corresponding to Q_1 and Q_2 at the point $\boldsymbol{r}^{\odot}$. The values for a_1 and a_2 follow from substituting the values for $r_0^{\odot}$ and θ_1^0 at the specific roots thus found into Equation 10.12. The specific root that returns the smallest value for ΔV (irrespective of the values of the other constraints at that point) is then the initial value.

If however *no* roots were found, all of the above must be repeated for a different value for $r_0^\odot$. New values for $r_0^\odot$ are calculated with an "educated guess":

$$r_0^\odot \leftarrow r_0^\odot - \frac{t_{f,\min}}{\partial t_f / \partial r^\odot},$$

which is basically a "naive" Newton-Raphson step. The partial derivative $\partial t_f / \partial r^\odot$ is found by differentiating Equation 10.18 with respect to $r^\odot$ while taking into account all substitutions that were made to come to this form, and $t_{f,\min}$ is the minimum of Equation 10.18 on all sub-intervals. The minimum value (when finite) was found to give the best results, however, finding the *true* minimum would require an additional computation of the roots of the partial derivative. Simply taking the smallest among all function values evaluated at the *midpoints* of the sub-intervals gave results of only slightly lesser quality, but with much improved robustness. Especially without the additional root-finding involved, this method is overall much more efficient.

This iterative process is repeated until a combination $[r_0^\odot, \theta_1^0]$ is found that yields at least one root of Equation 10.18. Naturally, during all iterations, the limits $r_{\min}^\odot < r^\odot < r_{\max}^\odot$ should be maintained. This provides a mechanism to convey some information to the user about the specific failure mode, in case one of these limits is violated. If $r_0^\odot = r_{\min}^\odot$ and the Newton-Raphson step suggests that $r_0^\odot$ should be reduced even further indicates that the given t_f is too small – the user-specified time-of-flight is simply too short to solve the SF-problem. Similarly, if $r_0^\odot = r_{\max}^\odot$ and the routine tries to increase $r_0^\odot$, the given travel time is too long to solve the SF problem. In both these cases the SF-routine is to return this information, in addition to its result-on-failure, and terminate.

10.3.4 Generalization

Although not directly part of this thesis research, it is fairly straightforward to *generalize* the algorithm described here so that it works in any arbitrary `seq` and any arbitrary set of input parameters.

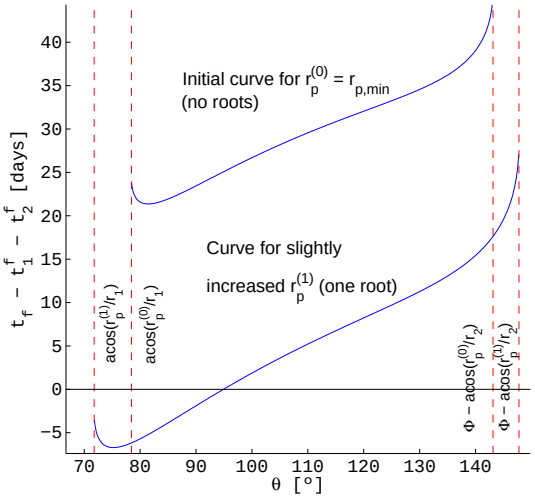
Departure and Arrival

The points P_1 and P_2 were assumed to be swingby planets throughout the analysis. Naturally, this is not general; P_1 can also describe the launch site, and P_2 the final target body. If one or both of these cases is true, the SF-problem has to take into account a *different* set of constraints. In case P_1 is the launch site, Equation 10.19 for the second set of constraints must be changed into

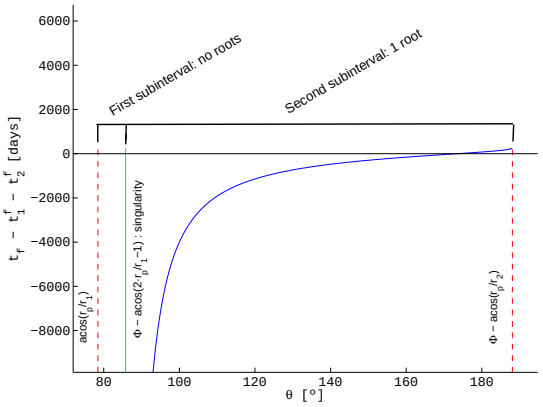
$$\text{Second set:} \quad \text{Launch site:} \tag{10.26}$$

$$C_3 - C_3^{\max} \leq 0. \tag{10.27}$$

And in case P_2 is the final target body, the third set must be changed into



(a) No singularities, single sub-interval



(b) One singularity, two sub-intervals (first one falls outside the figure's boundaries)

Figure 10.5 Routine to find suited initial values $r_0^{\odot}$ and θ_1^0. The parameters used to generate these figures are

Leftmost Figure :

$\boldsymbol{P}_1$: [1, 0, 0] AU
$\boldsymbol{P}_2$: [−1, −1, 0] AU
t_f : 0.25 year (long way)

Rightmost Figure :

$\boldsymbol{P}_1$: [1, 0, 0] AU
$\boldsymbol{P}_2$: [2.5, 19, 0] AU
t_f : 1.00 year (long way)

$$\text{Third set:} \quad \text{target site:} \tag{10.28}$$

$$C_{3,\text{capture}} - C_{3,\text{capture}}^{\max} \leq 0, \text{ or} \tag{10.29}$$

$$\alpha_{\text{approach}} - \alpha_{\max,\text{approach}} \leq 0, \text{ or} \tag{10.30}$$

$$\ldots$$

So basically all the constraints that deal with the arrival conditions. These are of course highly dependent on the type of mission (orbit insertion/entry/flyby/...), but should be satisfied nonetheless.

Multiple Solar Flybys ($m > 0$)

There is another possibility – that P_2 is *again* the Sun (see Figure 3.4 for example). In such cases it is better to call the problem a *multi-revolution* SF-problem, in analogy to the multi-revolution Lambert-problem. In case multiple SF's are demanded ($m > 0$), some additional considerations are necessary. The exact procedure for multiple SF's depends on the characteristics of the swingby-sequence `seq` (see section 9.2). Three different cases must be distinguished, depending on the multiplicity m:

Non-consecutive multiple flyby's When the fly-by sequence holds two or more SF's, but no two are adjacent (e.g., `seq`= [Earth, Sun, Venus, Sun, Jupiter, Sun, Bowshock]),

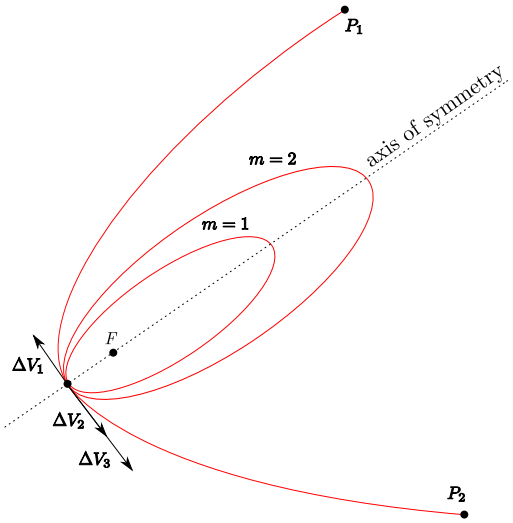


Figure 10.6 Performing multiple Solar flyby's. In the shown case, the pericenter is visited 3 times. Note that all orbits share the same axis of symmetry.

there is enough information to apply the procedure normally – no special measures need to be taken.

Consecutive multiple flyby's, $m = 1$ In case the flyby sequence has precisely 2 consecutive SF's (e.g., `seq`= [Earth, Sun, Venus, Sun, Sun, Jupiter, Bowshock]), the first SF has no initial or end-point to refer to. In this case, the end-point of the second SF (in mentioned example, Jupiter) may be used as a reference point to determine the pericenter for the first SF. The Sun-Sun leg in between will then be flown in an elliptic orbit, with a period equal to the prescribed transfer time, and a pericenter equal to the point found from the Sun-approaching- and Sun-departing legs. The ΔVs needed to acquire this orbit will then be the end result. This process is shown in Figure 10.6, with $m = 1$.

Consecutive multiple flyby's, $m > 1$ **(General case)** The same problem applies to more than two consecutive SF's (e.g., `seq`= [Earth, Sun, Sun, Sun, Jupiter, Bowshock]); not enough information is available to fully determine the resulting trajectories. In such a case, the same procedure as in the $m = 2$ case may be applied – flying $m - 1$ complete elliptic orbits in between the Sun-approaching- and Sun-departing legs. In this case however, several ΔVs are necessary at the pericenter for each intermediate elliptic orbit, to modify the periods of the intermediate elliptic orbits and meet the prescribed travel times. This general case (until $m = 2$) is shown in Figure 10.6.

10.3.5 Implementation Issues

To solve the SF problem, a gradient-based optimization method such as SQP is most desirable because of the associated high efficiency of such methods – the SF problem is only a small part of a global optimization, so in order to keep computation times as small as possible when including an SF in `seq` the optimization associated with every SF has to be executed

as quickly as possible. Although analytical gradients for Equation 10.10 can be found, computing gradient information for the constraints in Equation 10.19 is impossible. This fact forces the optimization method to resort to finite differences, which adversely affects its rate of converge and thus the required computation time. This is however not a problem in itself; vast amounts of numerical experimentation showed that using SQP with finite differences is still an order of magnitude faster than Nelder's/Mead's gradient-free method.

A much more pressing problem that occurred during these test runs was a complete failure of the SQP algorithm; failure in the sense that the SQP-optimizer ended up in an infinite loop, effectively halting the entire optimization process. The precise cause of this remains unknown, but it most likely occurs during the calculation of the gradients using finite-differences; these may slightly overshoot the boundaries of the interval, leading to negative eccentricity and thus non-numeric travel times and derivatives, or overshoot the tolerances set around the singularities discussed earlier this Chapter, leading to similar conditions. However, this problem persisted even after said tolerances were set to span half the interval. This issue must be addressed in any future work based on this research. For the remainder of this research the Nelder-Mead simplex method will be used, since that method did not show the same symptoms. The additional computation time must unfortunately be taken for granted.

11 Optimization Strategy

11.1. Dealing with the MPE in Global Optimizations

In full abstraction, the high-thrust MGA procedure discussed in chapter 9 can be written as follows:

$$F_{\text{cost}}\left(\texttt{seq}, t_0, \boldsymbol{t}_f, \boldsymbol{m}, \ldots\right) = \left[\Delta\boldsymbol{V}, \sum \boldsymbol{t}_f, \ldots\right] \tag{11.1}$$

where `seq` is the sequence of GAM-bodies, t_0 the launch date, $\boldsymbol{t}_f$ the vector of travel times between the bodies in `seq`, $\boldsymbol{m}$ the integer-valued vector that describes the number of complete revolutions per GAM, $\Delta\boldsymbol{V}$ is a vector of all ΔVs required to fly the trajectory, and $\sum \boldsymbol{t}_F$ is the total travel time to the target body.

The global optimizer evaluates this function for a large number of decision vectors of the form

$$\begin{aligned} X = [\ & t_0, \\ & \pm t_f^{(1)}, \quad \pm m^{(1)}, \\ & \pm t_f^{(2)}, \quad \pm m^{(2)}, \\ & \vdots \\ & \pm t_f^{(\upsilon)}, \quad \pm m^{(\upsilon)}]. \end{aligned} \tag{11.2}$$

As explained in chapter 9, negative t_f means taking the long way (as opposed to the short way for positive t_f), and negative m means taking the left branch of the two possible solutions (as opposed to the right branch for positive m).

11.2. Strategy per Scenario

11.2.1 Mission Scenario 1

Considering the general behavior of meta-heuristic global optimizers, it seems better to use penalty functions rather than re-initialization for trial solutions that violate one of the constraints; the value of the penalty functions provide *some* measure of how bad the trial solution actually was, and can prevent the global optimizer from spending large amounts of time re-initializing new trial solutions. This is particularly important for semi-directed optimization methods such as DE or PSO; penalized function values will direct individuals *away* from the associated regions at subsequent iterations, whereas re-initialization would spawn new individuals in the very same region at every iteration. Indeed, elaborate profiling, timing and tuning of a large number of runs on very constrictive problems in the fully functional version of Skipping Stone indicated that penalizing is *far* more efficient than re-initialization.

The value of the penalized MGA cost function for mission scenario 1 (or any "basic" case), F_{cost}, must be calculated according to the following steps:

Step 0 Select the sequence `seq` of planets (and/or Sun) to be used for the GAM's. Also select the launch date t_0 and transfer times t_f^i for each of the i legs that this sequence implies.

Step 1 The transfer times t_f^i in combination with the launch date t_0 fix the encounter dates for each of the planets (and/or Sun). Determine all the positions and velocities of the swingby bodies at their respective encounter dates.

Step 2 Solve the i Lambert problems associated with the transfer times t_f^i and the locations obtained from **Step 1**.

Step 3 For each GAM, calculate the minimum flyby distance r_p. To do so, first subtract the velocities of the swingby bodies themselves from the terminal velocities from the corresponding Lambert problems to obtain the approach and departure velocities with respect to the swingby body. From these velocities, the minimum distance r_p can be determined with Equation 9.11.

Step 4 For each r_p^i, check whether $r_p^i > r_{\text{min}}^i$. If this is not the case, use Equation 9.13 to calculate the additional ΔV_{add}. For every GAM, calculate the associated total ΔV^i required to perform the GAM.

Step 5 The total ΔV required to cover the whole sequence is a simple addition of all the ΔVs found in **Step 4**:

$$\Delta V_{\text{tot}} = \sum \left(\Delta V^i + \Delta V_{\text{add}} \right)$$

Now, the whole sequence must be penalized. To that end, compute the following quantities:

$$\begin{aligned} c_1 &= V_\infty^2 - C_3 \\ c_2 &= \Delta V_j - \Delta V_{\max}^j \\ c_3 &= \sum_{j=1}^{\upsilon} \Delta V_j - \Delta V_{\max} \end{aligned} \tag{11.3}$$

where V_∞ is the magnitude of the difference vector between the escape velocity at launch and Earth's velocity, ($C_3 = (V_\infty^{\max})^2$) the corresponding limit set by the selected launcher, ΔV_j the ΔV required at the j^{th} GAM, $\Delta V_{\max}^j$ the corresponding user-defined limit, and $\Delta V_{\max}$ the maximum *total* required ΔV for the whole trajectory. Note that the quantity c_3 serves as a safeguard, since ideally the values $\Delta V_{\max}^j$ at each flyby sum up to $\Delta V_{\max}$. However, the values for $\Delta V_{\max}^j$ should be adjustable by the user, and thus will not always sum up to said value.

Step 6 The value of the cost function the global optimizer will use, is equal to

$$F_{\text{cost}} = \Delta V_{\text{tot}} + \exp(c_1 + c_2 + c_3) - 1 \tag{11.4}$$

Note that the initialization **Step 0** is the most important part. Badly chosen launch dates and transfer times will result in a very high total ΔV, while other values (requiring the same total transfer time) will give much lower values. Therefore, these are the parameters to be optimized. Note also that for this study the departure planet is always the Earth, and the target "body" is always the location of the SBS (given by Equation 2.1).

The total ΔV is directly related to the spacecraft dry mass, which is a much more important parameter to optimize. It is more intuitive to return both the total ΔV and the final spacecraft mass m_{final} after (or during) an optimization. The latter quantity follows from the assumed engine parameters, ΔV_{tot} and Tsiolkovskii's equation (Equation A.31). Finally, when optimizing both the spacecraft end mass and the total time of flight (MOO), the second cost function is simply the sum of the times-of-flight:

$$F_2^{\text{cost}} = \sum_{j=0}^{\upsilon} t_f^j. \tag{11.5}$$

Choosing for the long-way or short-way solution can conveniently be implemented by passing *negative* values for some of the times-of-flight $\boldsymbol{t}_f$ into F; those that are positive will use the short-way solution, and those that are negative will use the long-way solution. When multiple revolutions ($m > 0$) are required for a specific leg, the associated Lambert problem will have two solutions. In a similar fashion, passing a *negative* but nonzero-m will return the solution associated with the left-branch in Figure C.1, and a positive value will return the one associated with the right-branch. Using this format for passing arguments to the MGA routine allows the global optimizer to automatically take the short or long way, the number of complete revolutions and the left or right branch solutions into account when trying to find the global optimum.

The range in which to choose the launch date t_0 is already defined by the launch window; it should lie between Jan. 1st, 2015 and Dec. 31st, 2025. Finding similar limits for the individual transfer times $\boldsymbol{t}_f$ between the different planets (and Sun) is however quite difficult; generally speaking, it is not even possible – there is no "minimum possible transfer time" (the spacecraft can assume any velocity lower than the speed of light), nor is there a "maximum possible transfer time" (multiple complete revolutions $m > 0$ can be flown before the next swingby-body is encountered). It is still very worthwhile to come up with a preliminary list of ranges, because the width of each range greatly influences the total size of the search space for the optimization. If set too wide, many "useless" calculations must be performed before realistic results will be found, leading to unacceptable calculation times. If set too narrow, the optimum solution might be missed. To this end, Table 11.1 is presented. This table lists the ranges for all travel times between the planets which will be used as the "default setting" in case these bodies are selected. These values are based on the required time of flight for a Hohmann-transfer between the bodies. The upper limit is twice this amount of time, the lower limit one-fifth. For transfers from bodies to themselves, the upper limit is once the body's orbital period, and the lower limit one-sixth. Solar flybys ("via Sun") uphold the same lower limit, but twice the upper limit as the associated regular transfer. Naturally, the upper limit on all values follows from the mission's constraint of 25 years (∼9131 days).

Note that Table 11.1 does not include Neptune, because Neptune is not in the right place for the time span of this mission as discussed in section 3.4.2. Also, many of the ranges have been omitted because the associated scenarios will simply never be used. For instance, flying from Uranus to Mars via the Sun will simply take too much time, however fast the spacecraft flies, and this sort of trajectory will thus not be included in the optimizations. Also,

11.2.2 Mission Scenario 2

Mission scenario 2 takes an optimized trajectory from mission scenario 1 as its initial value, and tries to perform a flyby at asteroids that come close to the spacecraft. Therefore, after performing some preliminary pruning on the MP data set as described in section B.4, it should first be determined *which* MP's come close to this trajectory:

Step 0 Using the optimized trajectory from mission scenario 1, determine which minor planets MP_{ok} come close to the spacecraft. "Close" will be interpreted as falling below the threshold

$$D_0(t) = 0.01 + 0.09\frac{r_a}{30\,\mathrm{AU}} \tag{11.6}$$

where r_a is the aphelion of the spacecraft's current trajectory. The time-dependent distances $\boldsymbol{D}(t)$ are to be computed with the methods outlined in section 11.4. It can be expected that a fully optimized trajectory resulting from mission scenario 1 will already be quite close to the final solution for mission scenario 2. Therefore, optimizing mission scenario 2 is best carried out by using a local optimization method, such as the Nelder-Mead algorithm. It is important to note that this (local) optimization should be carried out quite often, as the result from mission scenario 1 will generally be a complete Pareto front, which implies that

Table 11.1 List of default bounds on the transfer times to be used in the optimizations. The topmost value in each cell gives the lower bound, the bottommost value the upper bound. These values are based on fractions of either the associated Hohmann transfer orbit or the body's own orbital period. Cells without contents describe very unlikely scenarios, and have thus been omitted.

To \ **From**	Me	V	E	Ma	J	S	U
Me	15 88	15 151	21 211	34 341	171 1710		
Me (via Sun)	15 176	15 302	21 422	34 682	171 3420		
V	15 151	37 225	29 292	43 435			
V (via Sun)	15 302	37 450	29 584	43 870			
E	21 211	29 292	61 365	52 518			
E (via Sun)	21 422	29 584	61 730	52 1036			
Ma	34 341	43 435	52 518	114 687			
Ma (via Sun)	34 682	43 870	52 1036	114 1374			
J	171 1710	187 1866	200 1998	226 2256			
J (via Sun)	171 3420	187 3732	200 3996	226 4512			
S	407 4067	427 4274	445 4447	478 4781	735 7349		
S (via Sun)	407 8134	427 8548	445 8894	478 9131	735 9131		
U	1122 9131	1151 9131	1175 9131	1221 9131	1560 9131	1998 9131	
U (via Sun)	1122 9131	1151 9131	1175 9131	1221 9131	1560 9131	1998 9131	
SBS (via Sun)	2000 9131	2000 9131	2000 9131	2000 9131	2000 9131	2000 9131	2000 9131

many different initial guesses need to be tested.

The cost function to be used for the local optimization for mission scenario 2 method is then computed according to the following steps:

Step 1 Given some new estimate for the travel times t_f^i and the launch date t_0, perform all calculations required in mission scenario 1 described earlier in this section.

Step 2 For each resulting leg, calculate the time-dependent minimum distances $\boldsymbol{D}(t)$ from

the spacecraft to all asteroids MP_{ok} found in **Step 0**.

Step 3 The value of the cost function is

$$F_{\text{cost}} = \sum \Delta V + \sum \boldsymbol{D}(t), \tag{11.7}$$

where $\sum \Delta V$ follows from **Step 1**. Naturally, the values for ΔV and the sum of the distances $\boldsymbol{D}(t)$ must be of the same order of magnitude, so they must have the same scaling. To do so, it suffices to express all distances $\boldsymbol{D}(t)$ in AU.

It can be expected that it will not be possible (or at least very difficult) to visit *every* MP in the set MP_{ok}. Also, taking the sum of the distances will inherently have some bias towards minimizing only the largest distance. For this reason it is necessary to manually remove one or more of the MP's from MP_{ok} and retry to minimize the total distance. To facilitate this process an external record should be made that keeps track of the distances to each of the asteroids MP_{ok} at every step of the optimization. This external record can give a rough impression of how decreasing one distance influences the other distances (and the total ΔV required), thus aiding the identification of MP's that should be removed from the set at subsequent optimizations.

11.2.3 Mission Scenario 3

In this mission scenario, it is attempted to perform a flyby at some *pre-selected* MP's – the set $\text{MP}_{\text{selected}}$. Suppose this set consists of 20 preselected MP's. As it is rather difficult to find the exact sequence in which these 20 MP's should be flown-by, finding the most efficient sequence should be part of the optimization process itself. To best reflect all aspects of this particular scenario, it is best optimized by using MOO. As before, both the time-of-flight and the spacecraft's final mass should be optimized, but also the amount (and quality) of MP-flybys. Therefore, the first two cost functions for this mission scenario are exactly the same as in mission scenario 1.

The other cost function(s) should then reflect both the amount of MP's successfully flown by, and some measure of the quality of each of these flybys. Generally speaking, an MP-flyby becomes more valuable when the spacecraft's speed with respect to the MP is smaller. The relative speed at the flyby epoch can thus serve as a weighting function to be applied on the cost function itself. Unlike mission scenario 2, this mission scenario should be optimized with a *global* optimizer, since initial guesses are nearly impossible to find. Therefore, to have the most unbiased result, *each individual MP from the set* $MP_{selected}$ *must have its own associated cost function*. Thus, mission scenario 3 must be optimized using MOO, with $2 + 20 = 22$ separate objectives. The individual cost functions for each individual MP_i from the set $\text{MP}_{\text{selected}}$ should be computed as:

Step 0 Only compute this cost function for feasible (fully un-penalized) trajectories. If this is not the case, simply assign a dummy value

$$F_{\text{cost}} = 0 \tag{11.8}$$

and terminate the calculation.

Step 1 Compute the time-dependent minimum distance $D_{\min}(t)$ to MP_i for all separate legs in the trajectory. On the epoch of the absolute minimum distance thus found, also compute the relative velocity $\boldsymbol{V}_{\text{rel}} = \boldsymbol{V}_{\text{MP}} - \boldsymbol{V}_{S/C}$. Assign the function value

$$F_{\text{cost}} = -\begin{cases} \frac{\text{threshold}}{D_{\min}|\boldsymbol{V}_{\text{rel}}|} & \text{if } D_{\min} \leq \text{threshold} \\ \\ 0 & \text{otherwise} \end{cases} \tag{11.9}$$

with $D_{\min}(t)$ expressed in AU.

It would be an extremely painstaking affair to select the best solution from the 22-objective Pareto front that results from this optimization. Also, it is a considerable challenge to first select the "most interesting" MP's, since this concept is a matter of opinion and differs wildly from astronomer to astronomer, and subsequently prune the possible `seq`s which necessarily include cases with $m > 0$ and quite possibly, multiple DSM's per leg. Needless to say, this is *far too complicated* for the scope of this research. Although the idea behind mission scenario 3 is a logical extension of mission scenarios 1 and 2, the approach it requires as outlined above clearly justifies abandoning this scenario already at this stage. Therefore, this scenario will not be elaborated any further, and all focus will be shifted to the other 3 scenarios.

11.2.4 Mission Scenario 4

This scenario aims to optimize both the time of flight and the spacecraft dry mass m_{dry}, and also the *amount* and *quality* of MP-flybys. If successful, its solutions will give the guaranteed largest possible scientific output for a mission in the given framework, making it very worthwhile to attempt to find its globally optimal solution, despite the tremendous difficulties.

As the amount of flybys is unknown *a priori*, the amount and quality of the flybys can only be captured by an independent third objective function $F_{\text{cost}}^{(3)}$. Therefore, this scenario is optimized globally using a 3-objective MOO, where the first two objectives follow from the procedure followed in mission scenario 1. The cost function $F_{\text{cost}}^{(3)}$ can be computed as:

Step 0 Only compute this cost function for trajectories that are feasible (zero constraint violation). If this is not the case, simply assign a dummy value

$$F_{\text{cost}}^{(3)} = 1000 \tag{11.10}$$

and terminate the calculation.

Step 1 For feasible trajectories, calculate the minimum time-dependent distances $\boldsymbol{D}(t)$ from *all* legs in the resulting trajectory to *all* MP's in the (pre-pruned) data set.

Step 2 For the sub-set MP_{good} of MP's that result from this process, determine their relative velocities with respect to the spacecraft $\boldsymbol{V}_{\text{rel}}$ at the corresponding flyby epochs. Also

look up their type and orbital class to determine their corresponding scientific value $Q_{\rm sci}$ via Table 11.2. Then assign the third cost-function:

$$F_{\rm cost}^{(3)} = 1000 - \sum \frac{Q_{\rm sci}}{|\boldsymbol{V}_{\rm rel}|}. \tag{11.11}$$

Table 11.2 Simple scoring system to be used for mission scenario 4. These are the different types of MP's used in the MPCORB. The scores are intuited, based on the difficulty to reach them and/or their scientific value.

MP-type	**score ($Q_{\rm sci}$)**
Aten	2
Apollo	2
Object with $q < 1.381$ AU	2
Object with $q < 1.523$ AU	2
Object with $q < 1.665$ AU	2
Hilda	2
Jupiter Trojan	8
other Main Belt member	1
Centaur	10
Plutino	10
Other resonant TNO	10
Cubewano	10
Scattered disk object	20
Object is potentially hazardous asteroid	1000

The scores $Q_{\rm sci}$ used in **step 2** can be found in Table 11.2. This table is a copy of the list found in the MPCORB, with an intuited scoring system assigned to each type. The class of potentially hazardous asteroids (there is currently only one listed in the database; 99942 Apophis) should receive a score greatly exceeding that of all others to strongly encourage the optimization routine to include them in potential solutions. The scoring for the other classes is based on the difficulty of reaching them ("farther is higher"), and their level of exploration or value to the scientific community ("unexplored is higher").

Since **Step 1** in the calculation of this cost function will have *extreme* demands on the computation power, it is advisable to put considerable effort into making the (realtime) pruning methods discussed in section 11.4 as efficient as possible. Because this is by far the most difficult scenario to tackle, even more emphasis will need to be put on selecting sequences/paths that are likely to yield the best results.

11.2.5 Second Order Accuracy

Although the optimization of an MGA problem using the MPC provides good solutions, they are *first-order* solutions only. Due to the assumptions made in the MPC (impulsive ΔV manoeuvres, GAM's treated as instantaneous rotations of the Heliocentric velocity vector, etc.),

the accuracy of its solutions can certainly be improved upon. To do this, a more elaborate force model should be used. As discussed in section B.2.1 the optimum trajectory should be found by using the same force model as used during the generation of the DE405 ephemerides, that is, including the gravitational forces of the Sun, 8 planets, Pluto and the three largest MP's. Using this fairly complicated force model, the Lambert-targeting procedure can not be used anymore since it is based on a two-body assumption.

Although inaccurate, the assumptions made in the MPC only produce very modest errors compared to using the force model of DE405. The steps listed in section 11 should therefore remain largely unchanged for the second-order optimization. The only change should occur in the targeting step – this should replace the Lambert routine with a direct N-body integration of the spacecraft's position from one swingby body to the next. Naturally this introduces additional parameters to optimize – the perturbations due to small but non-central gravitational forces will perturb the spacecraft's final position slightly, causing it to *miss* its target by a small but non-negligible distance. Any optimization will have to bring this miss-distance down to *zero* in addition to optimizing the time-of-flight and ΔV. Therefore, the second-order cost function should at least incorporate the following changes:

- Instead of solving two-body Lambert problems between the swingby-bodies, the position of the spacecraft must be *integrated* using the full force model (Sun, 8 gravitating planets, Pluto, Ceres and Vesta).
- The positions of MP's should also be *integrated* using the full force model, in stead of recovered them from Kepler's equation (two-body assumption).

The perturbed Lambert problems have initial values

$$\boldsymbol{\chi}(t_0) = \begin{bmatrix} x(t_0) & y(t_0) & z(t_0) & \dot{x}(t_0) & \dot{y}(t_0) & \dot{z}(t_0) \end{bmatrix}^T \tag{11.12}$$

$$= \begin{bmatrix} x_0 & y_0 & z_0 & \dot{x}_0 & \dot{y}_0 & \dot{z}_0 \end{bmatrix}^T, \tag{11.13}$$

where x_0, y_0 and z_0 follow from the positions of the swingby bodies, which in turn follow from the assumed launch date t_0 and transfer times t_f^i between the swingby bodies. In addition to these times, the velocities $\dot{x}_0$, $\dot{y}_0$ and $\dot{z}_0$ are now also parameters to be optimized.

Since the first-order high-thrust MGA procedure is already quite accurate, it can be assumed that its solutions already lie well within the basin of attraction of the real optimum. Therefore, it is best to optimize the second-order accuracy problem with a local optimizer, such as the Nelder-Mead algorithm. To that end, the cost-function for the first-order procedure should be altered slightly:

Step 0 Execute **Step 0** and **Step 1** from the first-order cost function normally. Also assume (slightly) different values for the velocities $\dot{x}_0$, $\dot{y}_0$ and $\dot{z}_0$ to be used at launch, and each successive GAM.

Step 1 Integrate all trajectories using the full force model (DE405), from their initial positions, velocities and times to their final values. Also compute all minimum flyby

distances r_p^i and total ΔV_{tot}^i needed for each GAM i.

In contrast to the first-order algorithm, the integrations will generally put the spacecraft *close*, but not exactly *onto* the target body. The penalized cost function must reflect this via the miss-distance d_{miss} and total amount of ΔV_{tot}^i at each body i. To include both these factors, the following cost/penalty function will be employed:

$$F_{\text{cost}} = \sum \Delta V_{\text{tot}}^i + \exp\left(\sum \Delta V_{\text{tot}}^i \cdot \sum d_{\text{miss}}^i\right) - 1 \tag{11.14}$$

$$+ \begin{cases} r_p^i - r_{\text{min}}^i & \text{for all } i \text{ for which } r_p^i < r_{\text{min}}^i, \\ 0 & \text{otherwise} \end{cases} \tag{11.15}$$

where $d_{\text{miss}}^i = |\boldsymbol{r}^i_{\text{end, integrated}} - \boldsymbol{r}^i_{\text{body}}|$ is the miss-distance to the i^{th}-target body.

For mission scenarios 2 through 4, the time-dependent distances from each (now perturbed) leg to the (also perturbed) MP's must be determined. In second-order accuracy this is far more costly an operation, since each individual position for both the spacecraft and the MP in question must now be determined via a direct numerical integration of the respective equations of motion. Fortunately, the specific MP's which the spacecraft is to flyby have already been determined by the first-order algorithm. Also, the specific legs that will put the spacecraft close to a specific MP from this small subset are known in advance, so that the distances to only a handful of MP's must be calculated per leg. This justifies making the following adjustments for the calculation of the time-dependent minimum distances in second-order accuracy:

Step 0 Numerically integrate the positions of all candidate MP's from their respective initial positions to the selected launch date t_0.

Step 1 Integrate each leg of the new trajectories. Also integrate the positions of the candidate MP's for that leg. In these integrations, tweak the (adaptive) step size h_{n+1} used by the integrator based on the distance D_n to the MP at step n of the integration:

$$\hat{h}_{n+1} = \frac{h_{n+1}}{1 + 9e^{-3.5D_n}}. \tag{11.16}$$

This will ensure that as $D_n \to 0$, 10 times the amount of positions will be calculated for both the spacecraft and the asteroid than the integrator would normally do. The factor of 3.5 is to ensure that the step size only starts changing noticeably when the instantaneous distance is less than 2 AU.

Step 2 Aside from the above change, complete all integrations and calculations as in **Step 1** from the optimization without MP's.

Step 3 After these calculations, use the positions of the spacecraft and each MP that gave rise to the minimum relative distance D_{min} as initial values for a new integration, which

is to find the absolute minimum distance between the spacecraft and the MP in question. This can be a simple interval-halving scheme, as in

$$h_{n+1} = \begin{cases} \frac{-h_n}{2} & \text{if } D_n > D_{n-1} \\ h_n & \text{otherwise.} \end{cases} \tag{11.17}$$

Step 4 Add to the cost function the value of the following penalty function:

$$F_{\text{cost}} \leftarrow F_{\text{cost}} + \sum D^j_{\text{min}}, \tag{11.18}$$

for all j MP's from the given subset.

As discussed in appendix E.2.2, all integrations for the high-thrust optimizations will be carried out with a standard adaptive-step size Runge-Kutta-Nyström (RKN)8(6) integrator, which integrates the orbits via Encke's method to reduce the number of required steps.

11.3. Pruning MGA-problems

Because the computational cost involved with finding the global optimum for a specific MGA problem is rather high, it would certainly be beneficial to have some means of pruning the search space. One possible way to prune a given MGA problem is to use a generic *branch and bound* approach to pruning the MGA problem. As an example, consider a general, unconstrained, multi-objective MGA problem, of which the objective functions are given by

$$F(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) & f_2(\boldsymbol{x}) & \dots & f_N(\boldsymbol{x}) \end{bmatrix}^T \tag{11.19}$$

For the sake of clarity, each of the functions $f_i(\boldsymbol{x})$ are assumed to be given in penalized form, so that the problem's inherent constraints are included. Then, when it is assumed that each function $f_i(\boldsymbol{x})$ can be well-approximated by a quadratic model around some point $\hat{\boldsymbol{x}}$, it holds that

$$F(\hat{\boldsymbol{x}}) \approx \begin{bmatrix} \hat{\boldsymbol{x}}^T A_1 \hat{\boldsymbol{x}} + \boldsymbol{C}_1 \\ \hat{\boldsymbol{x}}^T A_2 \hat{\boldsymbol{x}} + \boldsymbol{C}_2 \\ \vdots \\ \hat{\boldsymbol{x}}^T A_N \hat{\boldsymbol{x}} + \boldsymbol{C}_N \end{bmatrix}, \tag{11.20}$$

where A_i is the design matrix corresponding to the objective function $f_i(\boldsymbol{x})$, and $\boldsymbol{C}_i$ are some constant vectors. To obtain a good approximation to the original objective functions on their entire domain, j such models need to be put up around several points $\hat{\boldsymbol{x}}_j$, gridded over the entire search space. When for each point $\hat{\boldsymbol{x}}_j$ the design matrices A^j_i and constant vectors $\boldsymbol{C}^j_i$ would be known, the corresponding minimum value (within the grid-point's range) could easily be determined entirely analytically.

With such grid-wise defined quadratic functions, it would be a trivial task to find those points whose associated quadratic model suggests that that particular region does not need

further investigation. For those regions that are found to be the most promising, the process can be repeated with a finer grid-spacing, but then on a search space reduced in size by several orders of magnitude. There is however a problem inherent in this method – the determination of the design parameters A_i^j and C_i^j. For each objective function $f_i(\boldsymbol{x})$ in a problem with $\mho$ swingby's,

$$\begin{aligned} f_i(\boldsymbol{x}) &= \boldsymbol{x}^T A_i \boldsymbol{x} + \boldsymbol{C}_i && (11.21)\\ &= \boldsymbol{C}_i + \sum_{m=1}^{\mho}\sum_{n=1}^{\mho} A_{mn}^{(i)} \boldsymbol{x}_m \boldsymbol{x}_n && (11.22)\\ \rightarrow \frac{\partial f}{\partial x_j} &= \sum_{k=1}^{\mho} \left(A_{kj}^{(i)} + A_{jk}^{(i)} \right) \boldsymbol{x}_j && (11.23)\\ \rightarrow \nabla f(\boldsymbol{x}) &= \left(A_i + A_i^T \right) \boldsymbol{x}. && (11.24) \end{aligned}$$

Thus, knowing the upper triangular half of the matrix $\left(A_i + A_i^T \right)$ is enough to solve for $\boldsymbol{C}_i$ and thus to solve the i^{th}sub-problem. This matrix can be obtained by computing the gradient $\nabla f(\boldsymbol{x}_j)$ at some point $\boldsymbol{x}_j$. Therefore, to calculate the design matrix A_i^j and the constant vector $\boldsymbol{C}_i^j$ for all objective functions $f_i(\boldsymbol{x}_j)$ at all grid points j will require

$$M \cdot (4\mho + 1) \cdot \prod_{q=1}^{\mho} \frac{\left(\boldsymbol{x}_{UB} - \boldsymbol{x}_{LB} \right)_q}{I_q}, \tag{11.25}$$

where M is the amount of objectives, $\boldsymbol{x}_{UB_q}$ and $\boldsymbol{x}_{LB_q}$ are the upper and lower bounds on $\boldsymbol{x}_j$ in dimension q, and I_q is the grid spacing in dimension q [1]. As a numerical example, consider a MGA problem with 4 GAM's and 2 objectives, with I_q = constant = 10 days, $\boldsymbol{x}_{UB_q}$ = constant = 400 days, and $\boldsymbol{x}_{LB_q}$ = constant = 100 days. Then the required number of FE's to use this naive branch-and-bound method, is

$$\begin{aligned} FE &= 2 \cdot (4 \cdot 4 + 1) \cdot \prod_{q=1}^{\mho} \frac{(400 - 100)\ \text{days}}{10\,\text{days}} \\ &= 2 \cdot 17 \cdot 30^4 && (11.26)\\ &= 27,540,000 \end{aligned}$$

which almost certainly *in*creases the computational cost, instead of *de*creasing it. The only reasonable way to reduce this number, is by assuming a larger value for I_q. Although for the outer planets this would be reasonable (although generally, the limits $\boldsymbol{x}_{LB}$ and $\boldsymbol{x}_{UB}$ are much larger for the outer planets), for Mercury (with orbital period ∼80 days) increasing I_q certainly does *not* lead to sufficiently accurate models to correctly predict the exact topology of the objective function. Therefore, this method is not suited to prune MGA problems.

A different method, called GASP (or LT/GASP for its low-thrust counterpart), is elaborated

[1] Note that the factor $(4\mho + 1)$ comes from the assumption of using a fourth-order method to compute the gradients (see appendix E.2). If forward/backward differences are used, this factor will be reduced to $(\mho + 1)$

in [Izzo et al., 2007] and [Vasile et al., 2006]. This method does not make the aforementioned complete approximation, but determines whether a small amount of points $\boldsymbol{x}_j$, equally spaced along the search space, can satisfy the problem's constraints. A point and its associated neighborhood (the surrounding grid) is marked for pruning when some constraint somewhere along the corresponding calculation is violated. As such, the LT/GASP method requires that a single grid point x_j returns a *unique* solution. This implies that in order to use LT/GASP, the general MGA-problem that it is to prune can only use *either* the short-way or long-way solutions ($\Delta\theta \leq \pi$) and *zero* complete revolutions ($m = 0$)[2].

These demands in themselves already reduce the search space by several orders of magnitude. However, as previously discussed, using only the short-way solutions might be too restrictive for the problem at hand. Moreover, for mission scenario 4 it might be desirable to use multiple revolutions before or after a swingby in the inner Solar system, to try and maximize the number of MP's encountered in the MAB. Also, further analysis of the LT/GASP pruning method shows that the computational complexity is $O(\mho^4) + O(\mho^2)$ for high-thrust propulsion systems, and $O(\mho^5)$ for low-thrust propulsion systems. In conclusion, preliminary pruning carried out with this method can indeed reduce the search space of simple problems ($\mho \sim 2$) by orders of magnitude, in a small amount of time. For more GAM's it becomes less and less useful compared to directly solving the problem with a generic global optimizer. Also, although the (correct) implementation of LT/GASP is not difficult, for the purpose of this research it will most likely consume more time than it will save. For these reasons the LT/GASP method will not be implemented in Skipping Stone, but will be left here as a recommendation for future versions.

11.4. Pruning Minor Planets in Costfunctions

It can very well be expected that a rather large fraction of the total computational cost for optimizing mission scenarios 2 and 4 must be devoted to testing the feasibility of MP flyby's, simply because so many MP's must be tested. For these optimizations it is imperative that the amount of MP's to be included in the calculation be reduced as far as can be reasonably justified.

When the spacecraft is equipped with a high-thrust propulsion system, the shape of its trajectories generally are (parts of) conic sections. The mathematics involved with such conic sections is simple enough to allow some quick and efficient pruning methods. The methods mentioned here are largely based on [Hoots et al., 1984], a work on minimizing the risk of collisions with space debris of Earth orbiting spacecraft.

11.4.1 Based on Apses

The simplest and quickest method to eliminate candidate minor planets, is by looking at their aphelia and perihelia. If an MP's aphelion always lies closer to the Sun than the spacecraft's perihelion of some intermediate elliptic orbit, the spacecraft will not ever be able

[2]Trajectories for which $m > 0$ generally have 2 solutions to the Lambert problem; see section C.1.2

to reach it. Conversely, if the MP's perihelion lies farther than the spacecraft's aphelion, the same conclusion follows. Such minor planets may then be pruned from the data set. Mathematically,

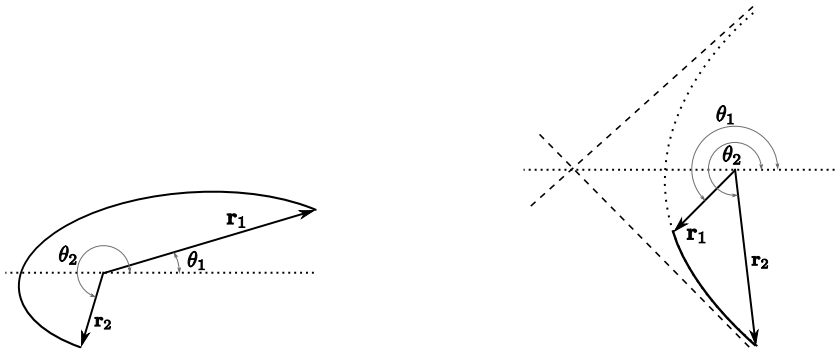
$$\text{Prune MP when } r_p > r_{\max} \text{ or } r_a < r_{\min}, \tag{11.27}$$

where $r_{\max}$ is the spacecraft's aphelion, and $r_{\min}$ its perihelion.

There are a few extra issues that need to be considered, which are not discussed in [Hoots et al., 1984]. The amount of minor planets within reach may be further lowered by looking at the turn angle $\Delta\theta$ that arises in the MGA problem for a specific leg from one planet to the next (see section 9.3). Referring to Figure 11.1(a), the minimum and maximum radii are to be taken equal to

$$\begin{aligned}
&\text{if } \theta_1\theta_2 < 0 \text{ and } |\theta_1| + |\theta_2| = \Delta\theta \quad \rightarrow \quad \begin{cases} r_{\min} = a(1-e) \\ r_{\max} = \max(r_1,\, r_2) \end{cases} \\
&\text{if } \theta_1\theta_2 < 0 \text{ but } |\theta_1| + |\theta_2| \neq \Delta\theta \quad \rightarrow \quad \begin{cases} r_{\min} = \min(r_1,\, r_2) \\ r_{\max} = a(1+e) \end{cases} \\
&\text{if } \theta_1\theta_2 > 0 \text{ and } |\theta_1 - \theta_2| = \Delta\theta \quad \rightarrow \quad \begin{cases} r_{\min} = \min(r_1,\, r_2) \\ r_{\max} = \max(r_1,\, r_2) \end{cases} \\
&\text{if } \theta_1\theta_2 > 0 \text{ but } |\theta_1 - \theta_2| \neq \Delta\theta \quad \rightarrow \quad \begin{cases} r_{\min} = a(1-e) \\ r_{\max} = a(1+e). \end{cases}
\end{aligned} \tag{11.28}$$

where it is assumed that $-\pi < \theta < \pi$. If the number of full revolutions $m > 0$, Equation 11.28 simply degenerates into the last case, where $r_{\min}$ and $r_{\max}$ are simply equal to the apses. Of course, Equation 11.28 also holds for both pro- and retrograde orbits, provided the angles θ_1 and θ_2 are defined as positive in the direction of motion.



(a) Elliptic orbit (b) Hyperbolic trajectory

Figure 11.1 Definition of the angles θ_1 and θ_2 used in the construction of the pruning methods on parts of conic sections.

Also, if the spacecraft's trajectory is hyperbolic ($e > 1$), its aphelion $r_{\max}$ is not defined. In those cases, $r_{\max}$ and $r_{\min}$ depend on the sign of the product of the angles θ_1 and θ_2 (see

Figure 11.1(b)):

$$\text{if } \theta_1\theta_2 > 0 \rightarrow \begin{cases} r_{\min} = \min(r_1,\, r_2) \\ r_{\max} = \max(r_1,\, r_2) \end{cases} \qquad \text{if } \theta_1\theta_2 < 0 \rightarrow \begin{cases} r_{\min} = a(1-e) \\ r_{\max} = \max(r_1,\, r_2)\,. \end{cases} \tag{11.29}$$

Based on Minimum Distance Between the Conic Sections

The apses-check discussed previously does not take into account the actual location of the apses; it may very well be that these apses are located on opposite sides of the Solar system. Therefore, it can still be true that the actual minimum distance between the two conic sections is always too large to perform a flyby. To determine if that is indeed the case for a given transfer section and MP, the minimum distance between the two conic sections needs to be determined. The general geometry of this problem is shown in Figure 11.2(a).

The problem of finding the minimum distances between to confocal ellipses is surprisingly complicated – there is no analytical way to locate the location of those vertices on both ellipses that minimize the distance between them *generally*.

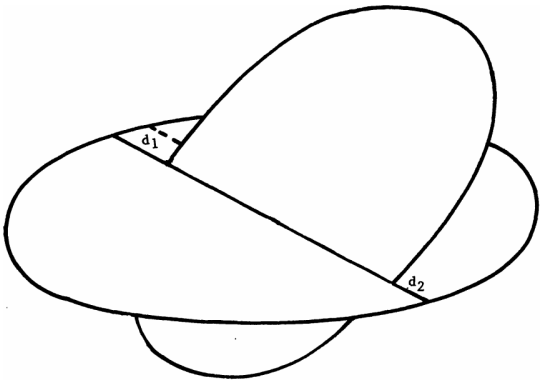
The function

$$\boldsymbol{r}_{\text{rel}}^2 = \boldsymbol{r}_p \cdot \boldsymbol{r}_p + \boldsymbol{r}_s \cdot \boldsymbol{r}_s - 2|\boldsymbol{r}_p||\boldsymbol{r}_s| \cos\gamma \tag{11.30}$$

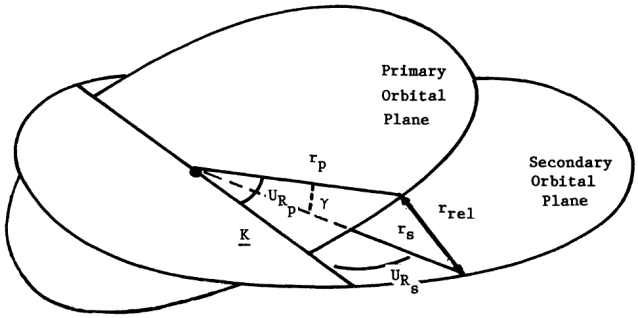
gives the distance between any two vertices $\boldsymbol{r}_p$ and $\boldsymbol{r}_s$ on the ellipses p (primary) and s (secondary) (see Figure 11.2(b)). It can be shown that this function can have as much as 16 stationary points (see [Gronchi, 2005]), of which at most 2 are indeed minima. The objective is thus to find the smallest of these two minima.

[Hoots et al., 1984] uses Newton's method for bivariate functions to find the roots of the *gradient* of the bivariate function associated with Equation 11.30. However, after only a few numerical experiments with this (rather complicated) method, it was found that it is quite unstable. Quite frequently, using the initial values provided by Hoots et al. [1984], the method started oscillating between two values quite widely spaced ($\sim 1 \times 10^{-2}$ radians) around the true minimum. Also, sometimes, the method converged to a local *maximum*, even with initial values quite close to the true minimum. Although the first problem could be remedied by taking the average value of both values after some fixed maximum number of iterations, the last problem is much more severe. Remedying this second problem would require a *second-derivatives test* for each of the converged values, and if that fails, a complete restart of the algorithm at a different initial value. This would dramatically increase the already quite high number of required iterations (~ 40 was not uncommon), as well as increase the computational cost per iteration. In conclusion, Newton's method applied to this problem is simply too costly for pruning purposes, so another method must be used.

The majority of work on this particular problem has been performed by Gronchi [2005, and references therein]. However, his principal method is very general in nature – it tries to find *all* 16 stationary points (i.e., all maxima, minima and saddle points) simultaneously. Because



(a) Simple overview. d_1 and d_2 are the minimum distances between the two ellipses. The objective is to find the smallest of these two values.



(b) Some definitions for minimizing the length of $\boldsymbol{r}_{\mathrm{rel}}$. Note that the relative angle γ is not simply the difference between the two inclinations.

Figure 11.2 Some definitions for the problem of finding the smallest distance between two ellipses. Adopted from [Hoots et al., 1984, Figures 2 and 4].

of this generality, it is unnecessarily complicated. It involves calculating the determinants of no less than 17 Sylvester-matrices of dimension 6×6 in each iteration, generally requiring ~ 10 iterations to converge. Therefore, also this method is far too costly to be used as a pruning method.

The fastest and easiest method that solves this problem, is outlined in [Kim, 2006]. Although this method lacks the mathematical rigor (and notational beauty) found in [Gronchi, 2005], the method was shown to work quite well. His method is based on repeatedly calculating the minimum distance from *one fixed point* on the first ellipse to the other ellipse, and the reverse:

Step 0 (re)define both ellipses α and β in their parametric representation (section A.1.5). This allows one to define the corresponding rotation matrices

$$R_\alpha = [\boldsymbol{u} \quad \boldsymbol{v} \quad \boldsymbol{u} \times \boldsymbol{v}]_\alpha \,, \tag{11.31}$$

$$R_\beta = [\boldsymbol{u} \quad \boldsymbol{v} \quad \boldsymbol{u} \times \boldsymbol{v}]_\beta \,, \tag{11.32}$$

so that the ellipses can be written in the standard form

$$E \equiv R\,[a\cos t,\ b\sin t, 0] + \boldsymbol{c} \tag{11.33}$$

Also pre-compute the inverse of these matrices, R_α^{-1} and R_β^{-1}, and set $k = 0$.

Step 1 Compute

$$\begin{aligned} \omega_{\alpha,\,k} &= [a_\alpha \cos t_{\alpha,\,k},\ b_\alpha \sin t_{\alpha,\,k}, 0]^T & (11.34)\\ \tilde{\omega}_{\alpha,\,k} &= R_\alpha \omega_{\alpha,\,k} + \boldsymbol{c}_\alpha, & (11.35) \end{aligned}$$

where $t_{\alpha,\,k}$ is the current estimate for E_α.

Step 2 The distance between point $\tilde{\omega}_{\alpha,\,k}$ and ellipse E_β can be found from the equation

$$Q(\boldsymbol{s},\ E_\beta,\ t_\beta) = (s_x - a_\beta \cos t_\beta)^2 + (s_y - b_\beta \sin t_\beta)^2 \tag{11.36}$$

in which $\boldsymbol{s} = R_\beta^{-1}\,(R_\alpha \omega_{\alpha,\,k} + (\boldsymbol{c}_\alpha - \boldsymbol{c}_\beta))$. This equation can be solved by first solving the quartic equation

$$v^4 + 2\left(\frac{A}{C}\right)v^3 + \left(\frac{A^2 + B^2 - C^2}{C^2}\right)v^2 - 2\left(\frac{A}{C}\right)v - \left(\frac{A}{C}\right)^2 = 0, \tag{11.37}$$

in which

$$\begin{aligned} v &\equiv \cos t_\beta,\\ A &= a_\beta s_x,\\ B &= b_\beta s_y,\\ C &= b_\beta^2 - a_\beta^2. \end{aligned}$$

The real valued solutions $-1 \le v_j \le 1$ each give *two* possible values for $\hat{t}_\beta$,

$$\hat{t}_{\beta,\,12} = \pm \arccos v_j,$$

of which the minimum distance follows from

$$\hat{t}_\beta \leftarrow \min Q(\boldsymbol{s},\ E_\beta,\ \hat{t}_\beta) = \min\left[\left(s_x - a_\beta \cos \hat{t}_{\beta,\,12}\right)^2 + \left(s_y - b_\beta \sin \hat{t}_{\beta,\,12}\right)^2\right] \tag{11.38}$$

Step 3 Repeat **Step 2**, but this time with the initial points $\tilde{\omega}_{\alpha,\,0}$ and $\tilde{\omega}_{\alpha,\,0}$ that follow from $\hat{t}_\beta$. Compute the minimum distance from this point to ellipse α. The result of this step is the new parameter $t_{\alpha,\,k}$.

Step 4 Set $k \leftarrow k+1$. Repeat **Steps 1–3** as long as the distance function

$$U = \|\tilde{\omega}_\alpha - \tilde{\omega}_\beta\|_\infty \tag{11.39}$$

decreases, that is

$$U_{k+1} < U_k.$$

Note that this process involves finding the roots of two quartic equations for a single iteration (Equation 11.37), which can be done analytically[3]. After some numerical experimentation, it was found that this method always converges to the minimum near the initial value, in less than 8 iterations. Note that in case one of the trajectories is hyperbolic (i.e., that of the spacecraft), the computation should be split in two parts – the first half computes the distance from a point to an ellipse, the second half from a point to a hyperbola. This can be accomplished by using the corresponding hyperbolic counterpart of the parametric representation in Equation 11.33 (see section A.1.5), and all subsequent steps:

$$H \equiv R\left[a\cosh t,\ b\sinh t, 0\right] + \boldsymbol{c}. \tag{11.40}$$

Initial values are again provided by Hoots et al. [1984, Equation 18], which are based on the true anomalies which minimize the distance between two corresponding *circular* orbits. They are

$$\theta_p = \Delta_p - \omega_p, \tag{11.41}$$
$$\theta_s = \Delta_s - \omega_s, \tag{11.42}$$

where

$$\cos\Delta_p = \frac{\sin i_p \cos i_s - \sin i_s \cos i_p \cos\left(\Omega_p - \Omega_s\right)}{\sin i_R}, \tag{11.43}$$
$$\sin\Delta_p = \frac{\sin i_s \sin\left(\Omega_p - \Omega_s\right)}{\sin i_R}, \tag{11.44}$$
$$\cos\Delta_s = \frac{\sin i_p \cos i_s \cos\left(\Omega_p - \Omega_s\right) - \sin i_s \cos i_p}{\sin i_R}, \tag{11.45}$$
$$\sin\Delta_s = \frac{\sin i_p \sin\left(\Omega_p - \Omega_s\right)}{\sin i_R}, \tag{11.46}$$
$$\tag{11.47}$$

in which the usual notation for orbital elements has been used. Note that the value of i_R, the relative inclination between the two orbits, is not required to compute either Δ_p or Δ_s.

[3]Actually, solving this quartic by finding the *eigenvalues of its companion matrix* is numerically more efficient. This is not so surprising, since the analytical solution to a quartic is quite long and messy, and actually requires more operations than this numerical method. It should be noted though that this method is slightly less accurate than the analytic solution.

With the minimum distance now known, it can be compared against some predefined threshold value D to see if the orbits' mutual geometry can bring the spacecraft close enough to the MP. This threshold value is critical to determine whether an MP should be analyzed further. It seems reasonable to assume that if the perihelion of the spacecraft's trajectory is relatively large, the spacecraft is further away from the Sun making the corresponding length of the leg also quite large. This allows much more time to adjust the spacecraft's trajectory than if the opposite were true. Therefore, the value for the threshold distance will vary according to the aphelion of the spacecraft's trajectory (see also Equation 11.6):

$$D = 0.01 + 0.09\frac{r_a}{30\text{AU}}(\text{AU}). \tag{11.48}$$

Note that this particular function is designed such that the threshold distance D has a minimum value of $\sim$0.01 AU (close to the Sun), and a value of 0.1 AU when the spacecraft is roughly as far as Neptune ($r_a \approx 30$ AU).

Minimum Distance Between a Minor Planet and Spacecraft

The previous section was based entirely on geometry, but of course *time* is also a factor. Even when the spacecraft's trajectory and the MP's orbit lie close together in a geometric sense, it still could be the case that the MP is nowhere near its minimum-distance vertex at the same time the spacecraft is just passing through it. The minimum distance **as a function of time** must again be found numerically. Hoots et al. [1984] discusses how to solve the time-dependent problem efficiently.

The problem is to find those times t when $d(t) \leq d_{\text{req}} = D$, with $d(t)$ the distance as a function of time and D some prescribed maximum distance threshold. As can be seen in Figure 11.3, this requirement is only valid inside an angular window of width $2U_R$ in both orbits p and s. [Hoots et al., 1984] derived an expression for $\cos U_R$:

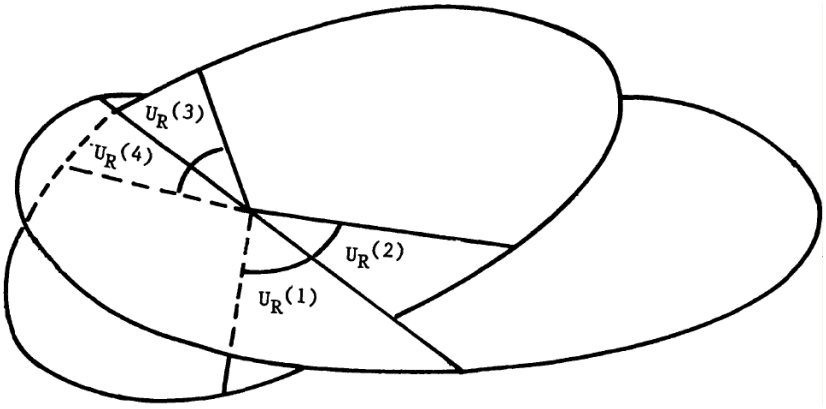


Figure 11.3 Angular windows to find the times when the distance between the MP and the spacecraft is smaller than some predefined value. Adopted from [Hoots et al., 1984, Figure 5].

$$\cos U_R = \frac{-D^2 e\cos(\omega-\Delta) \pm (p - De\sin(\omega-\Delta))\sqrt{Q}}{p\,(p - 2De\sin(\omega-\Delta)) + D^2e^2}, \tag{11.49}$$

where

$$Q = p\,(p - 2De\sin(\omega - \Delta)) - pD^2/a, \tag{11.50}$$

in which $p = a(1-e^2)$ is the usual semi-latus rectum, and Δ is given by either Equations 11.43 and 11.44 or Equations 11.45 and 11.46. Equation 11.49 holds for both orbits, so all quantities should be read with the appropriate subscripts. The $\pm$ sign in combination with the two possible outcomes of the arccos-function that determines U_R, define four possible outcomes for U_R. These in turn define the two angular windows $\left[U_R^{(1)}, U_R^{(2)}\right]$ and $\left[U_R^{(3)}, U_R^{(4)}\right]$, where the distance between the two orbits is smaller than the predefined distance D.

The four angular windows (two for each orbit) can be converted into time windows by solving the four corresponding Kepler equations. These time windows may then be projected forward or backward in time by simply adding or subtracting multiples of the period of the corresponding orbit. The initial values are then the centers of the overlap periods when the MP and spacecraft are within the same angular window at the same time. Note that if the first occurrence of such an overlap lies outside the interval of interest (usually, the length of time the spacecraft requires to cover its current leg), the MP can immediately be discarded without any further calculations. Note also that in case one of the trajectories is hyperbolic (e.g., that of the spacecraft), it suffices to use the hyperbolic counterpart of Kepler's equation.

To find the actual time-dependent minimum distance, Hoots et al. [1984] suggest using a Newton-Raphson method to find the roots of the time rate of change of Equation 11.30,

$$\dot{r}_{\text{rel}} = \boldsymbol{r}_p \cdot \dot{\boldsymbol{r}}_p + \boldsymbol{r}_s \cdot \dot{\boldsymbol{r}}_s - \boldsymbol{r}_p \cdot \dot{\boldsymbol{r}}_s - \boldsymbol{r}_s \cdot \dot{\boldsymbol{r}}_p. \tag{11.51}$$

However, the Newton-Raphson method requires also evaluations of the *second* derivative, which requires the explicit evaluation of $|\boldsymbol{r}_p|$ and $|\boldsymbol{r}_s|$ (to obtain the values for $\ddot{\boldsymbol{r}}_p$ and $\ddot{\boldsymbol{r}}_s$), and 7 additional scalar products. It was found by numerical experimentation that the cost of these additional calculations outweigh the benefits of the quadratic convergence of a Newton-Raphson method, when compared to using the Chebyshev root-finding method. Although this method is somewhat less accurate, the objective of this whole process is to check if the MP can be discarded or not; very high accuracy is not required for that. A few digits are more than sufficient.

A drawback of this method is that the times calculated from the overlap have no way of taking into account the phase angle. That is, even if two *time* windows are found to overlap, it might still be the case that the corresponding *space* windows are on opposite sides, so the spacecraft and the MP are on opposite sides of their trajectory. So to further speed up calculations, the distance at each mid-point of each interval is calculated prior to finding the actual root (see Figure 11.4). If this distance is found to be larger than 25 times the threshold distance D, this "opposite sides" scenario is likely to apply and the calculation of the actual root is skipped.

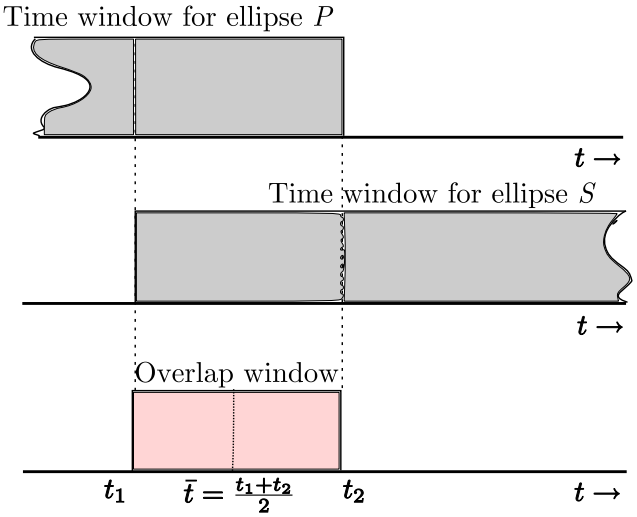


Figure 11.4 Evaluating the distance at each mid-point provides an opportunity to further cut-down the computational budget. If the distance at each mid-point is more than 25 times the threshold distance, the MP and the spacecraft are likely to be on opposite sides in their trajectory and the actual root does not have to be calculated.

Coplanar case

The value for Q in Equation 11.50 can sometimes be negative, *or* the absolute value of the right-hand side in Equation 11.49 can be greater than 1. It can be shown that these conditions imply that the geometric distance between the two orbits never exceeds the threshold distance D – the orbits lie in almost the same plane. [Hoots et al., 1984] handle such cases by simply "stepping through" the distance function (minus D) until it changes sign. The main focus of the [Hoots et al., 1984] paper was on *space debris* which generally has wildly different orbital planes, so that [Hoots et al., 1984] encountered this case only in ∼5% of their trials. However, this case applies far more often in the framework of the current mission, because the MP's are *all* more or less coplanar. Indeed, preliminary testing indicated that about 25% of the MP's needed to be treated with this method. Considering the fact that pruning MP's inside cost functions as described in this section must be executed *extremely quickly* for mission scenario 4; this scenario uses the number of reachable MP's as one of its objective functions, so that for a single optimization, the ∼ 90.000 MP's need to be processed with all methods described in this section for *thousands* of times. The only hope to do such an optimization in any reasonable amount of time is to make this entire routine run *as fast as possible.*

Simplistically "stepping through" the distance function has the major drawback that it is extremely slow compared to the methods used for non-coplanar cases. The ∼25% mentioned is therefore unacceptable, so a faster method has to be devised. Using the fact that the average eccentricity of all MP's in the pre-pruned set is about $e_{\text{avg}} \approx 0.14$, it seems reasonable to simplify the coplanar-cases by assuming these MP's are in a *circular* orbit. With that simplification it is fairly easy to come up with estimates of where the closest approaches are likely to occur; where the spacecraft's trajectory "intersects" the MP's circular orbit:

$$\begin{aligned} r_{S/C} &= \frac{a(1-e^2)}{1-e\cos\theta} = R_{\text{MP}} \\ \Rightarrow \theta &= \pm \arccos\left(\frac{1}{e}\left(\frac{a(1-e^2)}{R_{\text{MP}}} - 1\right)\right), \end{aligned} \tag{11.52}$$

which can again be converted to time via Kepler's equation. The true anomaly for the MP is calculated by progressing its true anomaly at the initial time t_0 forward by the same time step. Taking a value for $R = (r_a + r_p)/2$ (i.e., the average between its aphelion and its perihelion), and taking the width of the time window to be 1/5 of the maximum of the two periods, was found to give the best results. The assumption of a circular orbit was found to be quite efficient when $e_{\text{MP}} \lesssim 0.25$; MP's with higher eccentricities started to require more corrective measures than seemed justifiable, and moreover, such cases only constitute about 3% of the pre-pruned set. Therefore, cases for which $e_{\text{MP}} > 0.25$ are simply "stepped-through", as described in [Hoots et al., 1984].

Pruning for Parts of Conic Sections

In the MGA framework, after performing a GAM the spacecraft will only cover *part* of a conic section in the corresponding leg (except for cases where $m > 0$). In those cases, the computational cost for the pruning procedures outlined above can be reduced even further; the apses-check can be applied unaltered, but the geometric minimum distance might very well occur outside the region that is actually covered by the spacecraft for many cases. This implies the calculation of the second minimum distance between the conic sections (e.g., re-starting the calculation π rad further along the orbit) can be skipped, if the corresponding initial value falls well outside the angle covered in the conic section. That is,

$$\text{skip second calculation if } \theta_2 < \hat{\theta}_{p,0} + \pi < \theta_1, \tag{11.53}$$

with θ_1 and θ_2 as in Figure 11.1(a). Conversely, the calculation should be started at the *second* initial value if the above is true for the *first* initial value: (Equation 11.42):

$$\text{skip first calculation if } \theta_2 < \theta_{p,0} < \theta_1. \tag{11.54}$$

Obviously, for cases where the number of complete revolutions is nonzero ($m > 0$), the procedure should be left unaltered.

A similar reasoning holds for the calculation of the time-dependent distance; only *that* angular window that lies *entirely* in the region covered by the spacecraft should be selected as the initial value for the calculation. That is, select only that window $\left[U_R^{(1)}, U_R^{(2)}\right]$ or $\left[U_R^{(3)}, U_R^{(4)}\right]$ that falls entirely inside the interval $\theta_1 < \theta < \theta_2$. Again, both windows should be used in case $m > 0$. Note that for many cases, these relatively simple modifications can cut the computation time roughly in half; a fact that will most certainly prove beneficial later on.

Part IV

Low-Thrust Mission

12

The Low-thrust MGA-problem, first order

When examining Tsiolkovskii's equation more closely it can readily be concluded that the maximum possible speed change is directly proportional to the exhaust speed of the gasses from the engine. This points to an inherent weakness in high-thrust propulsion systems; no chemical reaction known produces enough energy to allow the heated reaction products to expand or move faster than about 6 km/s. This fact limits the theoretically attainable capacity of high-thrust engines. To still try and increase the efficiency of space propulsion systems, new types of engines have been developed that try to remedy this limitation. Instead of relying on chemical reactions for their energy, such *low-thrust* systems use electromagnetic forces to accelerate charged mass out of the engine. This principle poses practically no limits on the attainable exhaust speed, making low-thrust systems substantially more potent compared to high-thrust systems.

This opens up many new windows of opportunity in mission design – with these engines, many targets that were hard or impossible to reach with high-thrust propulsion suddenly become accessible. However, their increased efficiency comes at a price – as the name implies, the actual *thrust* generated by such engines is very small; usually in the order of 1/10 N or less. The power of low-thrust systems lies in the fact that their minuscule force can be exerted uninterruptedly for months at a time, but the low actual thrust level makes designing interplanetary trajectories for spacecraft relying on such an engine much more difficult.

Although low-thrust engines had been used already since the cold war, it was only in the last decade that such engines were perfected to the level where they could be applied in interplanetary missions. As such, low-thrust interplanetary trajectory optimization is a relatively young field and is consequently developing quite rapidly. New methods and techniques appear in print quite frequently, each new one better than the last. One of these methods, the *exponential sinusoid*, has proven to be an adequate method to be used in first-order optimizations. This chapter is therefore focussed around this method – its basics, advantages and pitfalls, how to use it for interplanetary missions and several implementation details.

All these aspects are treated in sections 12.2 through 12.5. First however, a more complete background of low-thrust propulsion systems is necessary, which is the subject of section 12.1.

12.1. Low-thrust Systems

High-thrust engines use chemical reactions as their source of energy. The energy that can be released in chemical reactions is limited by the energy stored in the specific chemicals used, but even the most potent mix known can store relatively little energy per unit mass. This fact puts limits on the theoretically achievable speed change deliverable by a high-thrust engine, as can be concluded from Tsiolkovskii's equation.

Increasing an engine's performance beyond the capability of current engines can be done in two ways. A close examination of Tsiolkovskii's equation reveals that increasing the mass-ratio Λ or increasing the exhaust velocity c_{eff} will do the job. Increasing the mass ratio Λ would however be a naive approach; it basically means using a larger rocket that has less structural mass (i.e., "lighter rockets"). The better way is increasing the exhaust velocity c_{eff}, and low-thrust systems do exactly that. They avoid the inherent limits of chemical propellants by relying on electromagnetic forces instead.

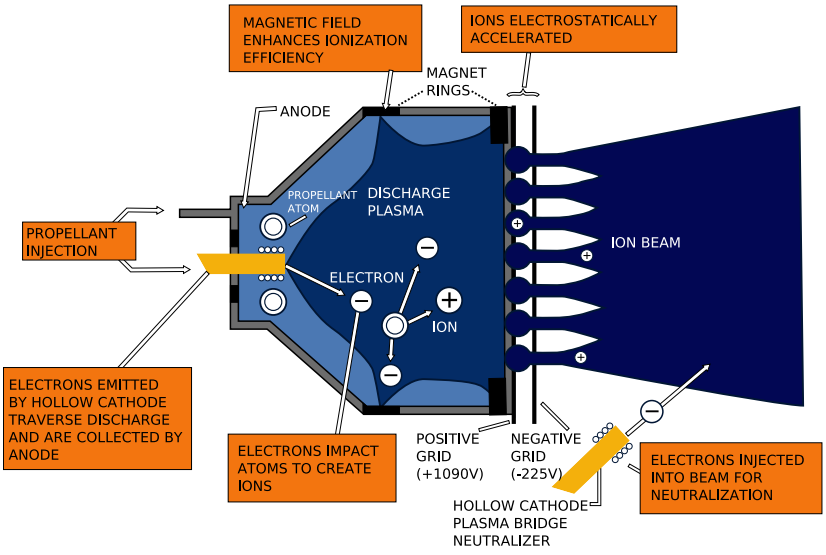


Figure 12.1 A basic Ion engine (Kaufman-type, schematically). From [Wikipedia.org, 2009a].

Figure 12.1 shows a schematic of a basic ion engine. In most types (gridded types, used for interplanetary missions) a relatively heavy, unreactive gas (usually Xenon (Xe)) is used as propellant. First the engine ionizes the propellant gas using electron bombardment. This process creates a short-lived plasma consisting of Xe-ions and electrons. A series of negatively charged grids attract the Xe-ions and repels the electrons, thus splitting the plasma into its charged constituents. The Xe-ions accelerate towards the grid and are ejected with high speed on the other side[1]. After the ions pass the last grid, they are ejected into space with extremely high speeds. Exhaust speeds of up to 70 km/s have been accomplished in

[1] In engines that use multiple grids, this process is repeated several times

space missions, and speeds of up to 250 km/s have been demonstrated in the laboratory.

There are some problems with these gridded-types of engines. Naturally it is very difficult to guide the ions *exactly* through the holes in the grids. Many ions will inevitably collide with one of the grids on their way to the nozzle. As the intention of the engine is to make these ions move as fast as possible, these collisions are highly energetic, causing these grids to degrade quite rapidly. Another problem is charge distribution – some ions will collide with the engine's interior, causing the engine to accumulate positive charge. Most engines remedy this problem letting a small fraction of the electrons from the plasma collide intentionally with the engine's interior. The rest of the electrons are injected directly into the exhaust plume to neutralize both the exhaust gas and the engine. However, it is quite difficult to find the correct fractions to use in order to accomplish a completely neutral engine, and most ion engines will indeed accumulate at least *some* charge. This might be a problem for on-board electronics, but the largest problem will be the perturbative Lorentz-force this charge implies when the spacecraft flies through an intense magnetic field (that of Jupiter, for instance). However, these problems are relatively minor compared to the benefits of an ion engine. As mentioned in section 9.1, chemical thrusters generally have $I_{\text{sp}} \approx 300$ s, while some ion engines can accomplish $I_{\text{sp}} \approx 4000$ s; an improvement by a factor of almost 15. Some experimental types have even demonstrated I_{sp}s of > 20.000 s.

An ion engine's thrust level is directly proportional to the power it receives. That is, the more power is available, the higher the thrust level that the engine can produce. Typically, ion engines do have a *maximum thrust level* $T_{\max}$, no matter how much power is available. Also, they require a minimum power level to operate, and have a minimum thrust $T_{\min}$ associated with that power level. For all intermediate power levels the generated thrust is less than $T_{\max}$ and larger than $T_{\min}$. Moreover, unlike high-thrust propulsion systems, low-thrust propulsion is capable of thrusting almost *continuously*.

These facts have implications for using such engines in the design of interplanetary missions. As the thrust is continuous, the trajectory also changes *continuously*. That means that the relatively simple two-body approximation no longer applies – the trajectory will not even by approximation be a conic section anymore. Moreover, the spacecraft's power either comes from Solar panels (SEP), or from nuclear batteries (NEP). In SEP the thrust level thus depends on the spacecraft's distance to the Sun, as the influx of Solar energy decreases as the square of the distance. In NEP the engine can not be used to its full potential, because the power levels associated with nuclear batteries are generally quite low. In either case, using low-thrust propulsion implies that the trajectory design process is entirely different from that used for high-thrust missions.

12.2. Exponential Sinusoids

The equations of motion of a continuously thrusting spacecraft, can be stated as [Petropoulos, 2001]:

$$\ddot{\boldsymbol{r}} - \boldsymbol{r}\dot{\theta} + \mu/r^2 = a_T \sin\alpha \tag{12.1}$$

$$\frac{1}{\boldsymbol{r}}\frac{d}{dt}\left(r^2\dot{\theta}\right) = a_T \cos\alpha \tag{12.2}$$

where a_T is the acceleration of the spacecraft due to its thrust, and α is the thrust angle (see also Figure 12.2). In these equations, no perturbations are included other than the spacecraft's thrust. [Petropoulos et al., 1999] showed that if it is assumed that this thrust is always tangential to the current flight path,

$$\alpha = \gamma + n\pi,\ n \in \{0,\ 1\}, \tag{12.3}$$

it is possible to represent the geometry of the path traversed by a spacecraft subject to these equations of motion by an Exponential Sinusoid (ExpoSin). Such an ExpoSin can be represented in polar coordinates as

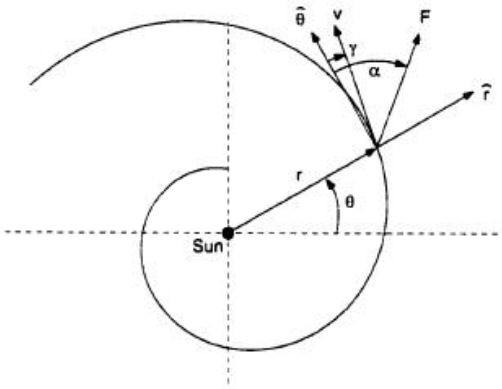


Figure 12.2 Definition of various parameters in a low-thrust trajectory. Figure copied from [Petropoulos, 2001].

$$r = k_0 e^{k_1 \sin(k_2\theta+\varphi)}, \tag{12.4}$$

where k_0 is called the *scaling factor*, k_1 the *dynamic range parameter*, k_2 the *winding parameter*, and φ is the *phase angle*. In this equation, r and θ are simply the planar polar coordinates, respectively.

The assumption of tangential thrust seems quite reasonable when looking at the orbital energy gain possible from a (continuously) thrusting spacecraft:

$$\Delta\epsilon = \frac{(V+\Delta V)^2}{2} - \frac{V^2}{2} = \frac{(\boldsymbol{V}+\Delta\boldsymbol{V})\cdot(\boldsymbol{V}+\Delta\boldsymbol{V}) - \boldsymbol{V}\cdot\boldsymbol{V}}{2} = \frac{2\boldsymbol{V}\cdot\Delta\boldsymbol{V} + \Delta\boldsymbol{V}\cdot\Delta\boldsymbol{V}}{2},$$

which indeed shows that when the thrust is tangential (i.e., $\boldsymbol{V}\cdot\Delta\boldsymbol{V}$ is maximal) the energy gain per unit of time and mass is maximal.

The following equations will make the geometry described by the ExpoSin physically relevant, and give a relation between the position on the ExpoSin and time:

$$\dot{\theta}^2 = \left(\frac{\mu}{r^3}\right)\Gamma, \tag{12.5}$$

$$a_N = \frac{(-1)^n \tan\gamma}{2\cos\gamma}\Gamma\left[1 - k_2^2(1-2k_1 s)\Gamma\right] \tag{12.6}$$

$$t_f = \int_{\theta_0}^{\theta_f} \sqrt{\frac{r^3}{\Gamma\mu}}\, d\theta, \tag{12.7}$$

where t_f is the time of flight,

$$a_N = \frac{a_T}{\mu/r^2}, \tag{12.8}$$

is the normalized thrust, and

$$\begin{aligned} \Gamma &= \left(\tan^2\gamma + k_1 k_2^2 s + 1\right)^{-1}, && (12.9)\\ s &= \sin(k_2\theta + \varphi), && (12.10)\\ c &= \cos(k_2\theta + \varphi), && (12.11)\\ \tan\gamma &= k_1 k_2 c. && (12.12) \end{aligned}$$

[Petropoulos, 2001] assumed that all motion takes place in two dimensions. The reference plane they used was the orbital plane of the departure point (in their case, Earth). Using this additional assumption, several constraint equations can easily be derived that delimit possible values for the parameter k_2. The first constraint equation depends on whether the trajectory is outbound or inbound. For outbound trajectories ($|\boldsymbol{r}_2| > |\boldsymbol{r}_1|$),

$$k_2^2 \leq \frac{\tan^2\gamma - 2k_{12s}\ln\left(r_{\min}/r_B\right)}{\left[\ln\left(r_{\min}/r_B\right)\right]^2}, \tag{12.13}$$

and for inbound trajectories ($|\boldsymbol{r}_1| > |\boldsymbol{r}_2|$),

$$k_2^2 \leq \frac{\tan^2\gamma - 2k_{12s}\ln\left(r_B/r_{\max}\right)}{\left[\ln\left(r_B/r_{\max}\right)\right]^2}, \tag{12.14}$$

where $k_{12s} \equiv k_1 k_2^2 s$, r_B is the current radius, (the radius at the departure planet), and $r_{\min}$ and $r_{\max}$ the minimum and maximum projected radii (the projection of the true radial vector onto the reference plane). The second constraint equation follows from the trigonometric identity $\cos^2(x) + \sin^2(x) \equiv 1$ applied to Equations 12.9 through 12.12:

$$k_1^2 k_2^4 - k_2^2\tan^2\gamma - k_{12s}^2 = 0. \tag{12.15}$$

If at the point of departure the flight path angle γ is known, this equation can serve as a check to see whether the chosen value of k_2 makes this ExpoSin flyable. This check will prove very useful in section 12.4, which deals with targeting with ExpoSin's and where the initial flight path angle γ_1 plays an important role. The third constraint equation follows from the demand that ExpoSin trajectories conform to the equations of motion:

$$\left|k_1 k_2^2\right| \leq 1. \tag{12.16}$$

If this constraint is not met for given values of k_1 and k_2, the corresponding ExpoSin trajectory does not describe a physical trajectory.

12.3. First-order Estimates from ExpoSins

Some simple but useful quantities that can be derived from the ExpoSin-equations are given here. The first of these, *Tsiolkovskii's equation for ExpoSins*, gives the end mass of the spacecraft when the initial mass and the ExpoSin parameters are known. This particular

equation will find its most important application in the chapter on cost functions, chapter 14. The second of these, an estimation of the *burn program* associated with an ExpoSin, will prove particularly useful as an initial estimate for the second-order procedure by Sims and Flanagan [1999], to be discussed in section 13.2.

12.3.1 Tsiolkovskii's Equation for ExpoSins

From the principles of rocket motion, it can be derived that

$$T = \dot{m} g_0 I_{\text{sp}} = \dot{m} c_{\text{eff}}, \tag{12.17}$$

where T is the rocket's thrust, g_0 the Earth's average gravitational acceleration at sea level, I_{sp} is the specific impulse, c_{eff} the effective exhaust velocity and $\dot{m}$ the mass flow. From Newton's first law it follows that for a thrusting spacecraft,

$$T = ma, \tag{12.18}$$

with m the instantaneous mass, and a the instantaneous acceleration caused by the thrust T. If Equation 12.17 is substituted in Equation 12.18, and it is assumed that the spacecraft follows an ExpoSin trajectory, it follows that

$$\dot{m} = \frac{m a_t}{c_{\text{eff}}},$$

where $a_t = (\mu/r^2) a_N$ follows from Equation 12.6. By virtue of the product rule from Calculus, this can be rewritten as

$$\frac{dm}{dt} = \frac{dm}{d\theta}\frac{d\theta}{dt} = \frac{dm}{d\theta}\dot{\theta} = \frac{m(\theta) a_t(\theta)}{c_{\text{eff}}},$$

with $\dot{\theta}$ as in Equation 12.5. This equation is a separable first-order differential equation, which is easily rewritten as

$$\frac{dm}{m} = \frac{a_t}{c\dot{\theta}} d\theta.$$

Performing an integration on both sides gives

$$\begin{aligned} \int_{m_0}^{m_e} \frac{dm}{m} &= \int_{\theta_0}^{\theta_f} \frac{a_t d\theta}{c\dot{\theta}} \\ \Rightarrow \ln \frac{m_e}{m_0} &= \frac{(-1)^n \sqrt{\mu}}{2 g_0 I_{\text{sp}}} \int_{\theta_0}^{\theta_f} \tan\gamma \sqrt{\frac{\Gamma\left(k_1^2 \tan^2\gamma + 1\right)}{r}} \left[1 - k_2^2 (1 - k_1 s)\Gamma\right] d\theta, \end{aligned} \tag{12.19}$$

with symbols as before. This is the equivalent of Tsiolkovskii's equation for ExpoSin's, which gives a first-order estimation of the dry mass m_e for a specific ExpoSin. Note that the Method of Patched ExpoSins (MPE) usually results in a *sequence* of ExpoSin's. Naturally, also Equation 12.19 can simply be "patched together" to give an estimate of the spacecraft's overall dry mass.

12.3.2 Burn Program

It is not only useful to have an idea of the acceleration implied by an ExpoSin trajectory, it is also imperative to have an idea on the actual *thrust direction* and *magnitude*. The easiest way to find such a thrust profile is by taking Equations 12.6 and 12.8, and multiplying a_T by the instantaneous mass (Equation 12.19) to find the thrust as a function of θ. If required, this can be transformed into a function of time with the help of Equation 12.7. Note that the actual thrust direction is simply *tangential* to the ExpoSin, because of the initial assumption of tangential thrust. Thus, it is equal to the direction of the instantaneous velocity:

$$\begin{aligned} \boldsymbol{V}(\theta) &= \dot{r}\hat{\boldsymbol{r}} + r\dot{\theta}\hat{\boldsymbol{\theta}} \\ &= r\dot{\theta} \cdot \left(\hat{\boldsymbol{r}} \cdot \tan\gamma + \hat{\boldsymbol{\theta}}\right) \end{aligned} \tag{12.20}$$

For the second-order method from [Sims and Flanagan, 1999], it is required to have a burn program that is given in terms of j small ΔV_j given in small but finite (and equal in length) segments of the trajectory. If an ExpoSin trajectory is divided up in j small segments, it is tempting to set the direction of each ΔV_j found from Equation 12.20 equal to that of the instantaneous velocity $\boldsymbol{V}_j$ at the same point j, because the thrust is everywhere tangential. However, since each ΔV_j is the *average* velocity change over the whole finite interval, also the direction should be the *time averaged* direction of the thrust during that interval. Finding the *correct* directions for the ΔV_j can indeed be done (using *Whittaker's theorem*), but it is rather involved. Additionally, as will be pointed out in section 12.5, ExpoSin's only provide correct estimates for the relative positions of the planets and transfer times, but *not* of the actual trajectories to be followed. Therefore, just using Equation 12.20 is more than accurate enough to obtain reasonable first estimates.

12.4. The Low-thrust Orbital Boundary-value Problem

Although the equations involved with ExpoSin's are fairly straightforward, solving the orbital boundary-value problem with ExpoSin's is less than straightforward. This is mostly due to the rather large number of degrees of freedom of an ExpoSin, as expressed in the constants k_0 through ϕ. Also, the time of flight can not be stated in a finite number of terms (see Equation 12.7) so that its value needs to be determined with *quadrature* methods. Any algorithm solving the orbital boundary-value problem has to put great importance on efficiency. Fortunately, the problem has been solved more or less generally by Izzo [2006]. Although his original formulation was found to have a few problems (see [Corradini, 2009]), these can be easily circumvented and incorporated into a Lambert targeting routine. Also, the efficiency of the original algorithm left much to be desired, so several improvements in this regard were made before it was implemented in Skipping Stone. Unfortunately, the full description of this algorithm (and its implementation issues encountered and algorithmic improvements made) grew too large to be included here. The interested reader is therefore referred to appendix C which describes the algorithm in full.

12.5. Method of Patched Exponential Sinusoids

ExpoSin's have successfully been used in various works (by Vasile et al. [2006] or Schütze et al. [2007] for example) in an MGA-context. That is, similar to the MPC (see section 9.2), the MGA problem for low-thrust propulsion may be solved by patching together *exponential sinusoids* instead of the simpler conic sections. This method will be referred to as the MPE. A strong advantage of the MPE procedure is that once the MPC is fully implemented, the MPE only requires two minor changes, making its implementation quite simple.

First, the Lambert targeter in the MPE must be capable of solving orbital boundary value problems using ExpoSin's instead of conic sections, and secondly, the planetary GAM must executed fully un-powered. This implies that no ΔV is to be executed at the planet's pericenter, and the additional ΔV_{add} must be kept zero at all times. While the use of a different Lambert targeter poses no problem, the second of these changes *does* make the MPE problem considerably more complicated. When GAM's are to be executed fully un-powered, *most* of the search space will be rendered infeasible. Although this is to be expected (low-thrust simply is not as powerful as high-thrust), the optimization will most likely require much more time before any viable solutions are found, if at all. In other words, the MPE method can be expected to place very high demands on the robustness and efficiency of the global optimizer that is to solve it.

It should also be noted that the MPE does *not* return a trajectory that is optimal in any respect. Indeed, as found by Schütze et al. [2007], other methods such as the pseudo-equinoctial shaping they used produce trajectories that have a much higher degree of optimality compared to the ExpoSin's, albeit at an increased computational cost. All that the MPE provides is a good first estimate on the initial and transfer times, and the relative positions of the planets. But again, it should be stressed that the actual *shapes* described by the ExpoSin's resulting from an MPE run should not be interpreted as the final, optimal trajectory.

It could be argued that one additional constraint should be checked after the Lambert-targeting procedure, namely whether the accelerations over the whole trajectory required by the ExpoSin can actually be met by the propulsion system. The thrust that can be realized by the propulsion system generally depends on the distance to the Sun $\boldsymbol{r}(\theta)$ (see section 3.4), which is known for all values of θ by virtue of Equation 12.4, and the instantaneous mass of the spacecraft, which can be determined with Equation 12.19. However, especially this last equation is quite costly as it involves many more integrations. Moreover, in light of the previous statement that the ExpoSin is rather inaccurate when it comes to optimality, the usefulness of performing this check is questionable. For that reason, this costly type of check will not be performed at all, and will be left to the more accurate second-order algorithms discussed in the next chapter.

13 Optimization of Low-Thrust Trajectories

The MPE's implicit assumption of tangential thrust is a strict and artificial restriction to the problem, imposed only to keep the governing equations analytical. This makes the MPE a nice first-order method, but its results can certainly not be directly applied to real missions. For that, more elaborate and complicated methods are required. This Chapter first gives an outline of some general aspects of low-thrust optimization, and the various pitfalls and advantages can be expected with low-thrust optimizations. Two direct optimization methods for low-thrust optimization are elaborated next; the method of *patched micro-conics*, and the method of *collocation*.

13.1. Direct vs. Indirect Methods

As mentioned before, optimizing a general low-thrust problem is a daunting task. Over the years, dozens of different methods have emerged that try to efficiently solve this optimization problem. These methods are usually subdivided into *direct* and *indirect* methods (see [von Stryk and Bulirsch, 1992] for a nice introduction on this subject). Indirect methods try to find the root of the first variation of the problem's constrained partial differential equations, which arises from the equations of motion by means of calculus of variations or constrained least-squares. Direct methods on the other hand, do not consider *variations* of the equations of motion, but rather employ a simplified representation of the equations of motion *themselves* to convert the problem into a nonlinear-programming problem.

13.1.1 Indirect Methods

Indirect methods are based on the *Pontryagin minimum principle*. They are "indirect" in the sense that these methods attempt to find stationary points in the first variation of the system's Hamiltonian, rather than in the objective function and its constraints. Indirect methods have the principle advantage of *accuracy* over direct methods – once a problem is

solved using an indirect method, the optimal thrust program at *every* arbitrary point in time is known, because the output of the indirect method is a *function* rather than a set of points. Also, it absolutely *guarantees* all the constraints are met, and the result is indeed the *optimal* result [Kemble, 2006].

However, indirect methods have some *major* disadvantages. The underlying theory is much more complicated and far less transparent than that generally used in direct methods. They are also exceptionally difficult to automate; most indirect methods require manual tweaking at nearly every stage of the optimization. Moreover, since the optimization is done on the first variation of the equations of motion rather than the equations themselves, many of the parameters do not describe actual physical quantities, so it requires *practice* to gain insight in the intermediate results it produces.

Also, since they converge to a *stationary* point of the first variation of the Hamiltonian, it is generally not known whether this point is a *maximum* or *minimum* of the functional in question. As a result, indirect methods require a rather well-chosen initial estimate to actually achieve convergence to the optimum. In fact, for many practical problems, their radius of convergence is almost ridiculously small, often leading to the paradoxical conclusion that indirect methods can only find the optimal solution if it is already known. As such, indirect methods are not suited for the high degree of automatization that is aimed for in Skipping Stone.

13.1.2 Direct Methods

The direct methods behave more or less the opposite way – the accuracy and optimality of its results are usually less, and the constraints are normally optimization parameters themselves rather than hard constraints, so they are sometimes (slightly) violated. However, convergence almost always occurs even for poorly chosen initial estimates. Also, implementing and using direct methods is considerably easier than indirect methods. For these reasons, the second-order accuracy for this thesis research will be met by employing direct methods. Two different methods are discussed here; the *method of patched micro-conics*, and *collocation*.

13.2. Method of Patched "Micro-Conics"

A very good second-order method is the direct method devised by Sims and Flanagan [1999]. Their method allows the MGA problem to be subdivided in several NLP's, which may readily be optimized with SQP (see section 6.2). The method is illustrated in Figure 13.1, and works as follows. The points where one leg of the trajectory starts and ends are called *control points* (or nodes). These are normally the (minor) planets that result from the corresponding Lambert problem performed in first-order accuracy (chapter 12). Each leg between two consecutive control points is subdivided into several segments of equal time duration. In the middle of each segment, a small ΔV is applied. In this fashion, the trajectory is propagated both forward from the start control node, and backward from the end control node, until a certain *match point* somewhere in between the two control nodes is reached. This propagation can be performed simply by solving the corresponding 2-body problem (see appendix F.3),

with the Sun as the central body. This is done for each leg of the whole trajectory. Any GAM that is performed at each last control node, is modelled simply as an instantaneous rotation of the heliocentric velocity.

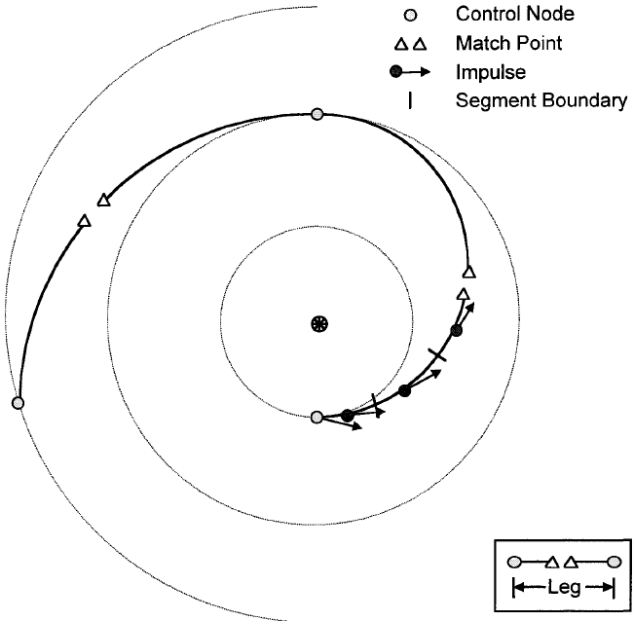


Figure 13.1 Principles of the MPμC. This is Figure 1 from [Sims and Flanagan, 1999]

Since this method is *in principle* the same as a high-thrust problem with a large amount of DSM's, it will be referred to as the Method of Patched Micro-Conics (MPμC). The accuracy of this method was tested by Sims and Flanagan [1999]. They optimized two scenarios (*Earth-Mars-Vesta* and *Earth-Tempel1*) using both this method and a well-established indirect method (SEPTOP, see [Sims and Flanagan, 1999, Ref. 4]). In this small comparative study, a relatively large segment-duration of $\sim$8 days was assumed. The optimized end masses resulting from both methods differed always less than $\sim$0.06%, which proves that the MPμC indeed is a simple yet excellent second-order approach.

Constraints The parameters to be optimized for this mission are the end mass and total time of flight. As this method does not allow an easy way to use a multi-objective approach, only *one* parameter will be optimized:

$$\text{find } \max m_{\text{dry}}(\boldsymbol{x}) \tag{13.1}$$

It will therefore be assumed that the initial positions of the planets, as provided by the MPE, will already result in the minimum time of flight for the final, optimized trajectory.

The introduction of the match points into the problem formulation introduces *equality constraints*: in a match point, the velocity, position and mass of the spacecraft must all be continuous:

$$m_1(\boldsymbol{x}_{\text{match}}) - m_2(\boldsymbol{x}_{\text{match}}) = 0, \tag{13.2}$$
$$\boldsymbol{\chi}_1(\boldsymbol{x}_{\text{match}}) - \boldsymbol{\chi}_2(\boldsymbol{x}_{\text{match}}) = \mathbf{0}, \tag{13.3}$$

where $\boldsymbol{\chi}_{\text{match}}$ is the complete state vector at the match point, and the mass m is computed with Tsiolkovskii's equation (the original high-thrust version). Additionally, the magnitude of each ΔV may not exceed the maximum possible value that can be achieved by the low-thrust engine during one segment, and the pericenter distances at the planets where a (un-powered) GAM is performed must always be larger than the minimum allowed values (see Equation A.32 and Table A.2). These latter two statements can be formulated as *inequality constraints*:

$$r_p(\boldsymbol{x}) - r_p^{\max}(\boldsymbol{x}) \geq 0, \tag{13.4}$$
$$T_{\text{possible}}^{\text{segment}}(\boldsymbol{x}) - T_{\text{required}}^{\text{segment}}(\boldsymbol{x}) \geq 0. \tag{13.5}$$

Taking the launch date t_0, the $\mho$ travel-times t_i^f, $i = 1 \ldots \mho$ and the magnitude and direction of each individual impulse as input variables, this constrained NLP problem will have a decision vector of very large dimensionality:

$$\boldsymbol{x} = [t_0,\, t_1^f,\, t_2^f,\, \ldots,\, \Delta V_1^x,\, \Delta V_1^y,\, \Delta V_1^z,\, \Delta V_2^x,\, \ldots] \tag{13.6}$$

Note that especially the directions and magnitudes for each impulse (3 variables per impulse), and the large number of impulses ($\sim$120 were used by Sims and Flanagan [1999]) produce many decision variables for the overall trajectory (already $3 \times 120 = 360$ for the impulses alone). These NLP's can only practically be solved by an SQP-algorithm (see section 6.2). Note that since the travel-times are also part of the decision vector, the positions of the control points (planets) are allowed to vary. Naturally, removing these variables will reduce the problem's complexity (and thus the computation time), which is useful for initial testing purposes. However, the true optimum is hardly likely to be located when the planets are at the exact positions as found by the MPE, so for realistic optimizations, the times-of-flight should indeed be included in the decision vector.

The complete procedure is then much like the one used by McConaghy et al. [2001] and Debban et al. [2002], with one important difference – both these investigations could not prune the solution space with the ExpoSin Lambert-problem devised by Izzo [2006] because this method did not exist at that time. Therefore, their search space was considerably larger and it can therefore be expected that results from the above procedure will have *less*, but *more promising* initial estimates, so that the optimization will yield better results while converging faster.

Initial Estimates Two different initial estimates can be used as starting values for the decision vectors. One initial estimate is the best result found from applying the MPE, and converting it into a burn program, as described in section 12.3.2. This estimate may however be too far off the actual optimum, since the ExpoSin's are usually far removed from optimality. Also, generating this estimate requires several integrations (to find the instantaneous mass), which may be too costly when large amounts of initial estimates need to be found. The

other initial estimate is the one used by Sims and Flanagan [1999] themselves. They simply took the heliocentric velocities at both control nodes in a leg (initially, simply the planet's velocities), and linearly interpolated all components of the velocities towards each other to the match point.

Implementation Issues Although the method is powerful and made intuitive by staying close to the physics of the problem, it has several issues that should be considered.

First, the spacecraft's position at each intermediate point should be given as *state vectors*, because a ΔV (given in Cartesian coordinates) is to be added to each part of the trajectory. Using Kepler elements to progress the orbit to another intermediate point would therefore require countless conversions between the instantaneous Kepler elements and Cartesian coordinates. Therefore, the most convenient way to incorporate all the ΔV-impulses, is to use the State Transition Matrix (STM) (see section B.3.2) to progress the spacecraft's orbit to the intermediate points in a leg.

The biggest drawback of the MPμC is that analytical differentiation of the constraint functions is rather complicated, if not impossible; it is quite hard to express in a function what the change in position of the spacecraft at a match point will be, if only *one* of the series of impulses is changed. This is even more complicated when the initial date is changed. Therefore, the gradient and Hessian of all the constraint functions *must* be computed numerically. If there are N dimensions in the decision vector, obtaining the gradient requires $2N$, and the Hessian $3N^2$ FE's (when using central differences). Considering the fact that every FE requires progressing the orbit with N different state-transition matrices, computing these derivatives is an extremely costly affair. All efforts to make this method more efficient should therefore be focussed on the computation of the derivatives of the constraint functions.

13.3. Collocation

A relatively simple method that can be used for trajectory optimization is *differential inclusion* (DI) [Seywald, 1993]. This method works by discretizing the path between two points into N separate parts. Interpolations between the N points give rise to an additional $N-1$ points (those points halfway through the interpolated paths), and all of these points then have to satisfy the following constraints:

- The equations of motion must be satisfied
- The thrust required at all points should be always less than or equal to what the engine can deliver at those locations.

The exact locations of the N associated points are then optimized by solving this NLP-problem. Thus, when this optimization is carried out, it will result in $2N-1$ dynamically consistent points, which together form an approximation to the optimal trajectory and its associated thrust plan.

Although simple and elegant, DI generally yields coarse approximations at best. The method of *collocation* is a drastic improvement to DI in the sense that collocation interpolates the intermediate paths by a *cubic polynomial*, instead of a line. This method was first outlined by Hargraves and Paris [1987], and can be formulated as follows (see also Figure 13.2):

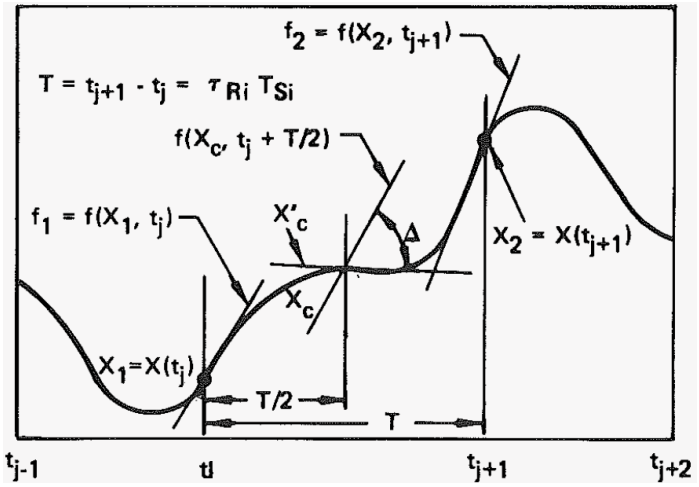


Figure 13.2 The basic principles of collocation. The locations of the vertices are optimized, while demanding that the mid-points of each interval satisfy the equations of motion. The resulting interpolated trajectory is then a dynamically consistent set of points, which form an accurate approximation to the true optimum trajectory. This is [Hargraves and Paris, 1987, Figure 2].

For the cubic (Hermitian) interpolation, it is assumed that the state x may be represented by the cubic polynomial

$$x = c_0 + c_1 S + c_2 S^2 + c_3 S^3, \tag{13.7}$$

where S is a scaled and translated version of the time t to a specific interval. For the sake of brevity it is assumed that the interpolation interval is $0 \leq S \leq 1$. Then, using the notation $x(0) = x_0$, $x(1) = x_1$, $dx/ds(0) = \dot{x}_0$, and $dx/ds(1) = \dot{x}_1$, the derivative of Equation 13.7 may be written as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ \dot{x}_1 \\ x_2 \\ \dot{x}_2 \end{bmatrix} \tag{13.8}$$

The (4×4) matrix may be inverted, which results in

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ x_2 \\ \dot{x}_2 \end{bmatrix} \tag{13.9}$$

Using this equation to evaluate Equation 13.7 at $S = 1/2$ gives the value of x_c at the center of the segment:

$$x_c = \frac{1}{2}(x_1 + x_2) + \frac{T}{8}(f_1 - f_2), \tag{13.10}$$

where T is the (real) length of the interval, and $f_i = f(i)$ is the differential equation from the equations of motion:

$$\dot{x}_i = f(x_i, u, m, t) \tag{13.11}$$

Doing the derivation for $\dot{x}_c$ gives

$$\dot{x}_c = -\frac{3}{2T}(x_1 - x_2) - \frac{1}{4}(f_1 + f_2). \tag{13.12}$$

The locations at the mid-points of the interpolated trajectory should be chosen such that the difference between their function value and interpolated value (the *defect*, Δ) goes to zero:

$$\begin{array}{rcll} \Delta & = & f_c - \dot{x}_c & \to 0 \\ & = & f_c + \frac{3}{2T}(x_1 - x_2) + \frac{1}{4}(f_1 + f_2) & \to 0 \end{array} \tag{13.13}$$

which should be the goal of the optimizer. Of course, the analysis outlined above holds for all states in the state vector $\boldsymbol{\chi} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$, and the mass m. However, applying the same technique is not possible for the acceleration $\boldsymbol{u}$ due to the thrust at each point; the equations of motion $f(x,\, u,\, m,\, t)$ alone can not determine the derivatives of $\boldsymbol{u}$, so that the corresponding condition $\Delta(\boldsymbol{u}) \to 0$ can not be used. To still include this term (which is required to formulate the correct equations of motion f), it can only be interpolated *linearly*, similar to DI:

$$u_c = \frac{u_1 + u_2}{2}. \tag{13.14}$$

With the above, the problem can be formulated as the NLP-problem

$$\text{Find} \qquad \max m_{\text{dry}}(\boldsymbol{x}) \tag{13.15}$$

s.t.

$$\begin{aligned} \Delta(\boldsymbol{x},\, \boldsymbol{x}_c) &= 0, && (13.16) \\ \dot{m}(\boldsymbol{x},\, \boldsymbol{x}_c) &\leq 0,. && (13.17) \\ |\boldsymbol{u}(\boldsymbol{x},\, \boldsymbol{x}_c)| - \frac{T_{\max}}{m(\boldsymbol{x},\, \boldsymbol{x}_c)} &\leq 0. && (13.18) \end{aligned}$$

where $T_{\max}$ is the maximum thrust deliverable by the spacecraft's engine, $\boldsymbol{x}$ is the decision vector

$$\begin{aligned}\boldsymbol{x} = \; & [x_0 \quad y_0 \quad z_0 \quad \dot{x_0} \quad \dot{y_0} \quad \dot{z_0} \quad u_0^x \quad u_0^y \quad u_0^z \quad m_0 \\ & x_1 \quad y_1 \quad z_1 \quad \dot{x_1} \quad \dot{y_1} \quad \dot{z_1} \quad u_1^x \quad u_1^y \quad u_1^z \quad m_1 \\ & x_2 \quad y_2 \quad z_2 \quad \dot{x_2} \quad \dot{y_2} \quad \dot{z_2} \quad u_2^x \quad u_2^y \quad u_2^z \quad m_2 \\ & \vdots \\ & x_N \quad y_N \quad z_N \quad \dot{x_N} \quad \dot{y_N} \quad \dot{z_N} \quad u_N^x \quad u_N^y \quad u_N^z \quad m_N]\,,\end{aligned}$$

e.g., the complete state vector, accelerations due to thrust, and the masses at all intermediate points N. The quantity $\boldsymbol{x}_c$ is the same vector, but then found by the interpolation from Equation 13.10. The non-linear inequality in Equation 13.17 is required to make the problem physically realistic – without this constraint, there is nothing that would stop the solution from assuming *positive* mass flow at some points, and thus an *increasing* mass and corresponding higher final mass than start mass. The non-linear inequality in Equation 13.18 is necessary to ensure that the thrust at all points is not greater than the maximum thrust deliverable by the spacecraft's engine $T_{\max}$. Finally, the function $f(\boldsymbol{x}_i)$ is defined as

$$f(\boldsymbol{x}_i) = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \\ -\mu x_i/r^3 + u_x^i \\ -\mu y_i/r^3 + u_y^i \\ -\mu z_i/r^3 + u_z^i \\ -\frac{um}{g_0 I_{\text{sp}}} \end{bmatrix}^T \tag{13.19}$$

where $r = \sqrt{x_i^2 + y_i^2 + z_i^2}$, $u = \sqrt{(u_i^x)^2 + (u_i^y)^2 + (u_i^z)^2}$, and μ is the standard gravitational parameter of the central body. The last of these equations follows from the standard equation for a thrusting rocket:

$$T = m \cdot |\boldsymbol{a}| = \dot{m} \cdot c_{\text{eff}} = \dot{m} g_0 I_{\text{sp}}.$$

As this NLP-problem is to be optimized with SQP, it would be convenient to have expressions for the gradients of both the objective function and its constraints. The gradient of the objective function is quite easy to determine (note the -1 in the bottom right):

$$\begin{aligned}\nabla F_{\text{obj}}(\boldsymbol{x}) = \; & [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ & 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ & 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ & \ldots \\ & 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -1]\,,\end{aligned} \tag{13.20}$$

since the objective function $F_{\text{obj}}(\boldsymbol{x})$ is equal to the negated last element in $\boldsymbol{x}$. Given that the gradient of the objective function consists of constants, it should be obvious that the Hessian matrix consists entirely of zeros. The gradients for the constraint functions (Equations 13.17 through 13.18) are much less easy to determine. Although it is possible to determine analytical derivatives for them, these derivatives will not be given here, since they are so lengthy that it is unrealistic to assume that they can be given error-free the first time round. Moreover, some constraints might need to be adjusted (or added) to keep the optimization efficient and accurate. This would necessitate doing the derivation again from scratch. To avoid these

issues, their gradients will be computed numerically, and the analytical derivatives will be left as a recommendation.

14 Optimization Strategy

14.1. Costfunctions, all Scenarios

Since the low-thrust MGA procedure closely resembles the high-thrust version, the cost functions that should be used for the low-thrust MGA procedure are indeed very similar as well. There are only a few issues that differ from the high-thrust MGA procedure:

- All Lambert problems should be solved with the ExpoSin's Lambert targeter
- The distances to minor planets should now be calculated from an elliptic orbit to an ExpoSin, as described later this Chapter.

The most important difference between the two is that in the low-thrust MGA procedure, all GAM's should be executed *un-powered*. That is, no ΔV or additional ΔV_{add} can be used during the GAM's:

- The amount of mass expelled in a low-thrust trajectory does not follow from the GAM's, but rather from the trajectories comprising the individual legs in a solution.

Since this difference severely impacts the amount of valid solutions in the search space, it is better to re-define some of the penalty functions from the high-thrust algorithm to improve the convergence rate of (semi-) directed global optimizations. To that end, the original cost function used for mission scenario 1 in high-thrust should be adjusted as follows:

Steps 0 Execute **Step 0** and **Step 1** from the high-thrust cost function.

Step 1 Compute the minimum flyby distance r_p^i and associated values for the total required mass m_{req}^i for each successive leg i. In addition, calculated the following values:

$$c_1 = r_p^{\min,i} - r_p^i \tag{14.1}$$
$$c_2 = m_{\text{dry}} - m_{\text{final}} \tag{14.2}$$

where $r_p^{\min,i} = R^i_{\text{body}} + h^i_{\min}$ is the minimum allowable flyby distance at the i^{th}GAM body and $m_{\text{final}} = \sum_{i=1}^{\mho} m^i_{\text{req}}$ is the spacecraft's final mass calculated by summing the propellant expenditures during all of the $\mho$ legs in the current `seq` using Equation 12.19. Note that unlike the high thrust procedure, the inclusion of c_2 is no longer a safeguard but an imperative step in the procedure; the user has no influence over its value.

Step 4 The value of the cost function is then:

$$F_{\text{cost}} = -m_{\text{final}} + \exp(c_1 + c_2) - 1 \tag{14.3}$$

Save for the adjusted cost function above, all the other cost functions for all mission scenarios can be left completely unaltered. How the (time-dependent) distances from the ExpoSin's to MP's are to be calculated will be discussed shortly.

14.2. Costfunctions, Second-Order Accuracy

As mentioned in the previous chapter, the optimization of low-thrust trajectories in second-order accuracy is to be carried out with two distinct methods – the MPμC and the method of collocation. One advantage of these methods is that both of them do not need a separate cost-function for the trajectory to the SBS without visiting MP's; the methods are designed to find the optimum trajectory automatically. It is however all the more problematic when including MP's in the optimization. The computation of the time-dependent minimum distances $\boldsymbol{D}(t)$ to MP's now heavily depends on the method used; the MPμC is based on conic sections, whereas the method of collocation uses Hermitian interpolation between points to approximate the integration.

Fortunately, the optimum results from the first-order low-thrust procedure will give this method fairly good initial estimates on the transfer times t^i_f and launch date t_0, as well as a (relatively small) sub-set of candidate MP's $\text{MP}_{\text{candidate}}$ for mission scenarios 2 through 4. Since both the second-order methods are fundamentally *local* and *constrained* optimizations, it is fairly straightforward to add MP-flybys in the form of additional inequality constraints. For both methods, an amount of inequality constraints will be added equal to the amount of candidate MP's in the set $\text{MP}_{\text{candidate}}$. The value of each MP inequality constraint must be computed as follows:

Step 0 Compute the minimum time-dependent distance $D(t)$ from the current estimate on the optimum trajectory, to the candidate MP.

Step 1 This distance must obey the constraint:

$$D(t) - D_{\text{threshold}} < 0, \tag{14.4}$$

where $D_{\text{threshold}} = 0.15$ AU, as discussed in section 14.4.

Note that both methods use a set of intermediate positions of the spacecraft to function; the MPμC implicitly determines these positions by progressing its orbit and applying a ΔV at each point, whereas the method of collocation requires intermediate points to serve as the support points for the interpolation. These positions can be converted to time (using Kepler's equation), providing the possibility to recover the position of the MP at the same times (again using Kepler's equation). The distance from the spacecraft to the MP can then be calculated at each of these epochs, giving a good initial estimate on the location and epoch of the minimum approach distance. The *actual* minimum distance must then be calculated with the methods outlined in section 11.4.1 (for the MPμC), or with the cubic-splines interpolation method outlined in section 14.4 (for the method of collocation). Note that since it is known beforehand which leg will bring the spacecraft close to which MP (as follows from the global first-order method), the calculation of this distance can be done quite efficiently by using only the initial points from that specific leg.

Note that even when the constraint of $D(t) < 0.15$ AU is met, the minimum distance will still be far too large to be considered an actual flyby. This is nevertheless required, because the initial estimate from the first-order ExpoSin's procedure differs greatly from the final second-order solution, which makes meeting the constraints defined above quite difficult. Therefore, the optimization that uses this relatively large threshold distance must be regarded as an attempt to create a better initial estimate for a *second* optimization, that uses the following *equality* constraints:

Step 0 Compute the minimum time-dependent distance $D(t)$ from the current estimate on the optimum trajectory, to the candidate MP.

Step 1 This distance must obey the constraint:

$$D(t) = 0 \tag{14.5}$$

This extremely strict constraint will no doubt be impossible to meet. Nevertheless, using this zero-distance demand will force the local optimizer to push the *maximum* distance found as far down as possible, thus obtaining the required results. This procedure will indeed require much manual tweaking and testing of the optimization parameters (allowed constraint violation, manually removing individual MP's, experimenting with additional weighting functions, etc.) necessitating many re-runs of the whole optimization procedure. This process is unfortunately a necessary burden in the current framework and the only practical means to tackle problems of this nature.

14.3. Dealing with the MPE in Global Optimizations

The MPE can be used in global optimizations in much the same fashion as the high-thrust MPC (see section 11.1). The decision vector for MPE problems still contains the launch date t_0, the times of flight $\pm t_f^i$ between the i bodies in `seq` (where the sign decides whether the long

or short way is to be used), and/or the number of complete revolutions $\pm m$ (where the sign decides whether the left or right branch is to be used). However, unlike the MPC, the MPE also requires the k_2-parameter for each ExpoSin-leg. This parameter can either be optimized by the ExpoSin-Lambert routine itself, or be chosen "blindly" (as part of the decision vector) so that the global optimizer will attempt to optimize it.

In principle, the first of these methods is equal to the method used for the SF; the value of the ΔV at pericenter is optimized for every SF problem. Preliminary testing with the SF indicated that it is no problem to optimize a few SF-problems, but can become problematic if it is executed inside a global optimization and thus needs to be done thousands of times. Optimizing the k_2-parameter to minimize propellant consumption *inside* the ExpoSin's Lambert targeter would involve many evaluations of the time-of-flight integral, its derivative, and Tsiolkovskii's integral (Equation 12.19, and quite possibly, *its* derivative), and *each* evaluation of an integral requires several hundred evaluations of the associated integrand. Needless to say, optimizing k_2 for each Lambert-problem is likely to be *too slow* to give reasonable computation times in a global optimization. Considering the fact that problems of this nature already occur with a *high-thrust* solution method (the SF, which has governing equations that are comparatively simple, and evaluate quickly), it seems better to optimize k_2 with the global optimizer. Therefore, to use MPE, the decision vector from Equation 11.2 should be modified to

$$\begin{aligned} X = [\ & t_0, \\ & \pm t_f^{(1)}, \quad k_2^{(1)}, \quad \pm m^{(1)}, \\ & \pm t_f^{(2)}, \quad k_2^{(2)}, \quad \pm m^{(2)}, \\ & \qquad \vdots \\ & \pm t_f^{(\mho)}, \quad k_2^{(\mho)}, \quad \pm m^{(\mho)}]. \end{aligned} \tag{14.6}$$

The obvious drawback of this is of course that the decision vector now has $\mho$ extra dimensions, which again increases the required time until convergence. However, this method is considerably less complicated to implement (and thus far less prone to implementation error), and the MPE can only provide the *approximate* location of the basin of attraction and does *not* generate optimal trajectories by itself. Therefore it is more than sufficient to find a value of k_2 that *approximately* optimizes the problem; finding the *actual* optimum will cost far more effort (computationally and programmatically) but is essentially "overkill".

What remains to be discussed is a lower and upper limit on k_2. As discussed by Izzo [2006] and Petropoulos et al. [1999], an absolute lower limit of $k_2 = 0.01$ applies in order to keep solutions physically realistic. Unfortunately, none of these authors mentions an *upper* limit – it appears their optimizations used no such constraint. However, the equation for the ExpoSin

$$r = k_0 \exp\left(k_1 \sin\left(k_2\theta + \phi\right)\right)$$

requires that $k_2 \leq 1$. This can be concluded from the periodicity of the sine-component; if $k_2 > 1$ the sine-function can complete *more than one period* in a single revolution, which means that the ExpoSin-trajectory is able to move past its largest and smallest radial distances *several times* per revolution. This sort of behavior is simply not possible for spacecraft

equipped with low-thrust propulsion systems, so that necessarily, $k_2 \leq 1$.

It is interesting to note that setting $k_2 = 1$ exactly results in behavior very similar to that of an elliptic orbit – it reaches its largest and smallest radial distances exactly once per revolution, and these distances are located on opposite sides of the central body and on the same axis. For suitable values of k_1, even the shapes of both trajectories are almost indistinguishable. It can however be shown that the location *as a function of time* as described by such an ExpoSin always differs from that in the corresponding elliptic orbit, increasingly so with increasing k_1. However, in the limit of $k_1 = 0$, there is no difference between the ExpoSin and a *circular* orbit[1].

Despite these differences it seems promising to assume $k_2 \leq 1$ so that in the limit the ExpoSin approximately models elliptic orbits. This is indeed what any low-thrust method should be able to do, as the spacecraft should also be able to *turn off* its engines and follow a ballistic (elliptic) trajectory.

14.4. Pruning Minor Planets in Costfunctions

Pruning minor planets from the search space in case the spacecraft is equipped with a low-thrust propulsion system, is considerably more complicated than compared to spacecraft with high-thrust systems. As a first order approximation, ExpoSin's will be used to find interesting regions in the search space. Although ExpoSin's can only produce a first-order approximation to the final situation and can be quite far from the optimal trajectory, they are analytical and fairly easy to work with. Therefore, some relatively simple methods will be developed that can prune minor planets which are well out of reach for the ExpoSin trajectory.

Based on Apses

An ExpoSin is a spiral trajectory, and as such, does not have apses. However, from its definition [Petropoulos and Longuski, 2004],

$$r = k_0 e^{k_1 \sin(k_2\theta+\phi)} \tag{14.7}$$

it can be readily derived that

$$\begin{aligned} r_{\max} &= k_0 e^{+|k_1|} \\ r_{\min} &= k_0 e^{-|k_1|} \end{aligned} \tag{14.8}$$

are the maximum and minimum *possible* heliocentric distances the spacecraft can assume. These values for $r_{\min}$ and $r_{\max}$ may be used to prune those minor planets that will always be further from the Sun than $r_{\max}$, or always closer to the Sun than $r_{\min}$. Thus,

$$\text{Prune MP when } r_p > r_{\max} \text{ or } r_a < r_{\min}, \tag{14.9}$$

[1]This is of course a rather unsurprising result, as with $k_1 = 0$ all the equations related to the ExpoSin degenerate into constants.

where r_a is the MP's aphelion, and r_p its perihelion. A more specific test may be derived from this principle. The extremal values $r_{\min}$ and $r_{\max}$ are only equal to the quantities 14.8 if the turn angle $\Delta\theta$ is large enough to put the quantity $\Phi = k_2\theta + \phi$ in the range $0 \leq \Phi \mod 2\pi \leq 2\pi$. Only when that is true can the sine-exponent assume its extremal values $+1$ and -1. However, if this is *not* the case (which will usually be the case if the MGA procedure is followed), the following rule should be used:

$$\begin{aligned} r_{\max} &= k_0 \max\left\{e^{k_1 \sin\Phi}\right\}, \\ r_{\min} &= k_0 \min\left\{e^{k_1 \sin\Phi}\right\} \end{aligned} \tag{14.10}$$

where the maxima and minima depend on the sign of k_1,

$$\begin{aligned} \max\left\{e^{k_1 \sin\Phi}\right\} &\rightarrow \max\left\{k_1 \sin\Phi\right\} \\ &= \begin{cases} k_1 \max\left(\sin\Phi\right) & \text{if } k_1 > 0, \\ k_1 \min\left(\sin\Phi\right) & \text{otherwise,} \end{cases} \end{aligned} \tag{14.11}$$

$$\begin{aligned} \min\left\{e^{k_1 \sin\Phi}\right\} &\rightarrow \min\left\{k_1 \sin\Phi\right\} \\ &= \begin{cases} k_1 \min\left(\sin\Phi\right) & \text{if } k_1 > 0, \\ k_1 \max\left(\sin\Phi\right) & \text{otherwise.} \end{cases} \end{aligned} \tag{14.12}$$

The maximum and minimum values of the sine function can readily be determined if the specific ExpoSin is the outcome of the low-thrust Lambert-targeter. In that case, $0 \leq \theta \leq \Delta\theta$, and so $\phi \leq \Phi \leq (k_2\Delta\theta + \phi)$. Then,

$$\max\left(\sin\Phi\right) = \begin{cases} +1 & \text{if } \Delta\theta \leq \frac{2\pi}{k_2}, \text{ or } (\phi \mod 2\pi) < \pi/2 \\ & \text{and } (\Psi \mod 2\pi) > \pi/2 \\ \max\left(\sin\phi,\, \sin(\Psi)\right) & \text{otherwise.} \end{cases} \tag{14.13}$$

$$\min\left(\sin\Phi\right) = \begin{cases} -1 & \text{if } \Delta\theta \leq \frac{2\pi}{k_2}, \text{ or } (\phi \mod 2\pi) < 3\pi/2 \\ & \text{and } (\Psi \mod 2\pi) > 3\pi/2 \\ \min\left(\sin\phi,\, \sin(\Psi)\right) & \text{otherwise} \end{cases} \tag{14.14}$$

where $\Psi = (k_2\Delta\theta + \phi)$ has been substituted for brevity. With this, the corresponding values for $r_{\min}$ and $r_{\max}$ can be determined, and the MP can be pruned with rule 14.9.

Based on the Distance Between a Minor Planet and the Spacecraft

The time-dependent problem described in section 11.4.1 becomes considerably more complicated in case one of the trajectories is an ExpoSin. In order to find the value for $\boldsymbol{r}_s$ for the ExpoSin at some point in time, the corresponding value for θ is needed – the calculation of which requires an integration of the time-of-flight Equation 12.7. If such an integration is done for ∼10 iterations per MP, for ∼400.000 minor planets, that implies performing 4 *million* such integrations. Needless to say, this is far too costly to be used as a simple pruning

function. An important realization here is that the fully optimized, second-order trajectory will differ significantly from an ExpoSin in the first-order analysis, unlike the conic sections encountered in the high-thrust analysis. For this reason it is not so useful to find the *exact* time and distance at the moment of closest approach between an ExpoSin and an MP.

In order to reduce the necessary computations, the distance curve will be approximated with a simple cubic splines interpolation, around the epoch the minimum distance is likely to occur (see Figure 14.1). To construct this interpolation, the exact positions at certain values of the ExpoSin's polar coordinate θ are calculated, and the corresponding times are found using the time-of-flight integral from Equation 12.7. The position of the MP is then calculated at these times using Kepler's equation, so that the exact distances are known at these epochs. The epoch of the minimum distance, and the smaller of the two adjacent points are then found. In this (much smaller) interval the process is repeated for an additional amount of points, after which cubic splines are fit through the resulting distance curve. A rough approximation to the minimum distance can then readily be determined, simply by taking derivatives of the cubic splines and solving the resulting quadratic equations.

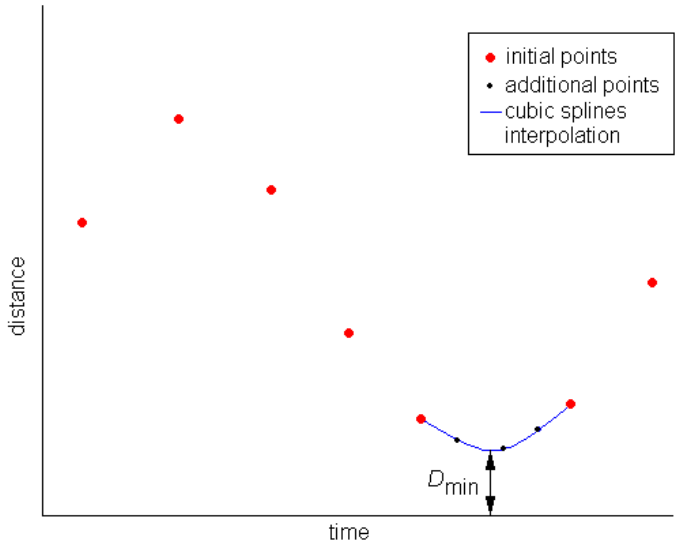


Figure 14.1 Method used to determine the minimum distance from an MP to an ExpoSin trajectory. First, a small number of positions is calculated for the two bodies, and the epochs of the minimum and second smallest distance are retrieved. In this much smaller interval, some more positions are calculated, and the distance curve is approximated with a cubic spline interpolation to determine the minimum distance.

There are some obvious failure modes with this method. It is possible that the quadratic derivatives never pass through zero in their respective intervals. This can mean two things: the interval between the minimum distance and the selected adjacent point (the one to the right of the minimum in Figure 14.1) does not contain the minimum distance, or the epoch of the minimum lies outside the interval of interest. The latter case is most easily remedied: if the initial minimum distance does not have another adjacent point (e.g., it is either the first or the last point in Figure 14.1), and the selected interval does not contain a minimum,

the initial minimum distance *is* the overall minimum distance. In case the other failure mode occurs, it simply means the interval is badly chosen. In that case, the process is simply repeated using the other adjacent point. If this interval also does not contain the minimum, the initial minimum distance will serve as the approximation to the true minimum distance D.

Some numerical experimentation indicated that this method works quite well. It was found that if both trajectories are in the same direction (e.g, both are prograde), using 5 to 10 initial positions suffices to find a reasonable approximation to $D_{\min}$ (depending a bit on the length of the interval). If the direction of one trajectory is reversed with respect to the other, it is necessary to use 25 to 30 points to make the method robust enough. Note that since the cubic splines only roughly approximate the true positions, it is necessary to use a larger threshold distance to decide whether to include the MP or not. Therefore, the threshold distance $D_{\text{threshold}}$ will be set to a (constant) value of 0.15 AU. If the (approximate) time-dependent minimum distance found this way is larger than $D_{\text{threshold}}$, the MP is pruned from the set. Otherwise, it will be considered a potential target in second-order analyses.

Part V

Final Design

15 First-Order Results

15.1. General Considerations

15.1.1 Parameters For The Optimizers

Global optimizers always need to be given the problem's lower and upper bounds (LB and UB), a population size *popsize* and some value for its control variables. Instead of passing these parameters manually at each new optimization it is far more useful to use well-chosen default settings for all these parameters, so that the users of Skipping Stone can focus on the *problem* rather than on the methods used to solve it. Throughout this thesis many such defaults have already been discussed; see for example Table 11.1. Automatization was also part of the intention when writing the GODLIKE optimizer; indeed, in chapter 7 it was found that GODLIKE's performance is relatively insensitive to the settings of its control parameters, so it is already a partial success in that respect. What remains to be discussed are the default settings for the number of allowable FE's, and appropriate population sizes.

Many authors normally use fairly huge population sizes for all problems to ensure that the optimum returned is indeed global. Using large (but fixed) population sizes indeed gives a large diversity for many cases, but naturally, that diversity is *not* constant for all problems; using a fixed population size on a 20-dimensional problem has roughly *half* the diversity of a similar 10-dimensional problem. Therefore, at the very least, *popsize* should depend on the problem's dimensionality to ensure a (more or less) *constant diversity* in all problems. Also, large population sizes are not always beneficial – large values for *popsize* also means large amounts of FE's per iteration, while for many global optimizers the rate of convergence depends only sub-linearly on *popsize*. In fact, [Izzo et al., 2007] successfully located the global optima in a vast 6-dimensional search space using a population of only 20 indi-

viduals[1]. Moreover, for many problems (including those encountered in this research) the intention of the global optimizer is *not* to give the absolute best possible result, but rather, to provide a good enough initial estimate for a *local* optimizer so that the final solution is *feasible*.

Therefore all single-objective global optimizations will be carried out with the following settings:

- Optimizations of a large number of different cases:

  High-thrust:

  - $popsize = 100\times$ the number of free variables in the decision vector X,
  - $\mathrm{FE}_{\max} \equiv 250,000$ Lambert-problems.

  Low-thrust:

  - $popsize = 75\times$ the number of free variables in the decision vector X,
  - $\mathrm{FE}_{\max} \equiv 150,000$ Lambert-problems.

- Optimizations on smaller sets for improved quality:

  High-thrust:

  - $popsize = 250\times$ the number of free variables in the decision vector X,
  - $\mathrm{FE}_{\max} \equiv 500,000$ Lambert-problems.

  Low-thrust:

  - $popsize = 175\times$ the number of free variables in the decision vector X,
  - $\mathrm{FE}_{\max} \equiv 250,000$ Lambert-problems.

MOO-problems will be carried with the following settings:

High-thrust:

- $popsize = 2500$,
- $\mathrm{FE}_{\max} = \infty$.

Low-thrust:

- $popsize = 1500$,
- $\mathrm{FE}_{\max} = 500,000$.

Larger population sizes are not only permissible for MOO (they will only be done for a few of the most promising solutions), but *necessary* – convergence in MOO is only achieved when *all* members of the population are non-dominated, and the quality of the resulting Pareto-front

[1]This was of course *after* pruning away 90% of that search space, which was the subject of that paper. However, the point is that the remaining search space was still far too large to expect 20 individuals to successfully locate the global solution.

strongly depends on the number of individuals used[2]. Experience also learns that Pareto fronts generated with such large numbers of individuals are much more stable and consistent, in the sense that separate runs nearly always converge onto the *same* result. Nevertheless, all fronts will be generated at least twice to verify whether this condition indeed holds.

The latter constraint of $FE_{\max} = 500,000$ for low-thrust MOO's is necessary to keep computation times reasonable. Preliminary testing showed that Pareto fronts for high-thrust problems could be generated in less than 5 minutes, while for low-thrust problems a full 8 *hours* was not sufficient to generate even *one* non-violated front with $FE_{\max} = \infty$ (the fraction of the population that *was* non-dominated after these 8 hours still showed much room for improvement). Imposing the limit $FE_{\max} = 500,000$ unfortunately can not guarantee all front members are indeed non-violated and non-dominated, but it does provide at least *some* insight into the relationship between the different objectives. In this chapter, all low-thrust problems are optimized using the MPE, which provides only rough estimates of low-thrust trajectories to the SBS; most likely still good enough to use as initial estimates in the next chapter.

One issue should be mentioned already at this stage – extensive preliminary testing showed that the MS optimization method elaborated in section 7.1 was considerably more difficult to "stabilize" in the context of trajectory optimization. Quite frequently it consumed more than four times the required amount of computation time compared to GODLIKE, and especially for problems of higher dimensionality (more than 2 GAM's) it rarely gave feasible results. In fact, most results were exceptionally poor compared to those found by GODLIKE on similar problems. In light of the enormous amount of optimizations that has to be performed (as discussed shortly), this optimization method will not be used for the actual optimizations even though it demonstrated very promising characteristics in chapter 7. Making this method more suited for trajectory optimization problems will be left here as a recommendation.

15.1.2 Selecting GAM-Sequences

In all of the previous chapters it was shown how a given sequence `seq` of GAM-bodies can be used in the optimization of a trajectory to the SBS. Unfortunately, there is no method to date that optimizes the sequence itself. Therefore, finding the best sequence can only be done with *brute force* – i.e., just optimize them all. However, the number of possible sequences for any number of GAM's quickly runs in the hundreds or even thousands of possible sequences. Such an amount of sequences is simply too large to test in full.

Therefore the length of the list of possible sequences will first have to be reduced. As discussed in section 3.4.2, Neptune is simply in the wrong corner of the Solar system during this mission's lifetime, so Neptune can safely be excluded from consideration altogether. When only looking at Figure 9.6 it seems quite reasonable to exclude Mars, Mercury and Uranus as well, because their possible energy gain only becomes attractive for large ΔV applied at large V_∞, which seems not a very likely scenario. However, considering the option of using SF's,

[2]It is easy to get 10 individuals non-dominated, but that does not guarantee they actually belong to the *true* Pareto-front; only in the more difficult case of finding 2500 non-dominated individuals can this be concluded with some degree of certainty.

it seems quite reasonable to include sequences like Earth-Jupiter-Mercury-Sun-SBS, because after a GAM at Jupiter it can be expected that the subsequent V_∞ at Mercury will indeed be very high which would enable it to facilitate the insertion for the SF with only a modest ΔV. Moreover, since retrograde solutions are allowed, it is possible that the V_∞ at any planet can exceed *twice* the planet's orbital speed, giving access to the more energetic parts of Figure 9.6. Also, GAM's can also have a different function than just gaining energy. For example, the location of the SBS's stagnation point also has a significant out-of-plane component, which normally requires large amounts of propellant to accomplish but can come "for free" from a GAM.

In short, no convincing argument can be found to exclude any planet other than Neptune from the set. The number of cases to test will necessarily be quite large. But considering the 25 year constraint on the time of flight, it seems reasonable to uphold a maximum of 3 GAM's total. Also, since Mercury requires a large ΔV to become an attractive swingby-planet (see Figure 9.6), Mercury will be included solely to facilitate reaching other planets (or the Sun) rather than for energy gain; any energy gained is a mere bonus. Therefore it will be included only *once* in the list of possible swingby bodies. A similar reasoning also holds for Mars – it too will be included only once in the set. Also, performing multiple swingby's with the planets Jupiter, Saturn and Uranus is not likely to be possible in the 25 year maximum, so these planets will also be included only *once* in the sequence. In summary, the planets/bodies that can be used for GAM's are

$$[\text{Sun, Mercury, Venus, Venus, Earth, Earth, Mars, Jupiter, Saturn, Uranus}]\,.$$

The sum of all unique combinations with these bodies for 1, 2 and 3 GAM's is equal to 444. Of course there are quite a few combinations among these 444 that will either require too long a travel time (e.g., Earth-Mercury-Uranus-Sun-SBS), are not possible with today's launch vehicles (e.g., Earth-Neptune-SBS) or are just very unpromising (e.g., Earth-Jupiter-Uranus-Saturn-SBS). Also, it is possible to give a back-of-the-envelope calculation of a lower estimate on the required orbital energy. To cover the 230 AU distance in a maximum of 25 years implies an average speed of 230 AU/25 years $\approx$ 43.7 km/s. Assuming $V_\infty \approx 45$ km/s and $R_{\text{SBS}} = 230$ AU, the required orbital energy is approximately

$$\frac{V^2}{2} - \frac{\mu}{r} = \epsilon_{\text{req}}, \tag{15.1}$$

$$\Rightarrow \epsilon_{\text{req}} \approx \frac{(45\text{ km/s})^2 + \frac{2\mu}{230\text{ AU}}}{2} - \frac{\mu}{230\text{ AU}} = 1000\text{ km}^2/\text{s}^2. \tag{15.2}$$

In addition, Tsiolkovskii's equation with the assumed values (for high thrust) $I_{\text{sp}} = 300$ s, $m_{\text{wet}} = 3000$ kg and $m_{\text{dry}} = 50$ kg gives $\Delta V_{\max} \approx 12$ km/s. Loosely assuming that energies from Figure 9.6 can simply be added together, these numbers also justify dismissing sequences containing only inner-planets and/or Uranus. Using these criteria to prune out all such combinations leaves a somewhat more manageable 146 `seq`s that must be tried. A complete list of all combinations is given in Table 15.1. The GAM-body names are hereby abbreviated as `Su` = Sun, `Me` = Mercury, `V` = Venus, `E` = Earth, `Ma` = Mars, `J` = Jupiter, `Sa` = Saturn, `U` = Uranus.

Table 15.1 List of all 146 sequences to be tested. The abbreviations are Su = Sun, Me = Mercury, V = Venus, E = Earth, Ma = Mars, J = Jupiter, Sa = Saturn, U = Uranus.

Su	J	SuMe	SuV	SuE	SuMa	SuJ	SuSa	SuU	MeSu
MeJ	MeSa	VSu	VJ	VSa	ESu	EJ	ESa	MaSu	MaJ
MaSa	JSu	JSa	JU	SuMeV	SuMeE	SuMeMa	SuMeJ	SuMeSa	SuMeU
SuVV	SuVE	SuVMa	SuVJ	SuVSa	SuVU	SuEMa	SuEJ	SuESa	SuEU
SuMaJ	SuMaSa	SuMaU	SuJSa	SuJU	SuSaU	MeSuV	MeSuE	MeSuMa	MeSuJ
MeSuSa	MeSuU	MeVSu	MeVJ	MeVSa	MeESu	MeEJ	MeESa	MeMaSu	MeMaJ
MeMaSa	MeJSu	MeJSa	MeJU	MeSaU	VSuMe	VSuV	VSuE	VSuMa	VSuJ
VSuSa	VSuU	VMeSu	VMeJ	VMeSa	VVSu	VVJ	VVSa	VESu	VEJ
VESa	VMaSu	VMaJ	VMaSa	VJSu	VJMe	VJSa	VJU	VSaU	ESuMe
ESuV	ESuE	ESuMa	ESuJ	ESuSa	ESuU	EMeSu	EMeJ	EMeSa	EVSu
EVJ	EVSa	EESu	EEJ	EESa	EMaSu	EMaJ	EMaSa	EJSu	EJMe
EJSa	EJU	ESaU	MaSuMe	MaSuV	MaSuE	MaSuJ	MaSuSa	MaSuU	MaMeSu
MaMeJ	MaMeSa	MaVSu	MaVJ	MaVSa	MaESu	MaEJ	MaESa	MaJSu	MaJMe
MaJSa	MaJU	MaSaU	JSuMe	JSuV	JSuE	JSuMa	JSuJ	JSuSa	JSuU
JMeV	JMeE	JMeMa	JMeSa	JMeU	JSaU				

As mentioned in chapter 12, a flyby relatively close to the Sun is indeed useful in a low-thrust context, but requires different methods than the high-thrust SF-method to tackle. In fact, when using the MPE, close Solar approaches can be modelled without altering the method altogether – setting $m > 0$ and/or an appropriate value for the k_2-parameter will generally accomplish the task. Therefore, to test the feasibility of a *low-thrust* mission to the SBS, sequences from Table 15.1 that do *not* contain the Sun must be tested with $m = 0$, and sequences that *do* contain the Sun must be tested with $m > 0$. Again, considering the constraint on the time of flight, it seems reasonable to uphold a limit of $m_{\max} = 1$ for such sequences.

15.1.3 Optimization Strategy

Generally speaking, high-thrust optimizations are significantly faster than low-thrust optimizations, due to the far more restricted search spaces and the much more demanding ExpoSin's Lambert targeter associated with low-thrust problems. Also, the effect of using a low-thrust system on the search space is to broaden and flatten the feasible regions in the corresponding high-thrust search space, so it can be expected that at least for the current mission, low-thrust optima lie fairly close to their high-thrust counterparts. Considering the fact that 146 possible `seq`s will have to be optimized several times each, it seems wasteful to optimize *all* these `seq`s for both high-thrust and low-thrust systems. So instead, the 146 combinations will *all* be tested assuming a high-thrust system, and only the best 10 results of this process will also be optimized assuming a low-thrust system.

For similar reasons it can be concluded that calculating the amount of nearby MP's in *all* 146 optimizations is not so useful – most `seq`s will not result in a trajectory that can reach the SBS with all constraints satisfied, which makes the associated MP-count much less interest-

ing. Moreover, although the calculation of MP's can be executed quite quickly, for most `seq`s it still implies a quadruplication of the required computation time (or more). For 146 combinations this simply takes too much time. Therefore the MP-count will only be computed for a few of the most promising solutions.

Because global optimization routines have a high *probability* of finding the globally optimal solution, yet can not *guarantee* that any solution found *is* the global optimum, global searches always need to be executed several times to increase the likelihood of finding the global solution. Optimizing each `seq` will therefore be done 5 times; once with GA, once with DE, once with PSO, once with SA and once with GODLIKE. This approach requires very reasonable computation times, and more importantly, provides two results at once; the globally best result of the sequence (with an acceptable probability), *and* a rough measure of how well each optimizer performs individually. Although each optimizer is allowed only *one* optimization per sequence, their combined results for 146 optimizations still provides a rough statistic on how their probabilities of finding the global optimum compare. Only for the best 5 solutions will each method be allowed to carry out 5 optimizations, improving the quality of this statistic on a per-`seq` basis.

15.2. Scenario 1 – Bow Shock Only

15.2.1 High Thrust

All 146 results

Fortunately, in Skipping Stone's current form, it is fairly trivial to (at least partially) automatize the process of testing all 146 cases. This means all sequences that do not include an SF could be tested in less than 2 days. An SF in its current form is however far more computationally expensive, so optimizing all those sequences took about one full week (including extensive manual tweaking). The best result of all 5 optimizations performed on each of the 146 `seq`s was used to construct Figure 15.1. This figure lists the (ordered) best 50 outcomes; all the other sequences required more ΔV than the maximum entry in this figure. A complete list of *all* results from this process can be found in appendix G.

Algorithm Performance

It is fairly straightforward to take the minimum of the 5 optimizations performed for all 146 sequences, and keep track of which optimizer found that particular optimum. This gives a "algorithm hitlist" as shown in Figure 15.2. Note that this figure only *roughly* represents the relative probability of each optimizer to find the global optimum during the *first* evaluation; it should *not* be interpreted as a reliable source to compare the true performance of these optimizers in trajectory optimization.

It *does* however indicate that, despite the expectations formed in chapter 7, the PSO-algorithm has a relatively poor chance of finding the global optimum in one go. This result agrees with the results found in many preliminary tests and experiments not mentioned in this thesis; PSO in a trajectory-optimization context is generally quite sensitive to the settings

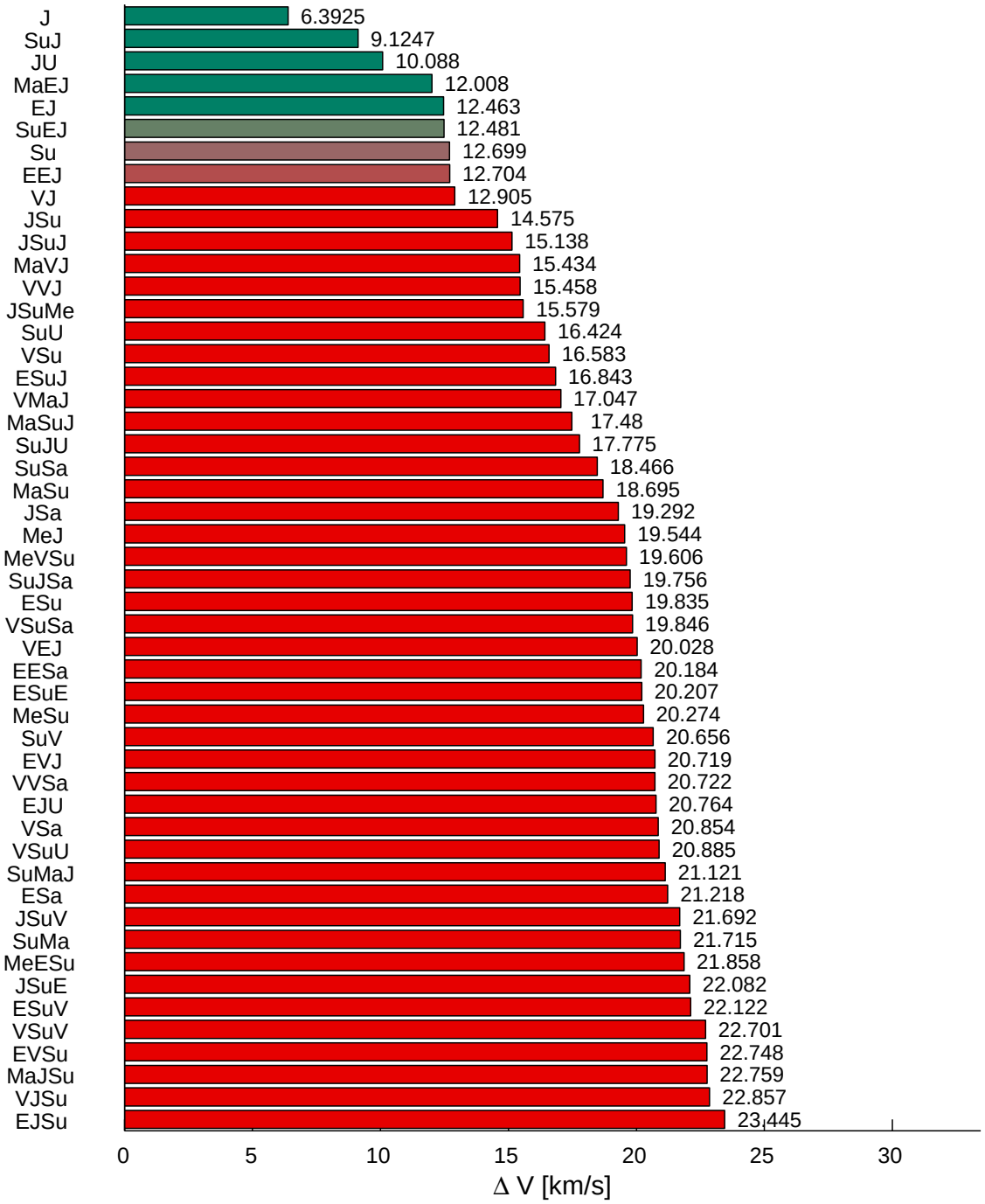


Figure 15.1 Results found for the best 50 sequences. As expected, there are only very few solutions that require a ΔV below 12 km/s. (Near) feasible solutions are indicated with green, infeasible solutions with red. The times of flight of all these results converged to the maximum allowed value of ~ 25 years. The abbreviations are `Su` = Sun, `Me` = Mercury, `V` = Venus, `E` = Earth, `Ma` = Mars, `J` = Jupiter, `Sa` = Saturn, `U` = Uranus.

of its control variables. Each new problem requires considerable manual tweaking before its performance supersedes that of SA/DE. Of course, doing so for all 146 sequences was considered to be too much work for this thesis, so the "optimum" settings found in chapter 7 were always used – this possibly explains its poor performance.

Also against the expectations is the high probability for the GA to find the solution on first evaluation; it found nearly 1/3 of all solutions. If anything, this figure proves that the jump from test-functions to compare the strength of global optimization algorithms, to their actual application (trajectory optimization in this case) is too large to allow transferring the conclusions regarding their performance as well.

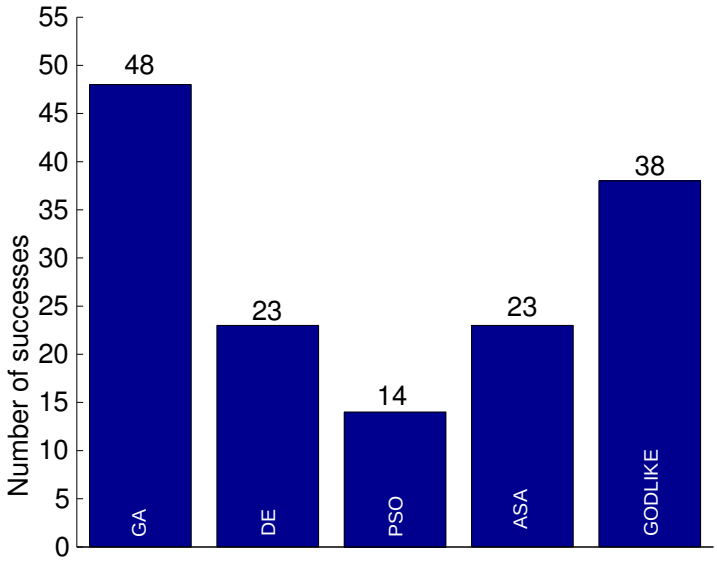


Figure 15.2 Rough measure on the performance of each algorithm thus far. These numbers represent the total number of times a particular algorithm located the global minimum during a specific optimization. Note that these numbers do not accurately represent each algorithm's true performance, as these numbers have a poor statistical basis and do not consider any context or comparison with each other.

Improving the Best 10

The best 10 sequences were optimized again with all 5 algorithms and 5 times each, but this time with the higher settings for the population and permissible FE as described in section 15.1.1. For the sequences `E-J-`SBS and `E-J-U-`SBS this proved hardly necessary, but for the other sequences involving an SF it was certainly worthwhile – the optima in search spaces involving an SF seem to be quite hard to find.

A considerable amount of effort was put in examining the sequences `J` and `JU` more closely; their initial results imply that the SBS can be reached entirely by "classical" means, and moreover, with a relatively low ΔV. These `seq`s would involve relatively little risk compared

to an SF, and especially the latter `seq` would also increase the mission's scientific output – the spacecraft will perform a GAM at Uranus, a planet that was visited *once* in the history of spaceflight (not to mention with a 30-year old spacecraft).

Earth-Jupiter-SBS This set of 5×5 optimizations was carried out while imposing an upper limit on the ΔV of 0 km/s. This strongly forces the optimizers to push the ΔV as far down as possible in the given 25-year search space. That is, the optimizers will not stop until their *whole* population has converged onto a single, unconstrained optimum. A subsequent local-SQP-Quasi-Newton minimizer then takes care of possible minor improvements. This test showed that the minimum ΔV necessary to reach the SBS in under 25 years, with an I_{sp} of 300 s and a minimum approach altitude to Jupiter of 42,895 km is 6.39 km/s (the actual global minimum of this sequence was found during the 146 tests). Although this is quite high, it is well within the capabilities of modern propulsion technology. Also, the associated C_3-value at the launch conditions of this optimum was 129.56 km^2/s^2; this is *definitely* within the range of modern high-capacity launchers.

Naturally, solutions thus found strongly depend on the minimum allowable approach distance to Jupiter. Because of Jupiter's extremely harsh radiation environment, this minimum allowable altitude was fixed to 42,895 km during all optimizations (see section A.4.1). However, justifications similar to those made in chapter 10 can be made here; the spacecraft's exposure to this environment would be of sufficiently short duration that the benefit of a *closer* approach could very well outweigh the drawback of including additional mass for radiation hardening. With this in mind, a relationship between the approach distance, ΔV and t_f can be constructed; this has been done in Figure 15.3. To construct this figure, the rightmost points (those at 40,000 km) were computed first by taking the best result from all 5 optimizers for the associated problem. Subsequent lower altitudes were then computed by inserting the previous best result in a local SQP-Quasi-Newton optimizer. As can be concluded from this figure, it is not possible to reach the SBS in less than 20 years using the `J-seq`, even with unrealistically close approach distances of 5,000 km.

Naturally, it is very worthwhile to get a more accurate idea on the functional relationship between the maximum m_{dry} and the minimum t_f. To that end, four Pareto fronts were created for this sequence, each with a different setting for the minimum allowable Jovian altitude. The results are shown in Figure 15.4. For this figure the population size was fixed at 2500 individuals and no limit was imposed on the number of FE's; GODLIKE was allowed to continue until *all* individuals were non-dominated and feasible.

It can be concluded by comparing Figure 15.4 with Figure 15.3 that the two figures *disagree*. Although they show the same trend, their absolute values are very different. This can indicate that the local optimizations used to generate Figure 15.3 do not always yield the globally optimal solution, even if its first initial estimate (the rightmost points) are global solutions. It can also indicate that the rightmost points were in fact *not* globally optimal, or that GODLIKE's MOO is more efficient in locating globally optimal solutions than any of the individual optimizers in single-objective mode. In any case, the results indicated by the

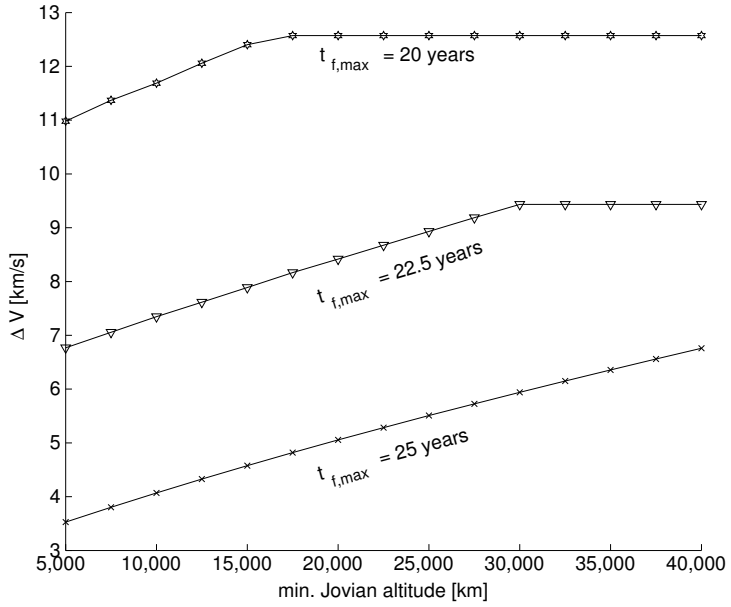


Figure 15.3 Relationship between t_f, $h_{\min}$ and ΔV for the seq J. These results show it is technically impossible (>11 km/s) to push the required time of flight down to 20 years. These results are computed by taking a global solution at the rightmost points, and optimizing these locally at all subsequent points at lower altitudes.

Pareto front are much better, so only the Pareto fronts are generated for subsequent similar analyses.

Earth-Jupiter-Uranus-SBS This sequence was again optimized 5×5 times using the original minimum permissible altitudes of 42,800 km and 81,533 km at Jupiter and Uranus, respectively. The best result found showed that this sequence requires a minimum of $\Delta V = 10.09$ km/s to reach the SBS in under 25 years (again, the global optimum was located during the 146 trials). The accompanying C_3-value at launch for this optimum is 128.04 km^{2}s^{-2}; again well within the reach of today's launch vehicles.

The minimum allowable distance to Uranus was kept at 81,533 km during all optimizations; a value that follows from the Voyager 2 flyby. The Voyager 2 mission designers used this large distance for three good reasons: a closer approach was not *strictly necessary* to reach Neptune, but would involve additional risk because it was entirely unknown at the time whether Uranus had a *magnetic field* (and what its strength was), and whether Uranus had a *ring system* (and what its exact extent was).

Now it is known that unlike Jupiter, Uranus' magnetic field does not give rise to the same "off-the-scale" levels of radiation and need not be considered a problem. However, its ring-system does put strict limits on the possible approach and departure directions if a spacecraft is to perform a close proximity flyby. The majority of the ring material is located between 41,000 and 52,000 km from the planet's center [de Pater and Lissauer, 2005, Par. 11.3], which implies that if the spacecraft is to approach the planet at altitudes between 15,000 – 26,000 km, a

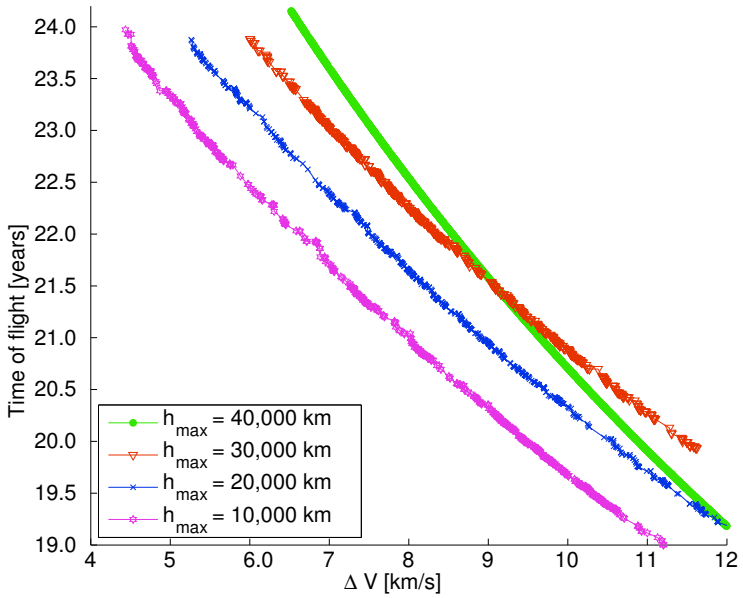
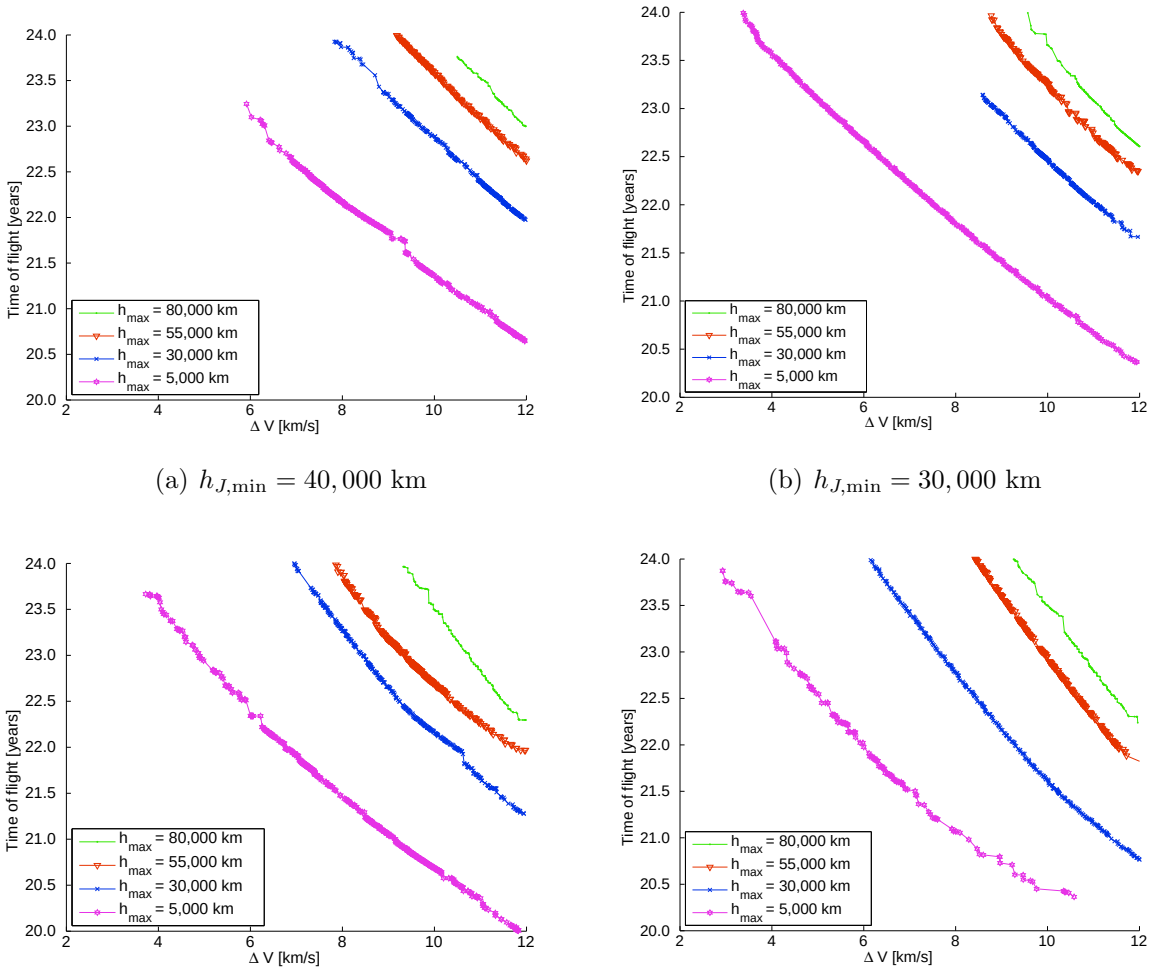


Figure 15.4 Pareto front (J), high-thrust. This plot is similar to Figure 15.3 but now computed with a *global* MOO routine – these results are indeed much better. The trend is the same though for both figures – irrespective of the minimum altitude, a time of flight lower than 20 years seems impossible to accomplish.

very careful analysis needs to be performed on whether the spacecraft will have to fly through the ring plane or not. From an orbital evolution and decay point of view it is however quite unlikely that ring particles can stay in orbits below ∼5,000 km altitude for longer periods of time, so that it can be expected that the risk of crossing the ring plane might actually not apply to such low altitudes. But even this does not negate the need for a more careful analysis. Such an analysis is quite laborious and depends on many factors, so for now the presence of the ring system is entirely ignored and will only be taken into consideration again if the final result requires a minimum altitude in the mentioned range.

To get a feel for how reducing the minimum allowable approach distance affects the time of flight and required ΔV, Pareto fronts similar to Figure 15.4 were generated while varying minimum allowable altitude from Uranus, in steps of 25,000 km. In addition, also the Jovian altitude was varied, in steps of 10,000 km. This gives the four sets of fronts shown in Figure 15.5. As these fronts clearly show, decreasing the altitude seems to push all fronts to the origin at an almost *quadratic* rate. Especially the Jovian approach distance of 20,000 km and Uranian approach distance of 5,000 km are a very appealing result; they imply the SBS can be reached in $\sim$ 24 years using a ΔV of only $\sim$ 3 km/s.

Remaining 8 sequences None of the 5×5 optimizations carried out on these sequences improved the original solution very much, *except* for the SuJ-sequence. In most optimized results this sequence used a ΔV of ∼0 km/s at a Sun-pericenter distance slightly less than Earth's radial position at launch, in other words: it converged onto the J-sequence. It was



(a) $h_{J,\min} = 40,000$ km

(b) $h_{J,\min} = 30,000$ km

(c) $h_{J,\min} = 20,000$ km

(d) $h_{J,\min} = 10,000$ km

Figure 15.5 Four Pareto fronts (JU), generated with different settings for the minimum allowable altitudes for both Uranus and Jupiter. Note that minimum altitudes at Uranian between 15,000 – 26,000 km might cross Uranus' ring plane. Corresponding solutions should be left out of consideration, or at least be subjected to a more careful analysis.

found to be less efficient to perform a ΔV manoeuvre near the Sun than at Jupiter. But in *some* optimizations the ΔV at pericenter was ~ 5 km/s, and the ΔV at Jupiter ~ 0.5 km/s, at an altitude of $\sim 100,000$ km; a *very* appealing result that is definitely worth much more attention.

Naturally, the other 7 sequences will not be used in any subsequent analyses. For completeness, Pareto fronts were generated for all 8 remaining sequences, which are shown in Figure 15.6. Note that all individuals on all these fronts are actually *infeasible* – to generate any front at all, the maximum ΔV needed to be taken limitless. Note that the Pareto-front for the SuJ-sequence shows some "odd jumps" to lower values of t_f. At first these seemed to be mere artifacts of how the plot was created; because the ΔV needed to be set limitless, also *infeasible* members had to be plotted in the Pareto front, which also showed the members that

violated the minimum pericenter distance. A careful analysis of all the associated solutions showed that their constraint violation was actually not that severe (at least for most of them). This indicates that the front shown is actually "semi-converged"; most solutions converged onto the J-front, e.g., with a ΔV of $\sim$ 0 km/s at the Su-pericenter. The "spike"-solutions used a very different launch date and minimum approach distance at Jupiter and a ΔV at pericenter far from 0 km/s, consistent with the 5×5 single-objective results found earlier for this sequence.

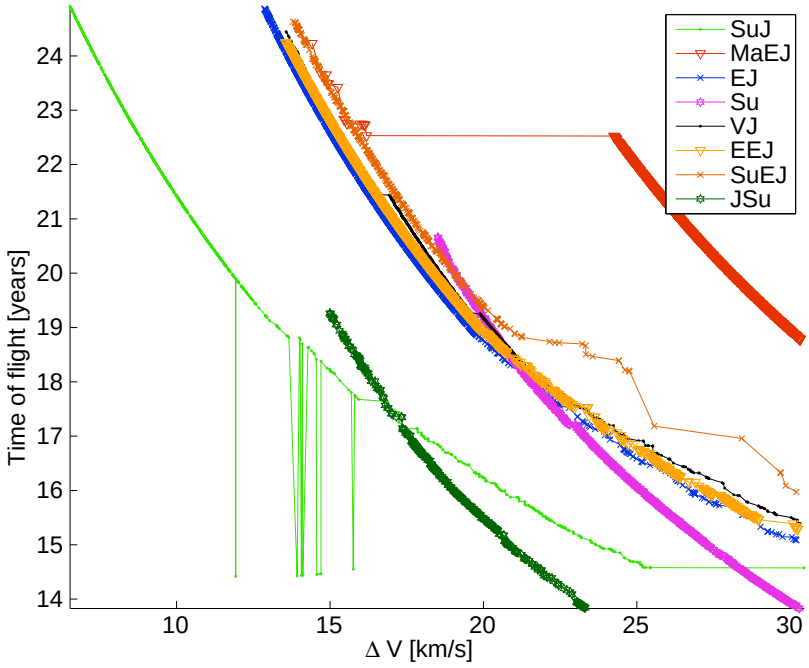


Figure 15.6 Pareto fronts for the remaining 8 best sequences. Except for the SuJ-sequence, none of these fronts contain any feasible solutions. The "spikes" in the front for the SuJ-sequence are "near misses", in the sense that they violate the constraint on the minimum distance to the Sun, but only mildly so.

Unfortunately, all further attempts to generate the *true* Pareto front for this sequence failed. Front generation was attempted with different settings for the maximum allowable ΔV at the Jovian pericenter (stepping from 0 km/s to 5 km/s), but all these optimizations failed to find more than 1 non-dominated solution after 500,000 FE. Only when this constraint was set to values larger than 6 km/s did the front converge onto the front shown in Figure 15.6 (with spikes included). This is quite odd, as single-objective optimizations following a similar tactic often converged onto solutions similar to the ones previously found. Most likely this indicates that an implementation error still resides *somewhere* in the SF-algorithm, or in the interface between the SF-algorithm and the MOO-algorithm. This is quite unfortunate because this sequence *does* give the best results thus far – the large Solar-pericenter and Jovian-pericenter distances involve almost no risk compared to the closer approaches to Jupiter/Uranus discussed previously, and the total ΔV of 6.29 km/s is actually 100 m/s *lower*

than the J-sequence. For now this issue has to be left here as future work, and the sequence will be explored further exclusively via single-objective optimizations.

Improving the Best 3

In all trials thus far, the last leg was always assumed to end at the (approximate) location of the SBS stagnation point (see Equation 2.1). Not only is the location of this point prone to some degree of uncertainty, it might also not actually be necessary to go to this *exact* point (see section 2.1.2). The SBS encloses roughly half the Solar system, so that travelling to a point close to the stagnation point will still allow the spacecraft to penetrate the shock front. The stagnation point was used both to simplify the optimizations and to attempt to reduce the distance to be covered as far as possible – a shock's stagnation point is (usually) the point closest to the body causing the shock. But it is easy to imagine that *different* points close to the stagnation point do not increase the distance consdierably, but do allow a greater degree of flexibility at the last swingby.

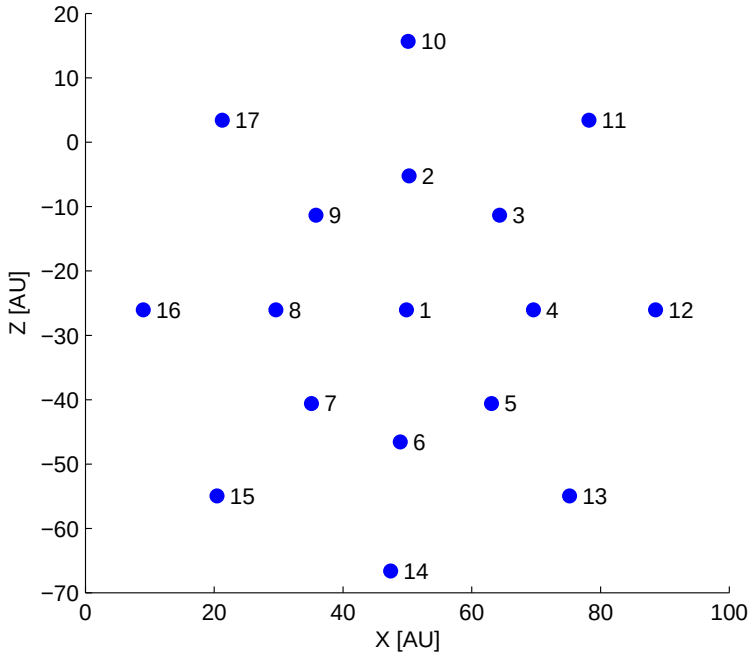


Figure 15.7 Locations of the 16 other points to be tested, projected onto the XZ-plane. These points all surround the stagnation point (point 1) and are separated 6° and 12° from the stagnation point, in the ecliptic longitude and latitude directions and their diagonals.

All 16 points To that end, the J, JU and SuJ sequences were tested again, but this time for 16 other points surrounding the original stagnation point. These 16 points were taken 6° and 12° removed from the stagnation point (which is point 1 in Figure 15.7), along the λ and β directions, and the directions diagonal to these. Each point was again optimized 5×5 times, upholding the aforementioned $\mathrm{FE}_{\max} = 500,000$ for every single optimization. The minimum approach distances at Jupiter and Uranus were of course reset to their original

values of 42,895 km and 81,533 km, respectively.

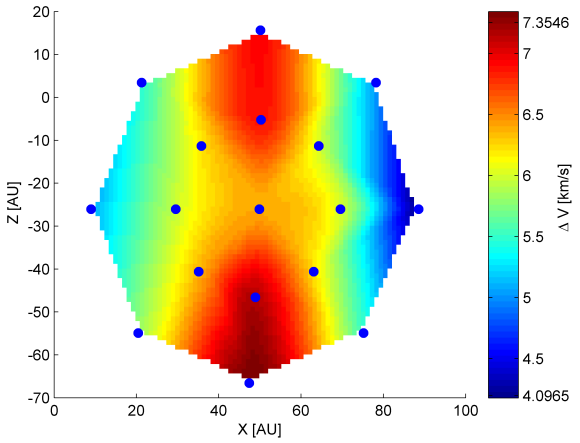
The results can be interpreted most conveniently by interpolating a surface through the 16 points in the XZ-plane, taking the required ΔV for each point as the plane's third coordinate. These surfaces are shown in Figure 15.8. The raw data on all these optimizations are listed in appendix G.3 for completeness.

These figures clearly indicate that shifting the target point away from the stagnation point *drastically* improves the quality of the solutions. Point 12 (the rightmost point) improves the ΔV required for the `J-seq` to 4.09 km/s, which means that travelling to the stagnation point (point 1) inherently requires too large a turn angle at Jupiter. The `JU`-sequence is improved most by both by shifting the target point to the right along the X-axis *and* reducing the out-of-plane component to practically zero – point 11 pushed the required ΔV for this sequence down to 4.954 km/s. It appears the the `JU-seq` is much more sensitive to the out-of-plane component than the `J-seq`, in addition to the required turn angle at Uranus. Note that for all solutions embedded in these surfaces, the time of flight was always between 23 and 25 years.

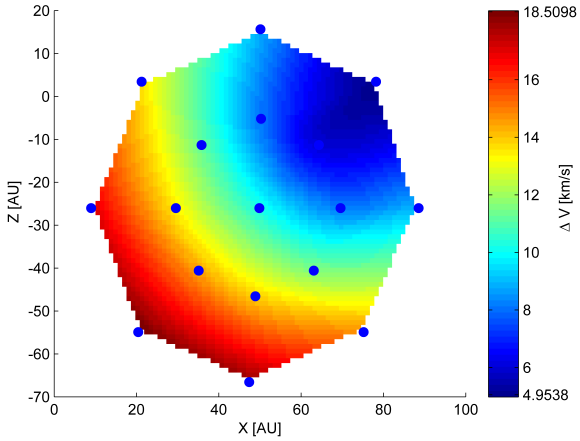
Note that the `SuJ`-surface shown above is remarkably similar to that of the `J-seq`; in fact, save for two different results, it is precisely the same. This is again due to the fact that solutions using this sequence practically always prefer the `J-seq` over a DSM just after launch. This is clearly shown in the ΔV required for the corresponding best point – *exactly* the same as for the `J-seq`. Further analysis of this sequence (repeated single-objective optimizations) revealed that the latter type of solution (DSM near the launch epoch) also demands a very large C_3 – in none of the ~ 40 manual trials did the C_3 ever fall below 200 km^2s^{-2}. In this light it is indeed to be expected that the results previously found for this sequence are better than the `J-seq` alone; the high C_3 in combination with a large ΔV close to Earth is essentially equal to a much higher C_3 at launch, causing a much higher V_∞ at Jupiter and thus a higher efficiency of the ΔV applied at the Jovian pericenter. But unfortunately, upholding the more realistic constraint of $C_3^{\max} = 157$ km^2s^{-2} (New Horizons' value) proved incapable of generating solutions significantly better than the `J-seq` alone, as could be expected when looking at Equation 3.3. Considering this fact, and considering the problems previously encountered with Pareto front generation for this sequence, it therefore seems best to abandon all further analysis of this sequence and focus on improving the other two sequences.

An obvious question that arises when looking at Figure 15.8 is whether solutions can be improved further by analyzing points that are *even further* out than 12° along the most promising directions. Although this might indeed be true, the 12° used here is already quite far removed from the expected location of the stagnation point. Taking points even further out would require a much more in-depth analysis on the expected shape of the shock front, to determine how such an additional deviation would impact the expected distance to be covered. Naturally, this is out of the scope of this research and will therefore be left here as future work. For now, only the best points found here will be analyzed further.

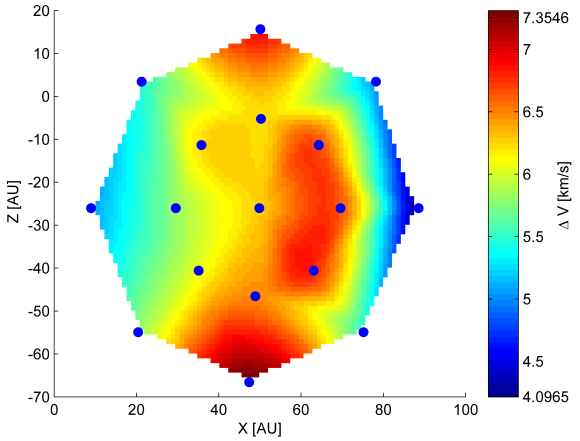
Analysis of the best points Until now, all optimizations have been carried out using the unrealistically high value $C_3^{\max} = 225$ km^2s^{-2} as upper limit (see section 3.4.3). To see whether solutions found so far can also satisfy the stricter demand of $C_3^{\max} = 157$ km^2s^{-2} (New Hori-



(a) Results for all 16 points, `J-seq`



(b) Results for all 16 points, `JU-seq`



(c) Results for all 16 points, `SuJ-seq`

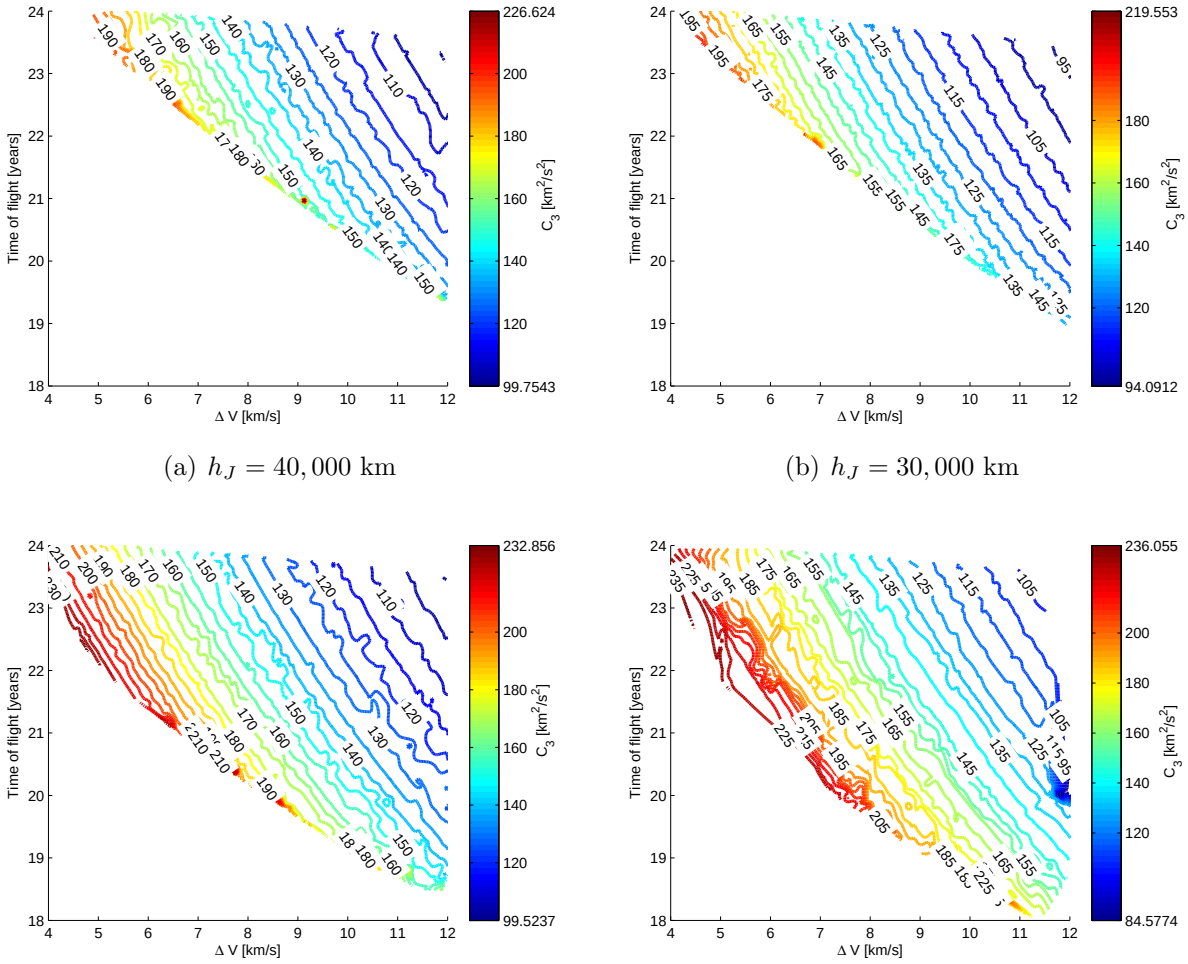
Figure 15.8 Results for testing all 16 points surrounding the stagnation point, for the `J`, `JU` and `SuJ` sequences. It should be clear that for these sequences the stagnation point is not the best choice as a target. Note that these surfaces are cubic interpolations through only 16 data points (the blue dots); the colored areas can thus only serve as an indication of the trend shown by these data points, and should not be interpreted as real data.

zons' value), the single best points found for both sequences (point 11 for JU, and point 12 for J) were analyzed further again by looking at both ΔV and t_f. The relationship between these quantities is quite complicated and must be examined by generating 3 dimensional $\Delta V/t_f/C_3$ Pareto fronts. Because an extra dimension is now needed, even more individuals are required to create fronts of acceptable quality. Using 7,500 individuals proved to be the highest setting that could still generate fronts of acceptable quality in under 15 minutes. The resulting fronts are quite difficult to interpret even when they are free to rotate in all three dimensions on a computer screen. So for clarity, a surface was again interpolated through the resulting Pareto front, which can best be displayed by means of its isolines in $\Delta V/t_f$-space. Unfortunately this technique creates some unwanted artifacts ("Amazone-shaped" isolines in stead of smooth curves), but it does provide the most convenient way to look at these data.

J sequence Taking point 12 as the new target, the minimum approach distance to Jupiter was varied from 40,000 km to 10,000 km with steps of 10,000 km as before. For each setting of the minimum altitude a 3 dimensional Pareto front was created, resulting in Figures 15.9. In the color bar belonging to each of these figures, the minimum and maximum values found for C_3 are also indicated. It should be obvious from these figures that even with the unrealistically close approach distance of 10,000 km the constraint $C_3^{\max} = 157$ km^{2}s^{-2} can not be met with a ΔV expenditure less than $\sim$6.5 km/s at $t_f \approx 24$ years.

Note that the isolines corresponding to the highest C_3 values seem to be "lost"; this is due to the interpolation routine to create the isolines. These extreme values are always strongly compressed against the lowest $\Delta V/t_f$-front, thus forcing the isoline plotting routine to skip them in favor of the isolines of lower value. The effect of reducing the minimum allowable approach distance seems to be to reduce this compression effect and expand and smooth the surface in these high-C_3 areas to lower regions in $\Delta V/t_f$-space, but *only* the higher C_3-regions; the lower C_3 areas appear completely unaffected by this setting. This is quite a beautiful way to show the correctness of the Oberth-effect; since $C_3 = V_\infty^2$, the governing equations depend *linearly* on C_3, but the V_∞ at Jupiter and thus the overall efficiency of the ΔV is affected *quadratically* but *inversely* proportional to the Jovian pericenter distance; which indeed gives the behavior shown.

JU sequence Note that the JU-seq showed strong preference towards a target with zero-degree deviation from the ecliptic plane. Although point 11 is *close* to the ecliptic, it still has a small non-zero Z-component. Therefore, the new target for this sequence is taken equal to point 11, but then with $Z = 0$ exactly. It is difficult to correctly display a large amount of 3 dimensional Pareto fronts for different settings for the minimum flyby distances at Uranus and Jupiter all in one figure, as was done before for this sequence. To still get an idea of the influence of these settings on the quality of the outcomes, only four different combinations were tested – the normal, "high" settings for the minimum allowable altitudes at both planets, one planet 'high" and the other "low", vice versa, and both "low". The results for these four combinations are shown in Figure 15.10. It appears that the results for $C_3 = 157$ km^{2}s^{-2} for this sequence are very similar to those found for the J-seq, again nearly independent of the allowed approach distances; about 6.5 km/s for $t_f \approx 24$ years. Given that these results are very similar, and that the scientific value of this sequence is superior to that of the J-seq, it seems best to use *this* sequence to generate the final results with.



(a) $h_J = 40,000$ km

(b) $h_J = 30,000$ km

(c) $h_J = 20,000$ km

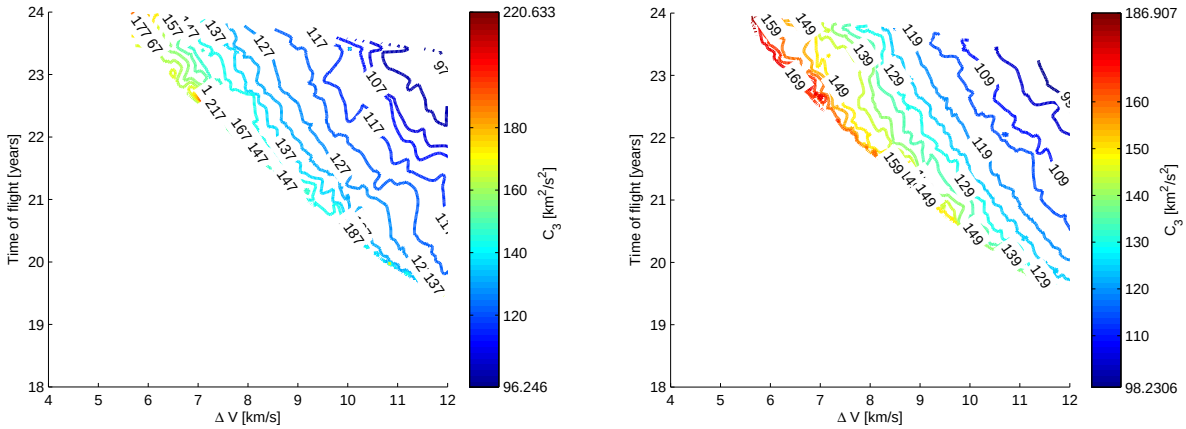
(d) $h_J = 10,000$ km

Figure 15.9 Analysis of the J−seq, taking point 12 as the new target.

15.2.2 Low Thrust

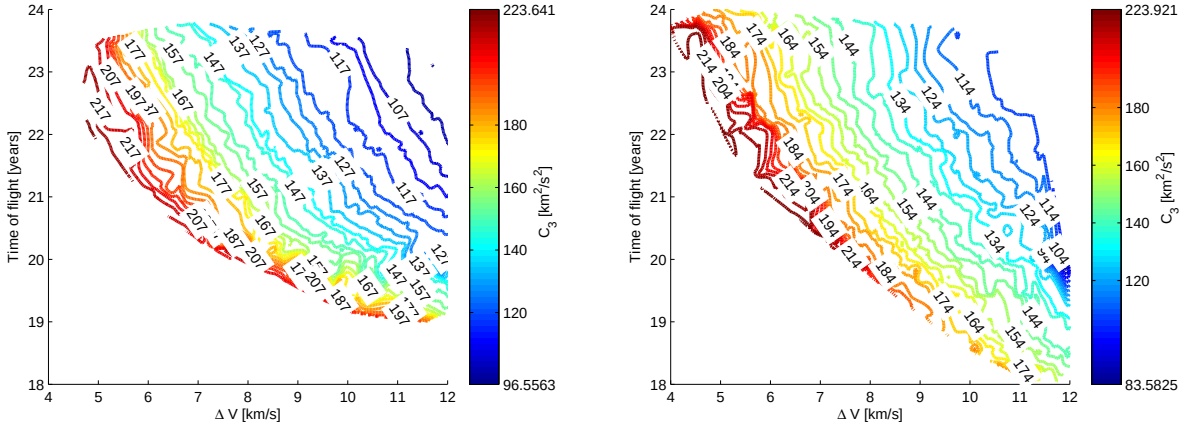
A preliminary grid-search indicated that there is almost no combination of $|\boldsymbol{r}_1| < 30$ AU, $t_f < 25$ years, $0.01 < k_2 < 1$ and $m = [0, 1]$ that yields a feasible ExpoSin-trajectory to the SBS in the required 25 years. Only very small and seemingly randomly distributed regions in the range for the k_2-parameter gave possible solutions, which means that ∼85% of the individuals failed to evaluate the patched conics procedure at each iteration. This is most likely due to the fact that the method of ExpoSin is unable to completely turn off the spacecraft's engine, that is, the ExpoSin equations can not degenerate into a purely ballistic conic section; only approximately so. This not only means trajectories to extreme distances are hard to find, it also implies that even though the vast majority of the leg to the SBS will be flown without thrusting, the ExpoSin's are unable to reflect this and consume propellant anyway. This is an imperfection inherent to the method and can not be negated.

This fact necessitates using a "work-around" for the low-thrust optimizations; the last leg (that to the SBS) will therefore have to be determined with the *high-thrust* Lambert-targeter.



(a) $h_J = 40,000$ km, $h_U = 80,000$ km

(b) $h_J = 40,000$ km, $h_U = 5,000$ km



(c) $h_J = 10,000$ km, $h_U = 80,000$ km

(d) $h_J = 10,000$ km, $h_U = 5,000$ km

Figure 15.10 Analysis of the JU-seq, taking point 11 (with 0° inclination) as the new target.

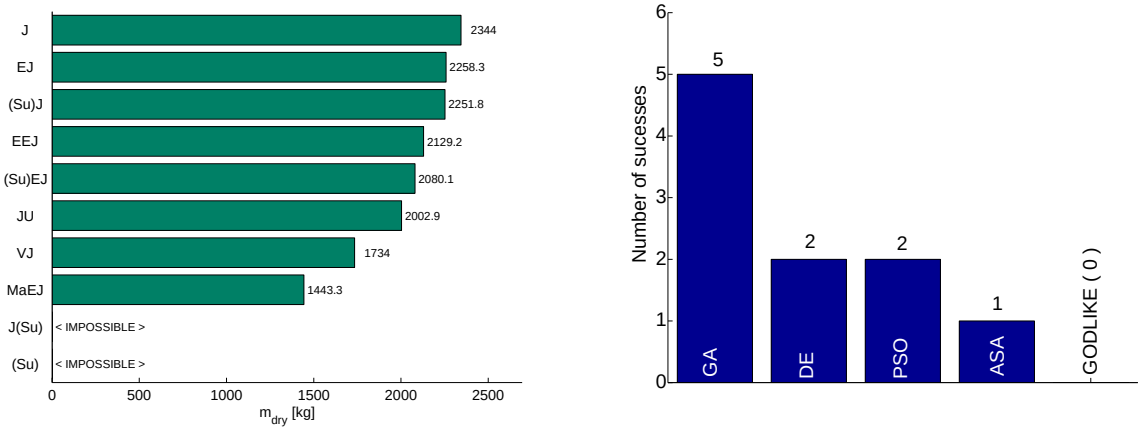
This seems quite reasonable when assuming SEP; such engines are rendered useless beyond Jupiter's orbit so that the spacecraft indeed flies a ballistic trajectory. However, it is *incorrect* when assuming NEP; engines powered by nuclear batteries can operate far beyond even Uranus' orbit. "Fortunately" the process of optimizing the seqs with the MPE only predicts the approximate locations of the basins of attraction, so in this light, it is probably sufficient to generate initial estimates with a ballistic flight from the last swingby to the SBS – a more accurate (second-order) optimization that assumes NEP will most likely still converge to more realistic trajectories.

Further testing (with this correction in place) indicated that it is still very difficult to find feasible trajectories. It is not uncommon that half of the Lambert-problems at each iteration can not be solved by an ExpoSin. For $m > 0$ this amount can even increase to roughly three-quarters. Unfortunately, when Lambert-problems have no solution, it is difficult to determine "how bad" the associated trial vector was. This makes it virtually impossible to assign appropriate penalties to them; in such cases there is no other alternative than to re-initialize

these unsuccessful trial-vectors. Because of this, results vary a great deal from optimization to optimization, so that a *single* optimization has a very small probability of yielding the global solution.

As mentioned earlier this chapter it seems necessary to use a somewhat smaller population size because of the higher computational demands of the ExpoSin-lambert targeter[3]. However, because of the large amounts of re-initializations at each iteration (thus "wasting" many FE's), setting the maximum amount of permissible FE's to 150,000 Lambert-problems as intended proved insufficient to give satisfying results. Therefore, for some optimizations, this maximum needed to be reset to 250,000 FE's total (independent of the amount of Lambert-problems involved), and for other problems the population size *also* needed to be set to the larger (high-thrust) value. Indeed, each problem required some manual tweaking before it could return acceptable results. As such, the 10 sequences that were optimized for low-thrust took 3 days of computation time[4].

Despite all this, the 10 best `seq`s were optimized as before (5 times total, once with each individual optimizer) to get a rough impression on the feasibility of a sequence. The results thus found are displayed in Figure 15.11. Note that this figure displays the spacecraft's *dry mass*, which follows from Equation 12.19. For completeness, a plot similar to Figure 15.2 was made for the low-thrust results, although with only 10 sequences the statistic represented by this figure is even more questionable than that of Figure 15.2.



(a) Low-thrust results

(b) Corresponding algorithm performance

Figure 15.11 The results for the optimizations of the best 10 sequences, with low-thrust. Note that these masses do not represent the actual mass delivered to the SBS; they only represent the mass deliverable to the last body in the sequence. Also note that the inclusion of the Sun (Su) for low-thrust trajectories means optimizing with a free $m > 0$, with a maximum of 1 complete revolution.

[3]The ExpoSin-lambert targeter was finally implemented as a (compiled) MEX-file that uses a 50/75-node Chebyshev-interpolation to integrate the time-of-flight integral and/or its derivative. Although this is $\sim 30\times$ faster than with Gaussian quadrature in interpreted `M`-language, it still remains several orders of magnitude slower than its high-thrust counterpart.

[4]Of course, this time includes optimizing several sequences many times, because their results were repeatedly found to have unacceptably large constraint violation.

The values for m_{dry} mentioned in this figure seem quite promising. Keep in mind however that for all these results the last leg to the SBS was evaluated with the *high-thrust* Lambert-targeter. This work-around necessitated *copying* the spacecraft's mass just prior to the last swingby to the m_{dry}-slot, so that the reported masses better describe a low-thrust mission to the last swingby planet than a mission to the SBS.

Note that the sequences `JSu` and `Su` are listed as "impossible" in Figure 15.11. For these sequences no accurate estimate on the mass could be obtained; the sequence `Su` involves a direct transfer from Earth to the SBS which is impossible because of C_3-violation ($\lesssim 800$ km^2/s^2 was not possible) and the sequence `JSu` requires forcing the ExpoSin-parameters such that it leads back to the Sun which always requires an impossible turn angle at Jupiter ($r_p \gtrsim 8000$ km not possible).

In fact, although most effort of the global optimizer was spent on minimizing all constraint violations, the differences between V_∞^+ and V_∞^- at the last swingby could *never* be reduced to less than ∼14 km/s difference, for any of the 10 sequences. This problem is shown more clearly in Figure 15.12, which shows the Pareto fronts with m_{dry} and $|V_\infty^- - V_\infty^+|$ as the two objectives.

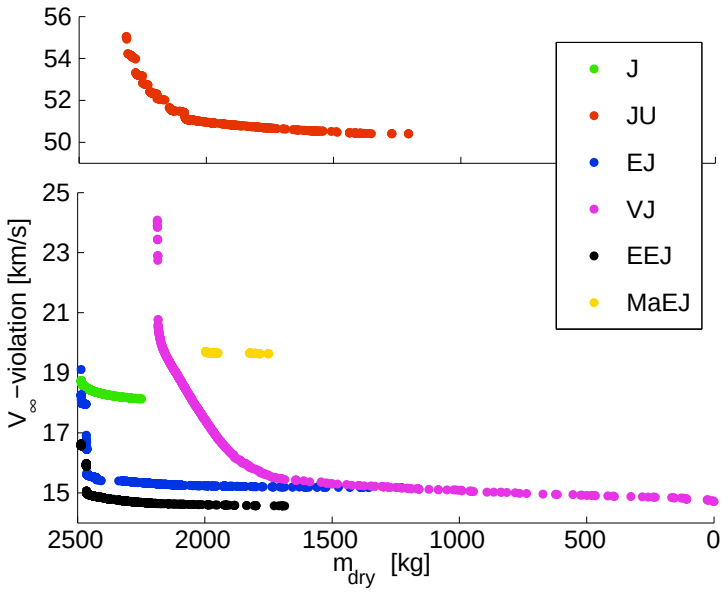


Figure 15.12 Pareto fronts for the 10 sequences considered for a low-thrust mission. Note that the two objectives are m_{dry} and the V_∞-*violation* at the last swingby-body. These fronts show that a low-thrust mission to the SBS is not likely to be possible.

These Pareto fronts[5] prove that planetary GAM's executed with a low-thrust spacecraft can *not* provide enough energy to reach the SBS. Some further attempts with different settings for I_{sp}, $C_3^{\max}$, `seq` (including the Sun) and a "normal" ExpoSin from the last body to the

[5]Even with a reduced population size of 750, some fronts required more than 3 *billion* FE's to converge.

SBS did not produce any feasible result either. Considering this (and the fact that the high thrust results already generated at this point proved so much more promising) it seems best to completely abandon analyzing low-thrust missions at this point and fully concentrate on designing a high-thrust mission.

15.3. Scenario 2 – Bow Shock, with Accidental Minor Planets

Results for this scenario are obtained by switching on the nearby-MP's post-processor in Skipping Stone elaborated in previous chapters, and continuously re-running single-objective optimizations for the `JU-seq` until a set of solutions is created with a satisfying amount of MP's. Setting the maximum allowable $C_3^{\max}$ equal to 157 km^2s^{-2}, these optimizations were then carried out with 3 different settings for the maximum allowable t_f (25 years, 22.5 years and 20 years) to force solutions onto the $C_3 = 157$ km^2s^{-2} isoline from Figure 15.10, but then with the MP-count included. Since the quality of solutions was found to be nearly independent of the minimum approach distance assumed, the default values of $h_J = 42,895$ km and $h_U = 81,533$ km were used for all optimizations to minimize the risks involved.

The best results found during these tests are listed in Table 15.2. Complete lists of the MP's thus found are included in section G.5. Unfortunately, all three scenarios only ever came close to MP's located in the MAB[6], so no KBO's or SDO's were ever found to be in range. This should certainly be part of future work based on this research; to try and "tweak" the final leg(s) so that such bodies *can* be flown by.

15.4. Scenario 3 – Bow Shock, with Pre-Selected Minor Planets

The reader must have noticed that very little attention seems to have been spent on mission scenario 3 throughout this thesis. This is due to the fact that extensive preliminary testing with the completed Skipping Stone made it clear to the author that pre-selecting MP's is simply *too complicated* for a mission to the SBS. It proved to be somewhat of an art to reach but *one* pre-selected MP when the spacecraft also has to fly to the SBS in under 25 years. The original idea of a many-dimensional Pareto-front is still true, but certainly not the only tool that would need to be used. The amount of work this scenario would amount to is certainly equal to (or more than) the amount of work for the other 3 scenario's combined. So unfortunately this scenario had to be abandoned at quite an early stage; all work that was done on the subject has been removed from this thesis accordingly.

[6]Some solutions also included a Near-Earth asteroid, but in all of these cases, no other MP's were found. Note how the number of MP's declines as the spacecraft flies faster through the MAB; a quite intuitive result.

Table 15.2 The best 3 results found for the JU-seq, taking into account the number of MP's and different settings for $t_f^{\max}$. These results were found by repeatedly optimizing the corresponding single-objective problem until a satisfying ΔV and MP-count were found. Note that the more realistic constraint $C_3 = 157$ km^{2}s^{-2} was upheld during all these optimizations.

(a) $t_f^{\max} = 25.0$ years

Launch
May 7$^{\text{th}}$, 2021 C_3: 157.0 km^2/s^2
Jupiter encounter
May 19$^{\text{th}}$, 2022 turn angle: 80.811° ΔV: 5.8594 km/s $h_{\min}$: 52,645 km
Uranus encounter
June 6$^{\text{th}}$, 2024 turn angle: 0.044° ΔV: ∼0 km/s $h_{\min}$: 9,891,800 km
Bow shock (200 AU)
May 5$^{\text{th}}$, 2046 V_∞: 38.983 km/s
Totals
Total ΔV: 5.8594 km/s Potential MP flybys: 14 Travel time: 24.99 years

(b) $t_f^{\max} = 22.5$ years

Launch
May 12$^{\text{th}}$, 2021 C_3: 148.85 km^2/s^2
Jupiter encounter
June 4$^{\text{th}}$, 2022 turn angle: 80.881° ΔV: 8.4628 km/s $h_{\min}$: 42,895 km
Uranus encounter
April 4$^{\text{th}}$, 2024 turn angle: 1.250° ΔV: ∼0 km/s $h_{\min}$: 251,390 km
Bow shock (200 AU)
October 28$^{\text{th}}$, 2043 V_∞: 43.717 km/s
Totals
Total ΔV: 8.4628 km/s Potential MP flybys: 12 Travel time: 22.50 years

(c) $t_f^{\max} = 20.0$ years

Launch
May 16$^{\text{th}}$, 2021 C_3: 135.51 km^2/s^2
Jupiter encounter
July 19$^{\text{th}}$, 2022 turn angle: 80.636° ΔV: 12.906 km/s $h_{\min}$: 42,907 km
Uranus encounter
February 23$^{\text{rd}}$, 2024 turn angle: 2.471° ΔV: ∼0 km/s $h_{\min}$: 82,744 km
Bow shock (200 AU)
May 15$^{\text{th}}$, 2041 V_∞: 49.707 km/s
Totals
Total ΔV: 12.906 km/s Potential MP flybys: 4 Travel time: 19.98 years

15.5. Scenario 4 – Bow Shock, with Intentional Minor Planets

15.5.1 High Thrust

Some further analysis of the feasible sequences (J, SuJ and JU) show that they provide relatively small windows of opportunity. Also, these feasible trajectories consist exclusively of the outer gas giants, which necessarily have long travel times associated with the intermediate legs. Moreover, the ΔV demands for these sequences are already quite high (in the order of 6-10 km/s), so that very little is left to manoeuvre towards multiple MP's. In addition, all the transfer legs are virtually *straight lines* (or very elongated elliptical shapes at least). These combination of these factors does not leave much room to optimize the amount and quality of MP-flybys – in the best case scenario, the leg to the stagnation point can be slightly altered to allow close passage to one or two outer-Solar system objects. Considering also that the computational cost for the optimization of mission scenario 4 would amount to just over a full work week, it hardly seems worth the effort to actually carry it out. A partial conclusion can thus be formulated: the given framework is too demanding for this scenario to succeed, and an optimization of this kind has to be left as future work; quite possibly in a different context altogether.

15.5.2 Low Thrust

In addition to the absence of feasible swingby sequences for a low-thrust mission, the process of finding nearby MP's with the method of ExpoSin's also proved too slow to allow such an optimization to complete in any reasonable amount of time. The estimated time to completion amounted to about 6.5 weeks per sequence. This estimate comes from assuming that each iteration would increase the non-domination count by 5 individuals on average, an almost unacceptably small population size of 50 individuals and loosening most of the constraints. An automated translation of the minimum distance routine written in interpreted `M`-code via *Embedded MATLAB®* to a compiled-C++ routine showed an improvement in speed by a factor of about ~ 1.5, so at least in this case that is clearly not the bottleneck. Parallelizing the computation (using the Parallel Computing Toolbox (PCT)) to 5 nodes of $\sim 1.2 - 2$ Ghz did indeed push the estimated required time back to slightly longer than 2 weeks, but in light of the absence of feasible swingby sequences such an optimization is simply not worth the effort.

Moreover, preliminary work on second-order low-thrust trajectories also showed that an ExpoSin-trajectory is actually a very poor approximation to the "true" optimal solution; the m_{dry} reported indeed gives a reasonable estimate of that value, plus the start and end-points reported as globally optimal nearly always were in a basin of attraction for second-order analyses, but the actual *trajectory* is approximated quite badly by an ExpoSin – the "nearby" MP's found with the MPE often remain quite far from the second-order trajectory. Therefore an MP/m_{dry}/t_f – Pareto-front found with the method of ExpoSin's would deviate quite much from such a front generated with a second-order optimization method – *too* much in fact to be valuable.

15.6. Algorithm Performance

A detailed record was kept of all single-objective global optimizations performed thus far, for both high-thrust and low-thrust (see appendix G). This record can be converted into a more reliable statistic on how each algorithm performs when left to its own devices. To this end, the global minimum found in each individual set of optimizations (per `seq`) is subtracted from all results for all optimizers for that particular set, and when the remainder of that subtraction is found to be less than 0.25 km/s (or 10 kg for low-thrust results), that particular algorithm gains one point. All points thus assigned for all algorithms are added, and divided by the total number of optimizations performed (444 per algorithm in total) to express the algorithm's overall success-rate in a percentage. These success-rates are shown in Figure 15.6. Note that these scores include the $(146 + 10) \times 5$ "single-shot" optimizations carried out for the 146 high-thrust sequences and 10 low-thrust sequences, all the $10 \times 5 \times 5$ optimizations for the best 10 sequences of these 146, and all $16 \times 3 \times 5 \times 5$ for the 16 different points for the 3 best sequences (so 2,220 single-objective global optimizations total).

It can readily be concluded that the GODLIKE algorithm has the highest success-rate ($\sim 59\%$), closely followed by the GA ($\sim 52\%$). Note especially the *very* disappointing result for the PSO optimizer; despite its high degree of sophistication and complexity, in its current form it is clearly not suited for problems of this nature; it is rather surprising that GODLIKE

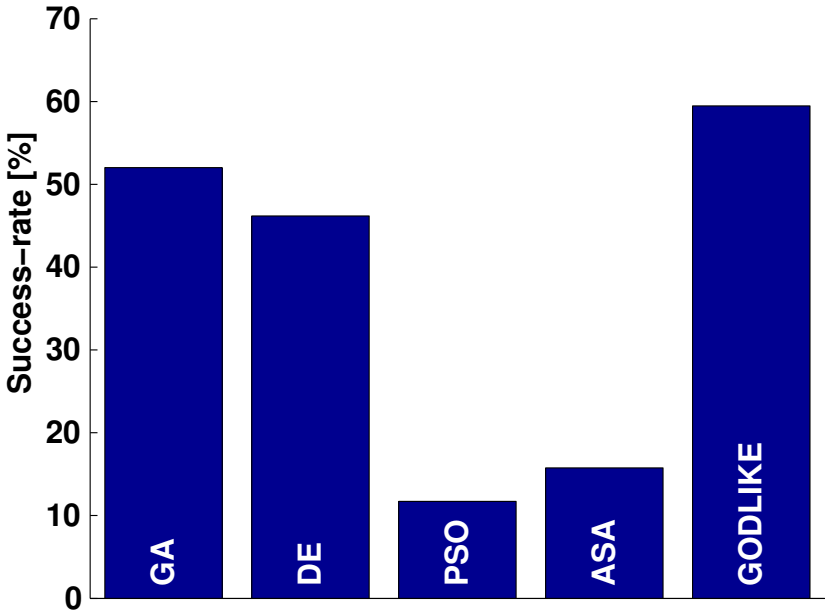


Figure 15.13 Overall algorithms performance for all 5 optimizers used during the optimizations. The score is based on subtracting the global minimum found for each optimization from all of the results from all optimizers for that optimization, and calculating how often each optimizer came within 0.25 km/s (or 10 kg) of the global solution.

still performs so well with PSO included. The poor performance of the SA optimizer was actually expected; simulated annealing was originally designed to be a global optimizer for small-scale problems, e.g., to overcome the problems of regular hill-climbers when a reasonable initial estimate had already been provided. The scale of these trajectory optimization problems is simply too large for the method. The fact that PSO performs even *worse* than SA is however, quite remarkable.

Note that this is a comparison of all *single*-objective optimizations performed thus far. Comparing the quality of the *multi*-objective optimizations is not possible at this stage; the SA and PSO algorithms had already been discarded as potential MOO-optimizers in chapter 7, and more importantly, only *one* algorithm was used for all MOO's (the NSGA-II algorithm), so there is not that much to compare. A comparison of the performances of NBI versus NSGA-II in the context of trajectory optimization for example, must be left here as a recommendation.

Note also that all algorithms (GA and GODLIKE included) exhibit relatively poor performance – $\sim 60\%$ is actually quite far from the 80-90% (100%?) one would like to see a global optimization algorithm accomplish, especially in this context. This result indeed proves that all global trajectory optimizations in similar contexts *demand* several re-runs before the true optimum can be located with a high degree of certainty; future versions of GODLIKE will certainly see some attempts to improve upon this problem.

16

Second-Order Results and Final Design

16.1. Best Result, Second-Order

Originally, the intention of this chapter was to provide a comparison between the best trajectory as found with the MPC and a much more accurate numerical integration. As mentioned in previous chapters, the difference between these two methods does not change the final solution significantly; it can safely be said that the differences would amount to roughly a few days in the travel times and a few hundred kilometers in the approach distances. The more important intention of this chapter was to determine the differences between the MP close-approach distances as found with the two-body assumption and pruning methods outlined in chapter 9, and a numerical integration taking into account the gravitational interaction of most of the larger bodies in the Solar system. This *could* indeed give significantly different results, as demonstrated by Figure D.2, providing a measure of how accurate and useful these two-body routines are in the context of interplanetary missions.

Unfortunately, *painfully* even, at the very last instant, it was determined that the routines embedded in Skipping Stone to read out the *names* of the MP's from the MPCORB and the corresponding *data* were in complete disagreement. This was determined to be due to a language conflict: the MPCORB is formatted specifically to be read by FORTRAN (data aligned by specific column ranges), while MATLAB® uses the C++-style `fprintf`-command which assumes the data is comma or tab delimited. Problems with this difference occur when some lines suddenly contain data in columns where there had been none on the previous line, as is the case for some 100 lines in the MPCORB. At these locations the readout fails *without* giving an error in MATLAB®, resulting in orbital data appearing in some of the names. This in itself would not be a problem (just skip that one or use the previous name), but some further analysis also brought to light that in fact quite a significant portion of the *orbital* data got mixed up as well.

This sort of problem also propagated through to the minimum-distance routines; trying not to go into too much language-specific detail, some parts of these routines used the *names* to index to specific MP-data, while other parts used their *numbers*. On top of that, there was a minor bug also in the *pruning* routines, causing also the *numbers* to be incorrectly assigned to new members of the set. There is now absolutely no way to know for sure which result belongs to which MP and which MP's have correct orbital data. So, unfortunately, also this second-order analysis has to be left here as future work.

"There is *always* one more bug."

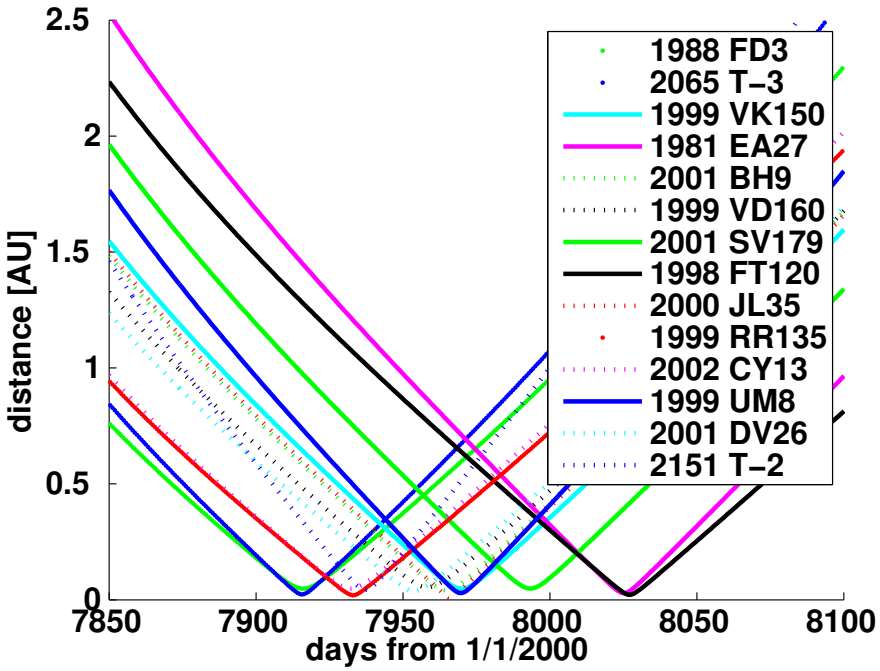


Figure 16.1 The time-dependent distances between the spacecraft and the 14 MP's located with the minimum-distance routines.

16.2. Final Solution

The best solution found thus far is given in Table 16.1. A nice overview of the shape of the trajectory is shown in Figure 16.2. This solution is indeed copied from Table 15.2(a), as that particular solution had already been selected based on its favorable ΔV and MP-count. In fact, in all $\sim$40 manual trials performed to find it, this solution was the one with the lowest demands on ΔV.

For completeness, the distances to the 14 MP's that *were* found is given in Figure 16.1. However, keep in mind it is not known whether *these* MP's approach the trajectory and/or *whether* these MP's approach the trajectory. But at the very least, the routine itself seems to work correctly, as this figure clearly indicates. As this fact is irrespective of exactly which MP is used, re-starting future work on this subject will not likely be much of a problem.

Table 16.1 The very best solution. Note that this is equal to Table 15.2(a) – a higher accuracy second-order analysis was not performed.

Launch
May 7th, 2021 C_3: 157.0 km^2/s^2
Jupiter encounter
May 19th, 2022 turn angle: 80.811° ΔV: 5.8594 km/s h_{min}: 52,645 km
Uranus encounter
June 6th, 2024 turn angle: 0.044° ΔV: ∼0 km/s h_{min}: 9,891,800 km
Bow shock (200 AU)
May 5th, 2046 V_∞: 38.983 km/s
Totals
Total ΔV: 5.8594 km/s Potential MP flybys: 14 Travel time: 24.99 years

16.3. Satisfying Secondary Requirements

This mission's original secondary requirements were (see section 3.2.2):

1. Explore various bodies in both the inner and outer Solar system, in search of clues to their origin and to the nature of other planetary systems.

2. Quantify the exact magnitude of the effect associated with the Pioneer and flyby anomalies, and investigate one or more of the proposed origins of these phenomena.

3. Investigate the mass distribution and the likely evolution history of objects in the Kuiper belt, in search of the origin of the belt's prevalent mass defect after the 1:2 resonance.

4. Investigate the validity of Einstein's equivalence principle, and quantify the spatial-temporal variation of the fine structure constant α in order to validate or reject grand unifying theorems proposed in some string theories.

Since the final trajectory does not come closer to the Sun than the Earth's orbit, it is not very fruitful to try and optimize the scenario where a small probe is launched into a circular orbit at 4 Solar radii. The last of these requirements was included under the assumption that the result involved a much closer Solar approach, but with the selected solution this requirements can definitely not be met. Also, the final result could not come close to any outer-solar system objects, so that also the third requirement is not met. However, the last leg towards the SBS

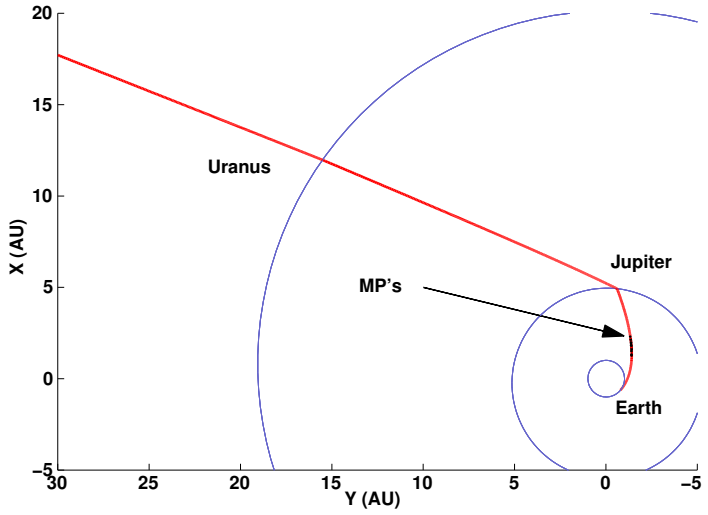


Figure 16.2 shape of the final trajectory. Naturally, because high speed is basically this thesis' primary concern, the resulting trajectory is indistinguishable from a straight line.

could be altered slightly in further analysis (using a slightly different turn angle at Uranus) to allow a close proximity flyby with one or a few outer-Solar system objects, so meeting this requirement (at least partially) would be possible with a bit more research. Fortunately, the 14 MP's encountered leave enough room to optimize the Earth-Jupiter leg in such that at least a few of these MP's can be flown by with a reasonably close approach distance and minimal impact on the overall ΔV. Therefore the selected solution *did* indeed partially meet the first secondary requirement.

17

Conclusions & Recommendations

17.1. Conclusions

This research investigated the feasibility for a space mission set to explore the entire heliosphere, the Solar bow shock and the interstellar space beyond. Considering the vast distance to be covered, a maximum of 15 years was imposed on the time of flight. The mission would exclusively use proven technologies and techniques and realistic constraints on the spacecraft's design.

Two techniques were explored in this research: powered/unpowered gravity-assist manoeuvres at various planets, and a single deep-space manoeuvre carried out in close-proximity of the Sun. Also, two propulsion techniques were analyzed individually: classical chemical rockets and ion engines. The original constraint on the time of flight of 15 years necessitated using a maximum of 3 consecutive planets to be used for gravitational-assist manoeuvres. This maximum, in combination with the other constraints, gave a total of 146 distinct combinations, of which 3 were found to produce feasible results. The original constraint on the time of flight of 15 years was found to be impossible to meet and had to be increased to 25 years. With this increased time of flight, the other constraints could only be met if a high-thrust system was used. The 3 different solutions were elaborated extensively, after which a best overall scenario could be selected. This selection was based in part on the number of minor planets the spacecraft would come close to, although the first of these criteria (the MP-count) had to be abandoned due to an implementation error found in the last stages of the research. This particular result enters interstellar space almost exactly 25 years after launch, and requires a total velocity change of 5.86 km/s. A total of 14 minor planets were found to come within 0.03 AU of the spacecraft's trajectory.

The selected solution requires a launch in May 2021 using a high load launch vehicle ($C_3 \sim 157$ km^2/s^2). The launch is to put the spacecraft on a course to Jupiter, which it will reach

after approximately 377 days. During this time there is an opportunity to perform a flyby with one or more of 14 candidate main belt asteroids. Once arrived at Jupiter, the spacecraft performs a gravity-assist manoeuvre requiring a ΔV of approximately 5.86 km/s and an approach distance no closer than 52,645 km. This manoeuvre will bring the spacecraft en-route to Uranus, which it will reach after approximately 749 days. The spacecraft's closest approach to Uranus will be approximately 9.8 million kilometers, providing an opportunity to observe the planet and its satellites for the second time in history. After the encounter with Uranus, the spacecraft will continue its flight through the heliosphere, crossing the termination shock and the entire heliopause, finally reaching a 200 AU distance approximately 8003 days after the Uranus encounter.

During the development of the novel technique used to optimize the aforementioned deep space manoeuvre, several secondary mission requirements were added to try and broaden the mission's scientific output. These were selected and added under the assumption that the optimal trajectory could contain a leg that passes closer to the Sun than Mercury's orbit. The final result described above does not conform to this expectation, so that most of these secondary requirements can not be met, or at the very least, require further analysis. The following list summarizes all mission requirements assumed throughout the research (primary and secondary), and the capacity of the end-result to meet them:

Mission requirement	**Can be met?**
Establish the existence of the Solar bow shock, and measure the wave's location, magnitude, shock strength and temperature.	√
Investigate the process responsible for the acceleration of the pick-up ions to anomalous cosmic rays.	√
Explore the contents of the interstellar medium, and the implications for the origin, evolution history and future of matter in our Solar system, galaxy and the universe thereof.	√
Establish the capability of existing and proven technologies to enable fast and affordable exploration beyond our own Solar system.	√
Explore various bodies in both the inner and outer Solar system, in search of clues to their origin and to the nature of other planetary systems.	?
Investigate the mass distribution and the likely evolution history of objects in the Kuiper belt, in search of the origin of the belt's prevalent mass defect after the 1:2 resonance.	×
Quantify the exact magnitude of the effect associated with the Pioneer and flyby anomalies, and investigate one or more of the proposed origins of these phenomena.	?
Investigate the validity of Einstein's equivalence principle, and quantify the spatial-temporal variation of the fine structure constant α in order to validate or reject grand unifying theorems proposed in some string theories.	×

Despite the fact that the original demand of a 15 year mission could not be met, the final result does prove that interstellar missions are possible with existing technologies, albeit barely. Although the Solar bow shock may not be reachable in realistic time spans with the

techniques tried, the results found in this research demonstrate that leaving the planetary region and reaching the *termination* shock can be accomplished in roughly 12.5 years; more than *twice* as fast as the Voyager missions. It thus seems very worthwhile to further develop this mission, but then in a slightly less ambitious setting.

17.2. Discussion

The vast distance to the bow shock casted much doubt on the very existence of feasible solutions from the very beginning. The fact that no less than 3 result were found requiring less than 7 km/s velocity change is a surprising and remarkable find in its own right. Nevertheless, the required time of flight of 25 years still makes this mission not a very realistic scenario. Additionally, the launch date (May 2021) is *just* too far in the future to save the mission's appeal.

As was found in appendix D, Skipping Stone (the program used in the research) was able to reproduce both the Voyager 1 & 2 trajectories and the trajectory flown by the Cassini/Huygens spacecraft. However, it was *unable* to reproduce the trajectory flown by the Galileo spacecraft. This is rather unsettling, because this renders all of this research' results and conclusions quite uncertain. Needless to say it is a rather pressing issue, which might at least partially explain the absence of more (or more viable) sequences. Nevertheless, the most prominent and important part of the spacecraft's trajectory (the final leg to the bow shock) is hard to get wrong, so the conclusions regarding the *feasibility* are not changed by this issue.

As was found during the final analysis, changing the target location from the SBS's stagnation point to points surrounding it drastically improved the solutions' quality. This hints at the possibility to further explore this concept and investigate in a more general way which locations could result in more realistic mission scenarios. This certainly seems worthwhile when examining the turn angle required at the Jupiter swingby in the final solution; the requirement of $>80°$ in itself almost certainly places high demands on the ΔV, while that same ΔV could be put to much better effect by using it to increase the heliocentric V_∞. Nevertheless, the deviations from the stagnation point considered (12°) were deemed large enough already; taking points an even larger distance away from the stagnation point would cast serious doubts on the correctness of the assumed 200 AU distance to the SBS, as larger deviations generally imply longer distances to be travelled. But exactly what this relationship between deviation angle and distance is, is subject for further study.

17.3. Recommendations & Future Work

Several issues cropped up during this research which leave room for improvement. Most of them are relatively minor, but some could significantly improve both the quality of the outcome of this study and that of future studies based on it. Therefore, any further study on this topic must first address the following issues (in no particular order):

Solar Flyby: Although the novel SF-algorithm did not provide this thesis' final solution, the algorithm itself is quite promising. In many cases the algorithm converged onto a

solution that one would expect; a close proximity-flyby at the Sun perfectly connecting two dissimilar trajectories. In other cases the algorithm returned a result where the pericenter distance r_p was almost equal to one of r_1 or r_2; much more like a regular DSM. Its ability to generate near-DSM's makes it a very appealing new tool for (interplanetary) trajectory design. There were however some implementation problems found, most notably the occasional failure of the SQP-algorithm and the markedly long computation times required. Also, towards the end of the research it was found to be impossible to generate Pareto-fronts with the algorithm in some situations. In all, before it can be used to its fullest potential, more work is definitely required to perfect it.

Accuracy, High-Thrust: The accuracy of the final trajectory design may be further improved by including the following aspects:

- Solar radiation pressure,
- Non-spherical planets,
- Finite burn-times,
- Relativistic Effects,
- More accurate masses for the minor planets,
- Atmospheric perturbations when performing GAM's,
- etc.

These aspects were not included, because the emphasis of this thesis was on the *feasibility* of a Solar bow shock mission. Since a broad search was necessary, such relatively small factors could not be taken into account. Most perturbative forces mentioned above require knowledge on the frontal surface area, instantaneous attitude of the spacecraft etc., which required designing the whole spacecraft as well. Nevertheless, some of these factors are relatively straightforward to implement in Skipping Stone, so this is definitely worth implementing in future versions.

ExpoSins method: Despite its appeal, using the method of ExpoSin's for the preliminary design of low-thrust trajectories is *not* recommended – it is quite costly in terms of computation time while the end-product is far from being optimal in *any* sense. The only thing the method of ExpoSin's seems to be able to do is give a *rough* estimate on the configuration to use for an actual low-thrust optimization. The trajectory described by an ExpoSin is in fact *so* far removed from a trajectory that follows from a more elaborate optimization, that it is quite meaningless to use that ExpoSin for any further analysis. The process of finding nearby MP's as investigated in this thesis is a good example of this; candidate MP's found from minimizing the distance between the ExpoSin and their Keplerian orbit did not come even remotely close to trajectories optimized with MPμC or collocation (both using the same ExpoSin as their initial estimate). From a theoretical standpoint this is to be expected; the ExpoSin's are derived from assuming a specific, easy-to-analyze *geometry*, rather than the physics involved with a continuously-thrusting spacecraft. This assumed geometry is then basically *forced* to obey the laws of physics, rather than the other way around. This method is indeed likely to be incompatible with

everything else in orbital mechanics. For example, one would expect that the ExpoSin for simple geometries and initial velocities would closely resemble a (perhaps slightly perturbed) conic section; but this is not the case. Instead, the result is *always* part of a spiral and *always* consumes propellant, even when that is not applicable to the situation at hand. It is therefore strongly recommended to investigate other methods that *are* derived from actual physics. One such method is briefly described in [Schütze et al., 2007]. These authors came to a similar conclusion regarding ExpoSin's, and came up with a method that (although more costly) *did* produce close-to-optimal trajectories.

Accuracy, Low-Thrust: The accuracy obtained in the second-order high-thrust optimizations (i.e., integrating both the spacecraft and the MP's with the full DE405 force model) is much higher than that obtained in the second-order *low-thrust* optimizations. This is because the first-order (ExpoSin's) and second-order (MPμC and collocation) methods are based on geometry and/or simplified (two-body) force models. None of these methods allow a direct integration of the spacecraft's trajectory using the full DE405 force model, giving final accuracies that are quite poor compared to the high-thrust optimizations. A low-thrust mission was finally shown to be unable to reach the SBS in a timely fashion, so this remark does not apply to this research. However, in future versions of Skipping Stone it is advisable to introduce *third-order* and/or *fourth-order* procedures for any low-thrust optimizations, which *will* give the desired high-accuracy. These procedures will first have to use an indirect method to obtain the exact functional relationship between the position of the spacecraft and time, followed by a direct integration using the full DE405 force model and further perturbations. These procedures are definitely not trivial, and probably require another complete thesis research to implement correctly.

Locating nearby MP's: A wide variety of methods was used to prune out large numbers of minor planets from search spaces inside cost functions. One of these pruning methods is determining the minimum geometric distance between two conic sections, as described in section 11.4.1. A large number of experiments with the various algorithms that determine this minimum distance showed that such a calculation is actually quite a lot more intensive than the calculation of the time-dependent distance. Moreover, Newton's method as suggested by Hoots et al. [1984] proved to be very unreliable; the minimum distance between conic sections was only located in roughly half of the cases. The other half converged onto some local minimum, usually a great deal larger than the actual minimum. The method invented by Kim [2006] was much better in this respect, but both algorithms actually *increased* the required calculation time compared to a calculation without this kind of pruning. This geometric pre-filter was therefore not used in this research, and this is also recommended for any other related work: until faster and more robust algorithms are developed that locate the minimal distances between conic sections, this pruning rule can better be omitted.

Communications: Considering the *enormous* distance the spacecraft will be removed from Earth when it reaches true interstellar space, a very important aspect of this mission is communication between the spacecraft and Earth. In this research, no serious attention has been given on the link budget, power requirements for communication and so on, which might prove to be a bottleneck for the feasibility of this mission. Communica-

tions were not considered in this research, because communications with the Voyager spacecraft seemed not to be a problem even at the distances these spacecraft currently are. Taking into account the advances in technology since the start of the Voyager missions, it was silently assumed communications would not prove problematic for this mission. However, if this research is to be used for any future mission, a substantial amount of time should go into research on communications, and how this translates into the mission's feasibility – the distance that will be attained exceeds *three times* the distance the Voyager spacecraft have thus far reached.

Global Optimization Algorithms: More research definitely needs to be done on the global optimization algorithms used in this research. The single-objective algorithms have been investigated quite thoroughly, but their multi-objective counterparts can almost certainly be improved significantly. Especially the multi-objective PSO (MOPSO) algorithm tested could not re-produce any of the the results obtained with the same algorithm in various publications on the subject, which indicates that some sort of error still resides in the current version. Also, although GODLIKE operated reasonably well for a new algorithm, its intention was to eradicate the need for fine-tuning for each new problem. Many pragmatic assumptions have been made during its construction (re-heating, re-starting PSO at each iteration, etc.) which likely have a negative impact on its performance compared to its potential. It is safe to assume that GODLIKE's performance can be improved significantly with more research.

NASA/JPL HORIZONS suite for MATLAB: The cubic-splines interpolation method described in appendix B is rather *ad-hoc*; it was only introduced to prevent a tedious translation of the HORIZONS software from `FORTRAN` to MATLAB®. Unfortunately, this method requires storing data on ephemerides of all the bodies in the DE405 model, with a 30-day step. This adds up to ∼12 MB of data, which renders Skipping Stone much less portable. Moreover, HORIZONS was developed specifically to reduce the required data for accurate ephemerides; this cubic-splines method goes directly against that philosophy. At the time it was a "necessary evil", because the ephemerides were the only thing that stood in the way of fully re-coding GALOMUSIT/OPTIDUS into MATLAB®. Fortunately, after a bit more research, it was found that the complete HORIZONS software has been ported to MATLAB®. It is freely available (from `http://www.openchannelsoftware.com/projects/Matlab-Based_Solar_System_Ephe/`) and provides exactly the same accuracy, speed and functionality as the original `FORTRAN`-implementation. It should be obvious that with some re-coding, the generation of ephemerides can be handled entirely by this toolbox. It will give better accuracy (no additional interpolation errors), less data (all data required are the Chebyshev-coefficients), and will probably also result in a faster Skipping Stone.

The following recommendations do not describe problems that turned up during the optimization of this mission. Rather, they describe relatively easy ways to increase Skipping Stone's computation speed, possibly by a factor of 500. This makes this section very worthwhile to read for all future developers of the program:

Automatic Differentiation: Local optimization methods such as quasi-Newton (L)BFGS, SQP or similar algorithms all require gradient-vectors (and/or Hessian-matrices) for the objective function and all of its constraints. For this research, extensive use was made of finite-differences to compute these derivatives numerically. As often mentioned in this thesis, this operation is extremely costly in terms of computation time. Fortunately, there exists a much better alternative: Automatic Differentiation (AD). This technique repeatedly applies the chain rule on *every operation* that is performed on a variable, so that in principle a *single* FE is enough to give both the local gradient and Hessian matrix for that function. In most implementations, this is accomplished by *operator overloading*, which means that it can work on all existing functions without *any* re-writing, provided the function works on members of a class different from `double`. An excellent toolbox that implements this technique (among many other very useful techniques) is called the *INTerval LABeratory* (INTLAB) (available from `http://www.ti3.tu-harburg.de/~rump/intlab/`). However, this technique came only to the author's attention when this thesis was already nearing completion, so that it is currently not used in Skipping Stone. But, as should be obvious, using this technique can lead to dramatically more efficient local optimizations.

Parallel Computation: At the time of writing, the improvements in the computational capacity of electronic micro-processors have reached the absolute limits of electronics; their speed can simply not be increased any further. However, already some time ago, microchip manufacturers started addressing this issue by incorporating *multiple CPU's* on a single chip; the so-called *multi-core CPU's*. More importantly, the ever increasing demands of computer game enthusiasts forced the designers of graphics processors (*GPU's*) to implement vast amounts of processing power onto later generation graphics cards. When this started happening, many computer enthusiasts started experimenting with re-programming their GPU's so that they could execute calculations *other then just graphics manipulation* on their GPU's. These attempts proved so successful that not long after that, GPU manufacturers came with dedicated programming languages that allowed *anyone* to access the (up to) 512 processing units that were now present on single graphics cards. These developments thus enabled consumers to have their very own *super-computer*, at the cost of just an off-the-shelf graphics card.

On single CPU's, computations are carried out *sequentially*, that is, every line of a computer program is executed individually, and execution of the next line is halted until the execution of the previous line has (successfully) ended. However, when multiple processors are available on the same machine, it is possible to execute programs in *parallel* – execution of lines or loops can be done *simultaneously*. This approach, especially with the inexpensive and large amounts of processors available on graphics cards today, will potentially speed up any program by several *orders of magnitude*.

Although the programming techniques needed to use parallel computing effectively and efficiently are quite complicated, much work has already been done on this subject. The standardized MPI-interface, already used on large computer clusters for decades, can be loaded as a simple library into most computer languages. It facilitates communication issues between different computers on a cluster, or more importantly, different proces-

sors on a single machine. Effectively, MPI-enables the programmer to do large chunks of the computation on separate machines/processors, with only moderate amounts of re-coding in existing software.

MATLAB® can however not use MPI directly, as it is an interpreted language. Newer versions *do* have native support for multi-core processors, but as of yet this only works for element-wise operators (`sin()`, `cos()`, etc.) and fairly basic routines (`sort`, `min/max`, `bsxfun`,...), *not* for entire loop-structures. Fortunately, several interesting "work-arounds" exist which enable full parallel computing in MATLAB®. One solution comes directly from The MathWorks® company: the PCT. This toolbox allows the programmer to execute each iteration in `for`-loops for instance on a different processor (provided each iteration is independent from the next). However, there are better ways (not to mention free of charge) to accomplish this. One is a completely user-developed collection of `m`-files, called MultiCore (available from `http://www.mathworks.com/matlabcentral/fileexchange/13775`). This program mimics most of the PCT's behavior, and promises significant speedups on multi-core systems, almost without *any* recoding of existing `m`-files.

The most promising alternative however is the free & open-source *GPU toolbox for MATLAB*, called *GPUmat* (available from `http://www.gp-you.org/`). It is based on NVIDIA's CUDA language, and allows MATLAB-users to directly access the vast parallel computing abilities of NVIDIA graphics cards, simply by extending the `m`-language with a few additional classes. Only a very small amount of re-coding is required to execute MATLAB-programs almost entirely on GPU's.

Unfortunately, at the time of writing, this author did not have the financial means to freely experiment with these extensions. However, since multi-core processors and many-processor GPU's have become completely mainstream, these improvements are so obvious that any future Skipping Stone developer is strongly advised to put his/her first efforts into implementing these improvements.

Bibliography

Angrum A. Voyager - the interstellar mission. Online resource, last retreived January 22st, 2010. http://voyager.jpl.nasa.gov/.

Angrum A., Medina E., and Sedlacko D. Voyager – Interstellar Mission. Online resource, http://voyager.jpl.nasa.gov/news/voyager_squashed.html. Last retrieved 15th Jun, 2009, 2007.

Axford W.I. and Suess S.T. The heliosphere. Online resource. http://web.mit.edu/space/www/helio.review/axford.suess.html#Inter. Last retrieved May 25th, 2009, 1994.

Battin R.H. *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition*. American Institute of Aeronautics and Astronautics, Virginia, 1999.

Benson T. Liquid rocket engine. Online resource, 2009. http://www.grc.nasa.gov/WWW/K-12/rocket/lrockth.html.

Bernelli-Zazzera F., Berz M., Lavagna M., Armellin R., Lizzia P. Di, and Topputo F. Global trajectory optimization: Can we prune the solution space when considering deep space maneuvers? Technical report, ESTEC, 2007. A0/1-5180/06/4101 Final Report, Contract no. 20271/06/NL/HI.

Berthoud Marc. ALP-SAT: Scientific mission planning for a micro satellite. Diploma Thesis, available through http://www.astro.phys.ethz.ch/edu/theses/berthoud/. Last retrieved 15th Jun, 2009, 1997.

Bertotti B., Iess L., and Tortora P. A test of general relativity using radio links with the Cassini spacecraft. *Nature*, 425:374–376, 2003. doi: 10.1038/nature01997.

Binney J. and Merrifield M. *Galactic Astronomy*. Princeton University Press, 1998. ISBN 0-691-02565-7.

Blackwell T.M. and Branke J. *Applications of Evolutionary Computing*, chapter Multi-swarm Optimization in Dynamic Environments, pages 489–500. Springer-Verlag, 2004.

Bombardelli C. and Izzo D. ESA/ACT's space mechanics toolbox – MGA.M. Online resource, 2007a. http://www.esa.int/gsp/ACT/doc/INF/Code/globopt/mga/.

Bombardelli C. and Izzo D. ESA/ACT's space mechanics toolbox – MGADSM.M. Online resource, 2007b. http://www.esa.int/gsp/ACT/doc/INF/Code/globopt/mgadsm/.

Bond V. and Allman M.C. *Modern Astrodynamics*. Princeton University Press, 1996. ISBN 0-691-04459-7.

Boyd J.P. Computing the zeros, maxima and inflection points of Chebyshev, Legendre and Fourier series: solving transcendental equations by spectral interpolation and polynomial rootfinding. *Journal of Engineering Mathematics*, 56(3):203–219, 2006. ISSN 0022-0833. doi: 10.1007/s10665-006-9087-5.

Burmen A. and Tuma T. Unconstrained derivative-free optimization by successive approximation. *Journal of Computational and Applied Mathematics*, 223(1):62–74, 2007. ISSN 0377–0427. doi: 10.1016/j.cam.2007.12.017.

Chobotov V.A. *Orbital Mechanics*. AIAA, 3 edition, 2002. ISBN 1-56347-537-5.

Corradini S. Low-thrust orbits for interplanetary transfers and implementation in GALOMUSIT. Master's thesis, Delft University of Technology, 2009.

Das I. and Dennis J.E. Normal-boundary intersection: a new method for generating Pareto optimal points in multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

Pater de I. and Lissauer J.J. *Planetary Sciences*. Cambridge University Press, 1 edition, 2005.

Deb K., Pratap A., Agarwal S., and Meyarivan T. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

Debban T.J., McConaghy T.T., and Longuski J.M. Design and optimization of low-thrust gravity-assist trajectories to selected planets. *AAS/AIAA Astrodynamics Specialist Conference and Exhibit*, August 2002.

Decker R.B., Hill M.E., and Krimigis S.M. JHU/APL voyager LECP information and data. Online resource, last retreived January 22st, 2010. `http://sd-www.jhuapl.edu/VOYAGER/`.

Delsanti A. and Jewitt D. *The Solar System Beyond The Planets*, chapter 11, pages 267–293. Springer Verlag, 2006. ISBN 978-3-540-26056-1. 10.1007/3-540-37683-6-11.

D'Errico J. FMINSEARCHBND and FMINSEARCHCON. MATLAB File Central, `http://www.mathworks.com/matlabcentral/fileexchange/8277.`, 2006. Last accessed 7th May 2009.

D'Errico J. DERIVEST - automatic numerical differentiation. MATLAB File Central, `http://www.mathworks.com/matlabcentral/fileexchange/13490`, 2009. Last accessed 19th May 2009.

Doblas F., Drigani F., Gall F. Le, Wilson A., and L'Abbate F. Space missions to asteroids. Online resource, `http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=2306#P16_2765`. Last retrieved May 31st, 2009.

Dormand J.R., El-Mikkawy M.E.A., and Prince P.J. High-order embedded Runge-Kutta-Nyström formulae. *IMA Journal of Numerical Analysis*, 7(4):423–430, 1991.

Dunbar B. and Boone K. New Horizons: Mission to Pluto and the Kuiper belt. Online resource, 2009. http://www.nasa.gov/mission_pages/newhorizons/main/index.html. Last retrieved 9th September 2009.

Dunne J.A. and Burgess E. *The Voyage of Mariner 10 Mission to Venus and Mercury*. NASA History Office, 1978. Online version, http://history.nasa.gov/SP-424/ch4.htm. retrieved on Dec. 18th. 2008.

ESPG . Interstellar medium learning pages. Online resource, http://www-ssg.sr.unh.edu/ism/Interdepth.html. Last retrieved May 24th, 2009, 2009. Written and upkept by the University of New Hapmshire Experimental Space Plasma Group.

Feoktistov V. *Differential Evolution In Search of Solutions*. Springer Verlag, 2006. ISBN 978-0-387-36895-5.

Ford J.A. Improved algorithms of illinois-type for the numerical solution of nonlinear equations. Technical report, University of Essex, 1995. Report CSM-257.

Frisch P. The galactic environment of the Sun. *American Scientist*, 88:52, 2000a.

Frisch P. The galactic environment of the Sun. *Journal of Geophysical Research*, 105(A5): 10.279–10.289, 2000b.

Gill E. and Montenbrück O. *Satellite Orbits – Models, Methods, Applications*. Springer-Verlag Berlin, 2000.

Gockenbach M.S. Introduction to sequential quadratic programming. Online resource, http://www.math.mtu.edu/~msgocken/ma5630spring2003/lectures/sqp1/sqp1.pdf. Last retrieved 16th May, 2009, 2003. Introductory Lecture for SQP on a course in optimization tought at the University of Michigan.

Goldberg D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Kluwer Academic Publishers, Boston, 1989.

Gooding R.H. A procedure for the solution of lambert's orbital boundary-value problem. *Celestial Mechanics and Dynamical Astronomy*, 48:145–165, 1990.

Gronchi G.F. An algebraic method to compute the critical points of the distance function between two Keplerian orbits. *Celestial Mechanics and Dynamical Astronomy*, pages 295–329, 2005. doi: 10.1007/s10569-005-1623-5.

Gruntman M., Roelof E.C., Mitchell D.G., Fahr H.J., Funsten H.O., and McComas D. J. Energetic neutral atom imaging of the heliospheric boundary region. *Journal of Geophysical Research*, 106:15767–15781, 2001.

Hargraves C.R. and Paris S.W. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987. doi: 10.2514/3.20223.

Heiligers J. Trajectory optimization for an asteroid/comet sample return mission. Master's thesis, Delft University of Technology, 2008.

Hoffman J.D. *Numerical Methods for Engineers and Scientists*. McGraw-Hill inc., 2001. ISBN 0-8247-0443-6.

Hoots F.R., Crawford L.L., and Roehrich R.L. An analytic method to determine future close approaches between satellites. *Celestial Mechanics*, 33:143–158, 1984. DOI 0008-8714/84/0332.

I. Das J. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural and Multidisciplinary Optimization*, 14(1):63–69, 2005. ISSN 1615-147X (Print) 1615-1488 (Online). doi: 10.1007/BF01197559.

Isakowitz S.J. *International Reference Guide to Space Launch Systems*. AIAA, 2 edition, 1991.

Izzo D. Lambert's problem for exponential sinusoids. *Journal of Guidance Control and Dynamics*, 29:1242–1245, September 2006.

Izzo D., Becerra V.M., Myatt D.R., Nasuto S.J., and Bishop J.M. Search space pruning and global optimization of multiple gravity assist spacecraft trajectories. *Journal of Global Optimization*, 38:283–269, 2006. doi: 10.1007/s10898-006-9106-0.

Izzo D., Becerra V.M., Myatt D.R., Nasuto S.J., and Bishop J.M. Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. *Journal of Global Optimization*, 38(2):283–296, 2007. ISSN 0925–5001. doi: 10.1007/s10898-006-9106-0.

Jeffrey J.C., Reeds J.A., Wright M.H., and Wright P.E. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

Kemble S. *Interplanetary Mission Analysis and Design*. Springer Verlag / Praxis Publishing Ltd., 2006. ISBN 3-540-29913-0.

Kennedy J. and Eberhart R. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, 1995.

Kim I.S. An algorithm for finding the minimum distance between two ellipses. *Communications of the Korean Mathematical Society*, 21(3):559–567, 2006.

Kirkpatrick S., Jr. C.D. Gelatt, and Vecchi M.P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Lancaster E.R. and Blanchard R.C. A unified form of Lambert's theorem. *NASA technical note TN D-5368*, 1969.

Lanzarini L., Sanz C., Naiouf M., and Romero F. Comparative analysis of the method of assignment by classes in GAVaPS. *Journal of Computer Science and Technology*, March 2000.

Leipold M., Fichtner H., Heber B., Groepper P., Lascar S., Burger F., Eiden M., Niederstadt T., Sickinger C., Herbeck L., Dachwald B., and Seboldt W. Heliopause explorer – a sailcraft mission to the outer boundaries of the Solar system. *Acta Astronautica*, pages 785–796, 2005.

Lim Young Il, Floquet P., and Joulia X. Efficient implementation of the normal boundary intersection (NBI) method on multiobjective optimization problems. *Industrial & Engineering Chemistry Research*, 40(2):648–655, 2001. doi: 10.1021/ie000400v.

Loy J. Conic Sections. Online resource, `http://www.jimloy.com/geometry/conic0.htm`. Last retrieved 20th January, 2010, 2001.

Luersen M.A., Riche R. Le, and Guyon F. A constrained, globalized, and bounded nelder-mead method for engineering optimization. *Journal Structural and Multidisciplinary Optimization*, 27(1–2):43–54, 2004. ISSN 1615-147X (print), 1615-1488 (online). doi: 10.1007/s00158-003-0320-9.

Lyngvi A., Falkner P., and Peacock A. The interstellar heliopause probe technology reference study. *Advances in Space Research*, 35:2073–2077, 2005.

Madonna R.G. *Orbital Mechanics*. Krieger publishing company, 1997. ISBN 0-89464-010-0.

Maleki L. and Prestage J. Spacetime mission: Clock test of relativity at four Solar radii. In Lämmerzahl C., Everitt C.W.F., and Hehl F.W., editors, *Gyros, Clocks, Interferometers...: Testing Relativistic Gravity in Space*, volume 562 of *Lecture Notes in Physics, Berlin Springer Verlag*, pages 369–+, 2001. ISBN 978-3-540-41236-6.

McComas D., Bartolone L., Mills W., and Salgado J.F. IBEX: Interstellar boundary explorer. Online resource, `http://ibex.swri.edu/archive/2006.11.shtml`. Last retrieved 2nd June, 2009.

McConaghy T.T., Debban T.J., Petropoulos A.E., and Longuski J.M. An approach to design and optimization of low-thrust trajectories with gravity assists. *AAS/AIAA Astrodynamics Specialist Conference*, 2001. AAS Paper 01468.

McDonald F. Voyager 1: Messages from the edge. Online resource, last retreived January 23rd, 2010. `http://www.spaceref.com/news/viewpr.html?pid=17883`.

Mckinnon K.I.M. Convergence of the Nelder-Mead simplex method to a nonstationary point. Technical report, SIAM Journal of Optimization, 1998.

Meeus J. *Astronomical Algorithms*. Willmann Bell Inc., second edition, 2005.

Melman J. Trajectory optimization for a mission to Neptune and Triton. Master's thesis, Delft University of Technology, 2002.

Mewaldt R.A. and Liewer P.C. An interstellar probe mission to the boundaries of the heliosphere and neaby interstellar space. NASA Internal Report, 1999.

Michalewicz Z. *Genetic algorithms + data structures = evolution programs*. Springer, New York, 3 edition, 1996. ISBN 3-540-60676-9.

Mihalas D. and Binney J. *Galactic Astronomy*. W.H. Freeman & co., 2 edition, 1981. ISBN 0-7167-1280-6.

Mikkola S. A cubic approximation for Kepler‘s equation. *Celestial Mechanics*, 40:329–334, 1987.

Mishra and Sudhanshu . Some new test functions for global optimization and performance of repulsive particle swarm method. *MPRA*, August 2006. Online Resource, `http://mpra.ub.uni-muenchen.de/2718/`. retrieved Dec. 7th, 2008.

Morbidelli A., Levison H.F., Tsiganis K., and Gomes R. Chaotic capture of Jupiter's trojan asteroids in the early solar system. *Nature*, 435:462–465, 2005. doi: 10.1038/nature03540.

Mundim K.C. Stochastic dynamics through generalized simulated annealing. Online resource, 2009. `http://www.unb.br/iq/kleber/GSA/`. Last retrieved 3rd May.

NASA/JPL . Cassini – equinox mission. NASA mission pages, Online resource, 2009a. `http://saturn.jpl.nasa.gov/`. Last retrieved 14th December, 2009.

NASA/JPL . Solar system exploration – Galileo legacy site. NASA solar system exploration pages, Online resource, 2009b. `http://solarsystem.nasa.gov/galileo/`. Last retrieved 15th December, 2009.

NASA/JPL . NASA – MESSENGER. NASA mission pages, Online resource, 2009c. `http://www.nasa.gov/mission_pages/messenger`. Last retrieved 11th December, 2009.

Nelder J.A. and Mead R. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965. doi: 10.1093/comjnl/7.4.308.

Oldenhuis R.P.S. A many-body flyby mission. In *Proceedings of the S4*, 2009. Internship project performed at ISAS/JAXA.

Papakostas S.N. and Tsitouras C. High phase-lag-order Runge-Kutta and Nyström pairs. *SIAM Journal of Scientific Computaton*, 21(2):747–763, 1999. ISSN 1064–8275. doi: 10.1137/S1064827597315509.

Patel M.R. *Spacecraft Power Systems*. CRC Press, 2005. ISBN 0-8493-2786-5.

Payne W.E. Earth and its electromagnetic environment. Online resource, last retreived January 23rd, 2010. `http://www.altair.org/atmoelec.html`.

Petropoulos A.E. *A shape-based approach to automated, low-thrust, gravity-assist trajectory design*. PhD thesis, Purdue University, W. Lafayette, 2001.

Petropoulos A.E. and Longuski J.M. Shape-based algorithm for automated design of low-thrust, gravity-assist trajectories. *Journal of Spacecraft and Rockets*, 41:787–796, 2004.

Petropoulos A.E., Longuski J.M., and Vinh N.X. Shape-based analytic representations of low-thrust trajectories for gravity-assist applications. *Advances in the Astronautical Sciences*, 103:563–581, 1999.

Piazza E. and Wessen A. Cassini equinox mission. Online resource, last retreived January 23rd, 2010. http://saturn.jpl.nasa.gov/video/videodetails/?videoID=196.

Press W.H., Flannery B.P., Teukolsky S.A., and Vetterling W. T. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

Price C. J., Coope I. D., and Byatt D. A convergent variant of the Nelder-Mead algorithm. *Journal of Optimization Theory and Applications*, 113(1):5–19, 2002. ISSN 0022–3239.

Price K.V., Storn R.M., and Lampinen J.A. *Differential Evolution–A Practical Approach to Global Optimization*. Springer Verlag, 2005. ISBN 978-3-540-20950-8.

Prussing J.E. and Conway B.A. *Orbital Mechanics*. Oxford University Press, New York, 1993.

Reyes-Sierra M. and Coello-Coello C.A. Multi–objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3): 287–308, 2006. ISSN 0973-1873.

Russell C.T. Instrumentation, data processing and data analysis in space physics. Online lecture notes, http://dawn.ucla.edu/personnel/russell/ESS265/. Last retrieved 15th June, 2009, 2008.

Sandford S.A., Aléon J., Alexander C.M.D., Araki T., Bajt S., Baratta G.A., Borg J., Bradley J.P., Brownlee D.E., Brucato J.R., Burchell M.J., Busemann H., Butterworth A., Clemett S.J., Cody G., Colangeli L., Cooper G., d'Hendecourt L., Djouadi Z., Dworkin J.P., Ferrini G., Fleckenstein H., Flynn G.J., Franchi I.A., Fries M., Gilles M.K., Glavin D.P., Gounelle M., Grossemy F., Jacobsen C., Keller L.P., Kilcoyne A.L.D., Leitner J., Matrajt G., Meibom A., Mennella V., Mostefaoui S., Nittler L.R., Palumbo M.E., Papanastassiou D.A., Robert F., Rotundi A., Snead C.J., Spencer M.K., Stadermann F.J., Steele A., Stephan T., Tsou P., Tyliszczak T., Westphal A.J., S.Wirick , Wopenka B., Yabuta H., Zare R.N., and Zolensky M.E. Organics captured from comet 81P/Wild 2 by the Stardust spacecraft. *Science*, 314 (5806):1720–1724, 2006. doi: 10.1126/science.1135841.

Schilling G. *De Jacht op Planeet X*. Fontaine Uitgevers, 2008. ISBN 9789059561946.

Schütze O., Vasile M., Junge O., Dellnitz M., and Izzo D. Designing optimal low thrust gravity assist trajectories using space pruning and a multi-objective approach. *Engineerin Optimization*, 0(0):1–22, June 2007.

Sebah P. and Gourdon X. Newton's method and high order iterations, 2001. http://numbers.computation.free.fr/Constants/Algorithms/newton.html, retrieved on Dec. 19th, 2008.

Sendin J.H., Otero-Muras I., Alonso A.A., and Banga J.R. Improved optimization methods for the multiobjective design of bioprocesses. *Industrial & Engineering Chemistry Research*, 45(25):8594–8603, 2006. doi: 10.1021/ie0605433.

Seywald H. Trajectory optimization based on differential inclusion. Technical report, NASA Langley Research Center, 1993. NASA contractor Report 4501, under contract NAS1-18935.

Shepperd S.W. Universal Keplerian state transition matrix. *Celestial Mechanics*, 35:129–144, 1984. doi: 0008-8714/85.15.

Sims J.A. and Flanagan S.N. Preliminary design of low-thrust interplanetary missions. *AAS/AIAA Astrodynamics Specialist Conference*, August 1999.

Singer S. and Singer S. Efficient implementation of the Nelder-Mead search algorithm. *Applied Numerical Analysis and Computational Mathematics*, 1(3):524–534, 2004.

Smithsonian Astrophysical Observatory Division III of the IAU. Minor planet center. Online resource, `http://www.cfa.harvard.edu/iau/mpc.html`, 2009. Database downloadable from `ftp://mpcorb:Ceres@ftp.astro.cz/MPCOrb.DAT`. Last retrieved on Jan 05$^{\text{th}}$, 2009.

Srinivas N. and Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2:221–248, 1994.

Stone E.C., Cummings A.C, McDonald F.B., Heikkila B.C., Lal N., and Webber W.R. Voyager 1 explores the termination shock region and the heliosheath beyond. *Science*, 309: 2017–2019, 2005. doi: 10.1126/science.1117684.

Stoyanova S. and Dunbar B. NASA's 'impressionist' spacecraft to view solar system's invisible frontier. Online resource, `http://www.nasa.gov/mission_pages/ibex/index.html`. Last retrieved 2$^{\text{nd}}$ Jun, 2009, 2009.

Sun W. and Yuan Y-X. *Optimization Theory and Methods – Nonlinear Programming*. Springer, 2006. ISBN 978-0-387-24975-9.

Vallado D.A. *Fundamentals of Astrodynamics and Applications*. Microcosm Press and Kluwer Academic Publishers, second edition, 2001. with contributions by W.D. McCain.

Vasile M., Schütze O., Junge O., Radice G., and Dellnitz M. Spiral trajectories in global optimisation of interplanetary and orbital transfers. Technical report, ESA/ACT, 2006.

Vinkó T., Izzo D., and Bombardelli C. Benchmarking different global optimisation techniques for preliminary space trajectory design. 57$^{\text{th}}$ International Astronautical Congress, Hyderabad, India, 2007.

Stryk von O. and Bulirsch R. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37:357–373, 1992.

Wakker K.F. *Astrodynamics I & II*. Delft University of Technology, 2007. Lecture notes for the course AE4-874.

Walker R. ESA and ANU make space propulsion breakthrough. Online resource, `http://www.esa.int/esaCP/SEMOSTG23IE_index_0.html`. Last retrieved 3$^{\text{rd}}$ June, 2009, 2006.

Walker R., Izzo D., and Fearn D. Missions to the edge of the Solar system using a new advanced dual-stage gridded ion thruster with very high specific impulse. In *ISTS 2006-k-35*, 2006. 25$^{\text{th}}$ International Symposium on Space Technology and Science.

Weisstein E. Newton-cotes formulae. Online resource, http://mathworld.wolfram.com/Newton-CotesFormulas.html. Last retrieved 23rd June, 2009, 2009.

Wertz J.R. *Mission Geometry; Orbit and Constellation Design and Management*. Microcosm Press / Kluwer Academic Publishers, 2001. ISBN 1-881883-07-8.

Whatmore R., Greicius T., and Perez M. Radioisotope power systems for NASA. NASA fact sheet, Online resource, 2009. http://www.jpl.nasa.gov/news/fact_sheets/radioisotope-power-systems.pdf. Last retrieved 9th September, 2009.

Wikipedia . Euler angles. Online resource, http://en.wikipedia.org/wiki/Euler_angles. File retrieved Jan. 13th, 2009, 2009.

Wikipedia.org . Ion engine. Online Resource, http://en.wikipedia.org/wiki/Ion_engine. Last retrieved 17th September, 2009a.

Wikipedia.org . New Horizons. Online Resource, http://en.wikipedia.org/wiki/New_Horizons. Last retrieved 09th September, 2009b.

Wikipedia.org . Eccentric anomaly. Online resource, 2009c. http://en.wikipedia.org/wiki/Eccentric_anomaly. Last retrieved 1st May.

Wikipedia.org . Orbital elements. Online resource, 2009d. http://en.wikipedia.org/wiki/Orbital_elements. Last retrieved 1st May.

Wikipedia.org . Kuiper belt. Online Resource, http://en.wikipedia.org/wiki/Kuiper_belt. Last retrieved 26th May, 2009e.

Wikipedia.org . Voyager 1. Online Resource, http://en.wikipedia.org/wiki/Voyager_1. Last retrieved 15th June, 2009f.

Wilke D.N. Analysis of the particle swarm optimization algorithm. Master's thesis, University of Pretoria, 2005. Available online, http://upetd.up.ac.za/thesis/available/etd-01312006-125743/. retrieved on Jan. 7th, 2009.

Witte M., Banaszkiewicz M., and Rosenbauer H. Recent results on the parameters of the interstellar helium from the Ulysses/GAS experiment. *Space Science Reviews*, 78(1):289–296, 1996. doi: 10.1007/BF00170815.

Wolff Peter. GPS orbits. Online resource, http://www.wolffdata.se/gps/gpshtml/anomalies.html. Last retrieved 15th Jun, 2009, 2009.

Wolff S. A local and globalized, constrained and simple bounded Nelder-Mead method. Online Resource, www.uni-weimar.de/~wolff3/software/BNM_GBNM.pdf. Last retrieved 11th May, 2009, 2004.

Xiaohui H. and Eberhart R. Multiobjective optimization using dynamic neighborhood particleswarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation*, volume 2, pages 1677–1681, 2002. doi: 10.1109/CEC.2002.1004494.

Yeomans D.K. and Chamberlin A.B. HORIZONS system. Online Resource, 2009. `http://ssd.jpl.nasa.gov/?horizons`. Last retrieved Dec. 31st, 2009.

Zandbergen B.T.C. *Thermal Rocket Propulsion.* Delft University of Technology, Faculty of Aerospace Engineering, 2004. Lecture notes AE4-S01.

A Fundamental Concepts

In this chapter the fundamental concepts used throughout this thesis will be briefly discussed. Section A.1 treats the motion of objects subject to a Newtonian gravity field. The resulting motion of such bodies can be described by conic sections, and (most) equations involved with this type of motion are listed in this section. Section A.2 gives some additional useful definitions, the purpose of which is to avoid confusion which may arise in the treatise of more complicated theory. Finally, section A.4 explains the general GAM. This type of manoeuver allows a spacecraft to increase its total energy considerably, while using only a small amount of propellant, or none at all. This manoeuvre will again serve as the basis for more advanced concepts later on in the thesis.

A.1. Motion in a Newtonian Gravity Field

Isaac Newton (1643-1727) postulated that every massive body exerts a force on every other body. He called this force of mutual attraction between two bodies the force of *gravity*, and Newton's original model is still widely used today, even though it has long been superseded on theoretical grounds. For the purpose of orbit design (which is the main subject of this thesis), Newton's relatively simple model of gravity is more than sufficient to describe the motion of spacecraft under the influence of the gravitational attraction of the Sun and planets.

The motion of a body with negligible mass and dimensions (the *point mass*), which moves in a Newtonian gravity field of a **single** gravitating body (the *central body*), is fully described by the differential equation

$$F = m\frac{d^2\boldsymbol{r}}{dt^2} = -\frac{\mu \boldsymbol{r}}{r^3}, \tag{A.1}$$

where $\boldsymbol{r}$ is the radius vector of the point mass, $r = |\boldsymbol{r}|$ is its magnitude, t is time and $\mu = GM$ is the *standard gravitational parameter* of the central body, equal to the product of its mass

M and the universal gravitational constant $G = 6.67300 \times 10^{-11}$ m^3 kg^{-1} s^{-2}.

Such motion is usually called *two-body motion* (since it involves only two bodies), and problems that employ such a model are called *two-body problems*. As Johannes Kepler showed in 1605 (which Newton later *proved* with the above postulate) is that the shape of the trajectory of the massless body in a two-body model is a *conic section* – either an *ellipse*, a *parabola* or a *hyperbola*. It can be shown that a parabolic trajectory is a limiting case and therefore physically impossible to attain[1]. For this reason, only the elliptic orbit and the hyperbolic trajectory are treated in detail, and the parabolic trajectory will only be mentioned for completeness. All theory in this section (with derivations) can be found in Wakker [2007].

A.1.1 General Conic Sections

All types of conic sections relevant in celestial mechanics are displayed in Figure A.1. Mathematically, these types are discriminated through their *eccentricity* e. This is a measure for the amount by which the shape of the section deviates from a perfect circle. For the circle itself, $e = 0$.

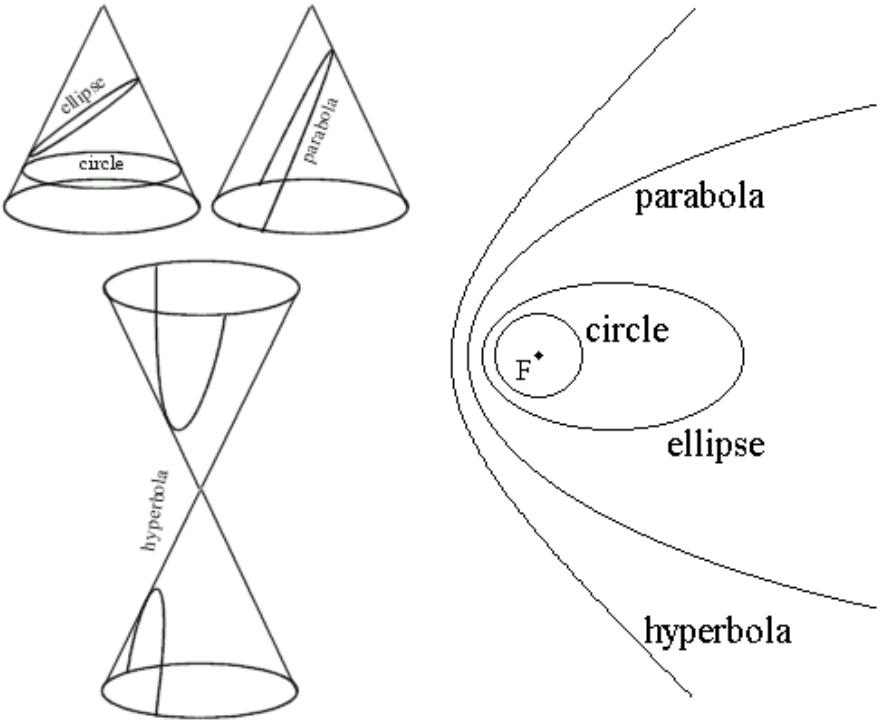


Figure A.1 All types of conic sections. Adopted from [Loy 2001].

The collection of points on a conic section can most easily be described in polar coordinates,

[1] A parabolic trajectory implies that the orbit's eccentricity must be *exactly* equal to one; the required "infinite accuracy" is a physical impossibility.

with the focus F as the origin:

$$r = \frac{p}{1 + e\cos\theta} \tag{A.2}$$

with r the radius, p the semi-latus rectum and θ the corresponding polar angle (see also Figure A.2(a)). It can be shown that

$$p = a(1 - e^2) = \frac{H^2}{\mu},$$

with $H = |\boldsymbol{H}| = |\boldsymbol{r} \times \boldsymbol{V}|$ the angular momentum. From analytical mechanics it is known that if no moments act on a body, its linear and angular momentum are integrals of motion. This implies that the vector $\boldsymbol{H}$ is constant in both magnitude and direction for any body in a simple orbit around another body. In turn, this implies that the motion will always stay in the same plane. For any motion describing a trajectory along a conic section, this plane is referred to as the *orbital plane*[2].

Also from analytical mechanics, the total energy per unit of mass of the body in orbit can be written as the sum of its potential and kinetic energies:

$$\epsilon = \frac{V^2}{2} - \frac{\mu}{r}. \tag{A.3}$$

It can be shown that

$$\epsilon = -\frac{\mu}{2a}$$

with a the *semi-major axis* of the conic section (see section A.1.2). These equations lead to the *vis-viva* equation:

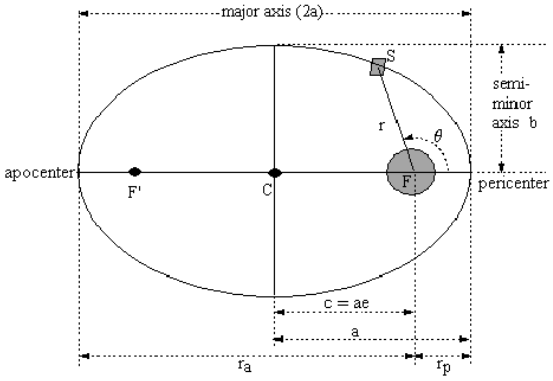
$$V^2 = \mu\left(\frac{2}{r} - \frac{1}{a}\right) \tag{A.4}$$

explicitly giving the speed at any point $V(r)$ as a function of r.

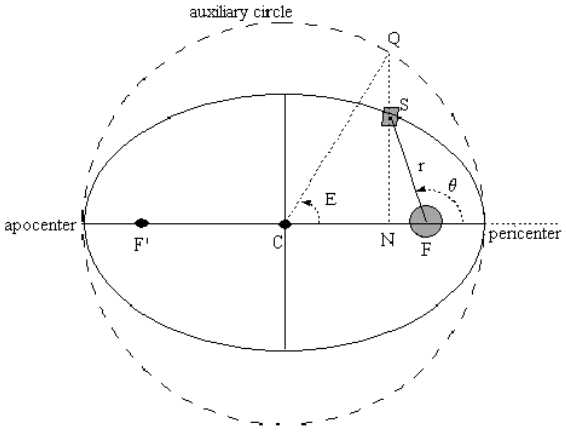
A.1.2 Elliptic Orbit

As the name implies, the elliptic orbit is a *closed orbit*, which means that it will repeat itself after a certain amount of time. The elliptic orbit (and all common definitions) is shown in Figure A.2. For all elliptic orbits, the eccentricity lies in the range $0 \leq e < 1$.
Many useful quantities can be derived by analyzing the special case $e = 0$, e.g., a *circular* orbit. Taking a circular orbit with constant radius R, the closely related quantities *circular period* T_c and *angular motion* n_c can easily be derived with basic physics:

[2]The fact that a body in orbit follows a conic section also implies that the motion is confined to a plane. This is what Johannes Kepler implicitly showed by formulating his three laws. Newton later *proved* the statement using his law of conservation of angular momentum.



(a) All useful definitions for the elliptic orbit, in the conventional notation.



(b) Definition of eccentric anomaly (E) and true anomaly(θ).

Figure A.2 The elliptic orbit, and all common definitions. Adopted from [Wolff 2009; section "Notes on GPS satellite orbits"]

$$n_c = \sqrt{\frac{\mu}{R^3}} \quad \text{and} \tag{A.5}$$

$$T_c = 2\pi\sqrt{\frac{R^3}{\mu}} = \frac{2\pi}{n_c}. \tag{A.6}$$

In the circular case, also a simple expression can be found for the speed of the massless body anywhere in the orbit:

$$V_c = \sqrt{\frac{\mu}{R}}. \tag{A.7}$$

The circular period describes the amount of time required for the massless body to complete one full circular orbit. From the definition of the angular motion, it can be seen that it defines the amount of radians per second the body traverses over the circle. It can be shown that the same two relations also hold for an elliptic orbit, provided that $R \to a$:

$$n = \sqrt{\frac{\mu}{a^3}} \quad \text{and} \tag{A.8}$$

$$T = 2\pi\sqrt{\frac{a^3}{\mu}} = \frac{2\pi}{n}, \tag{A.9}$$

where a is the semi-major axis. For the elliptic case, n is defined as the *mean* angular motion, or simply *mean motion*. The difference in terminology arises from the fact that the angular motion in the circular case is a constant, whereas in the elliptic case, the actual angular motion varies over time but the *mean motion* is an "averaged" value, and thus constant.

The circular velocity V_c can be regarded as a *local* property, e.g., a circular velocity V_c can be assigned to *any* point in space, also for elliptic orbits. The *local* circular velocity is the velocity a massless body would need to have in order to be in a circular orbit around the central body. This interpretation proves quite useful later on.

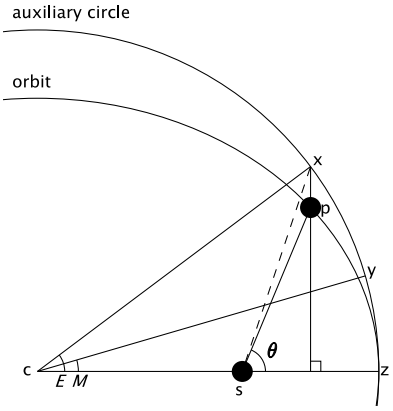


Figure A.3 The definition of the eccentric anomaly E and the mean anomaly M. From Kepler's laws it may be derived that M is the angle for which the area of the triangle $c-z-y$ is equal to that of the elliptic section $s-z-p$. Adopted from [Wikipedia.org 2009c].

The classical way to relate the polar angle θ to the time t, is by looking at the auxiliary circle (see Figures A.2(b) and A.3). In these figures, the quantity E is called the *eccentric anomaly* and M the *mean anomaly*. The "mean" in mean anomaly again comes from the fact that the *true* polar angle θ (usually called the *true anomaly*) is a non-linear transcendental function of time, whereas the mean anomaly M increases linearly over time. This is reflected in the relations

$$\tan\frac{E}{2} = \sqrt{\frac{1-e}{1+e}}\tan\frac{\theta}{2}, \tag{A.10}$$

$$\begin{aligned} M &= E - e\sin E \\ &= M_0 + nt, \end{aligned} \tag{A.11}$$

the latter of which shows that M indeed increases linearly over time, since the mean motion n is defined as constant. Note that the angles E and θ can be found uniquely by virtue of the `atan2`-function:

$$E = 2\cdot\texttt{atan2}\left(\sqrt{1-e}\sin\frac{\theta}{2},\ \sqrt{1+e}\cos\frac{\theta}{2}\right), \tag{A.12}$$

$$\theta = 2\cdot\texttt{atan2}\left(\sqrt{1+e}\sin\frac{E}{2},\ \sqrt{1-e}\cos\frac{E}{2}\right). \tag{A.13}$$

The mean anomaly serves as a coupling between the position of the massless body and time, which is essential to deal with problems of this nature. The importance of the discovery of this last relation is underlined by naming it after its discoverer – the equation is best known as *Kepler's equation*. Note that when the true anomaly θ is known, it is easy to compute the mean anomaly M from Kepler's equation. The reverse operation is unfortunately not so trivial, since the eccentric anomaly E can not be isolated in Equation A.11 – the equation is transcendental. Thus solving this equation requires a numerical approach. A more detailed treatise on this subject is given in appendix F.1.

There is a plethora or other quantities which can be derived for an elliptic orbit. Some of the most useful are (see also Figure A.2(a))

b	$= a\sqrt{1-e^2}$	semi-minor axis
r_p	$= a(1-e)$	pericenter distance
r_a	$= a(1+e)$	apocenter distance
p	$= a(1-e^2)$	
	$= b^2/a$	semi-latus rectum
	$= H^2/\mu$	
$\dot{r}$	$= \sqrt{\frac{\mu}{p}} e \sin\theta$	
	$= \frac{e\sin\theta}{H}$	radial speed
$\tan\gamma$	$= \frac{e\sin\theta}{1+ecos\theta}$	
	$= \frac{e\sin E}{\sqrt{1e^2}}$	flight path angle

A.1.3 Parabolic Trajectory

When the speed of a body at some fixed location location is increased, it follows from Equation A.3 that its total energy comes closer to zero. A result of this is that the eccentricity of its orbit increases, and the orbit's apocenter r_a is moved further away from the central body. The distance of the apocenter can essentially grow without bound, and when a certain limit is reached ($e = 1$), the orbit's apocenter r_a lies *infinitely* far away from the central body. In such cases, the total energy is equal to zero (the magnitude of the kinetic energy is *equal* to that of the potential energy in all points), and the massless body is no longer bound to the central body; the body can essentially *escape* from the central body's gravitational pull.

Trajectories with energies $\epsilon \geq 0$ are therefore called *escape trajectories*. The limiting case ($e = 1$, $\epsilon = 0$) is called a *parabolic trajectory*. In the parabolic case, the massless body can *just* escape from the central body and move infinitely far away without ever returning to the central body. However, the speed of the massless body decreases asymptotically to zero as the distance grows to infinity. This is because the speed of a massless body in a parabolic trajectory is everywhere equal to the *local escape speed*:

$$V_{\text{esc}} = \sqrt{\frac{2\mu}{r}} = V_c\sqrt{2}, \tag{A.14}$$

with V_c the local circular velocity defined in Equation A.7. The meaning of the local escape velocity is that if a body is given precisely that speed at that point, it will *just* escape the central body[3]. It can be quite cumbersome to solve Kepler's equation in cases where $e \approx 1$. In such cases, convergence of the numerical procedure can become a serious problem if a naive implementation is used. More details on this issue are given in appendix F.1.

A.1.4 Hyperbolic Trajectory

In escape trajectories where $e > 1$, the massless body can move infinitely far away from the central body *and* have a non-vanishing velocity. This type of trajectory constitutes the second class of escape trajectories and is called a *hyperbolic trajectory*. A general hyperbolic trajectory (and most common definitions) is shown in Figure A.4.

[3]Provided the speed is given to the body such that it will not collide with the central body, of course.

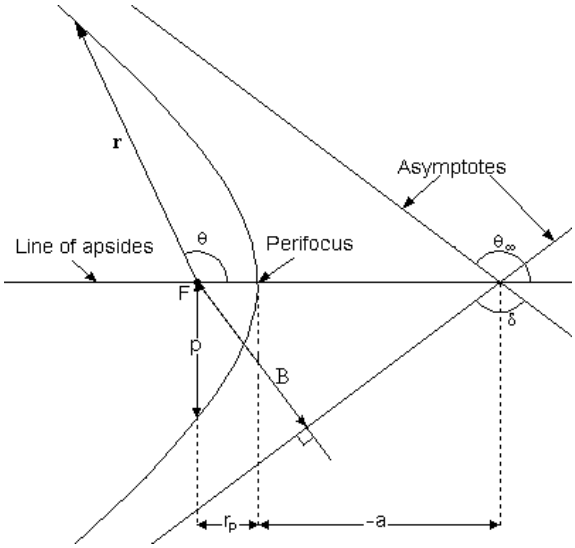


Figure A.4 A general hyperbolic trajectory, and most common definitions. Adopted from [Berthoud 1997]

In basic analytical mechanics, bodies are said to be *free* or *unbound* to a system when that body has positive total energy with respect to that system. Hyperbolic trajectories are completely unbound to the central body. It would be convenient to have a set of definitions for all trajectories along conic sections that does not differ too much between the different types, and is consistent with conventional mechanics. Central to the governing equations of a conic section is Equation A.3, so it is demanded that this equation holds for elliptic, parabolic and hyperbolic trajectories. That implies that the semi-major axis in a hyperbolic trajectory must be assigned a *negative* value, to be consistent with the *positive* energy of an unbound trajectory.

The equations governing elliptic orbits break down in case $e > 1$ or $\epsilon > 0$, since many of the equations would give complex-valued results. However, when interpreting a hyperbolic trajectory as an ellipse along the *complex plane*, this is not necessarily a problem. In fact, making the substitution $F = iE$, with i the complex unit, and recognizing that

$$\cos E = \frac{e^{+iE} + e^{-iE}}{2} = \frac{e^{+F} + e^{-F}}{2} = \cosh F, \tag{A.15}$$

$$\sin E = \frac{e^{+iE} - e^{-iE}}{2i} = \frac{e^{+F} - e^{-F}}{2i} = -i \sinh F, \tag{A.16}$$

the half-angle relation and Kepler's equation for hyperbolic trajectories become

$$\tanh \frac{F}{2} = \sqrt{\frac{e-1}{e+1}} \tan \frac{\theta}{2}, \tag{A.17}$$

$$\begin{aligned} H &= e \sinh F - F \\ &= H_0 + nt, \end{aligned} \tag{A.18}$$

where F and H are the hyperbolic counterparts of the eccentric anomaly E and the mean anomaly M, respectively. This is indeed consistent with physical reality, provided that

$$n = \sqrt{\frac{\mu}{-a^3}}, \tag{A.19}$$

since the semi-major axis was assigned a negative value. Note that the angles F and θ can still be found uniquely:

$$\theta = 2 \cdot \texttt{atan2}\left(\sqrt{e+1}\sinh\frac{F}{2}, \sqrt{e-1}\cosh\frac{F}{2}\right), \tag{A.20}$$

$$F = 2 \cdot \operatorname{arctanh}\left(\sqrt{\frac{e-1}{e+1}}\tan\frac{\theta}{2}\right), \tag{A.21}$$

the latter of which is unique because the arctanh–function increases monotonously and has a one-to-one correspondence with all angles $0 \leq \theta \leq 2\pi$.

When the point mass is allowed to move indefinitely along a hyperbolic trajectory, its speed will decrease asymptotically to a non-zero value when the distance to the central body tends to infinity. This non-zero speed is a characteristic parameter of the hyperbolic trajectory, and is constant both before and after the approach to the central body. It is called the *hyperbolic excess speed*, or V_∞. The subscript ∞ comes from the fact that the speed of the point mass will have decreased to this value only when it has moved infinitely far away from the central body.

With this definition, a very important relationship can be derived for the speed in any point in a hyperbolic trajectory:

$$V^2 = V_\infty^2 + V_{\text{esc}}^2, \tag{A.22}$$

which will be used very often in further analyses.

A.1.5 Representations of Position and Velocity

The above relations follow from orbits and trajectories in two dimensions. Any real body moves in three dimensions, and some methods must be provided to represent its position. Normally, not only the position, but also the velocity of a body is required to make any meaningful prediction about future positions of that body, so these *two* quantities are usually provided together. When no other forces besides gravity are present, there are thus 6 degrees of freedom, and also 6 coordinates to represent them.

There are three common ways to represent these quantities – by means of the Euclidian *state vector*, by *Orbital Elements* or (somewhat less common) by a *parametric representation.*

Euclidian State Vector

This is the easiest and most intuitive way of representing the coordinates and velocity of a body in space. Given any Euclidian coordinate system $x - y - z$, the Euclidian state vector

simply groups all values in a single vector:

$$\text{state vector} \equiv \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \boldsymbol{\chi} \tag{A.23}$$

Orbital Elements

These elements constitute the six *classical orbital elements*, or sometimes *Keplerian elements*, as introduced by Kepler. They are (see also Figure A.5)

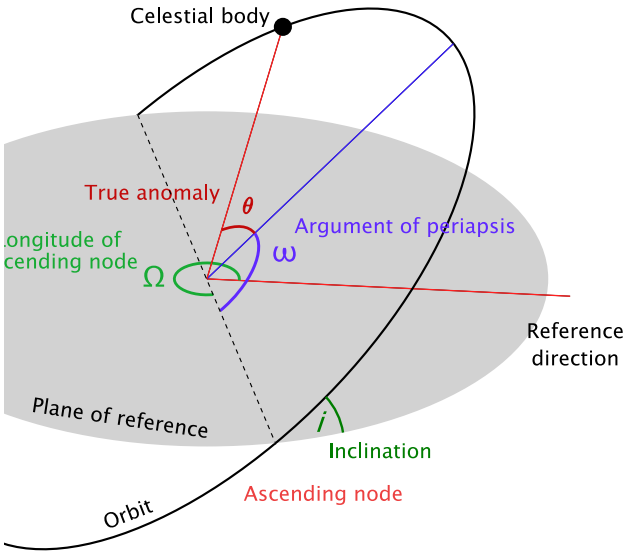


Figure A.5 The six classical orbital elements. Adopted from [Wikipedia.org 2009d].

a – **Semi-major Axis** The definition for this quantity is different for elliptic orbits than for hyperbolic trajectories. For elliptic obits, the semi-major axis is simply half of the long axis of the ellipse. For hyperbolic trajectories, it is defined as the *negative* of the distance between the pericenter and the center.

e – **Eccentricity** A convenient measure for the amount a conic section deviates from a circle. For elliptic orbits, $0 \leq e < 1$, for hyperbolic trajectories, $e > 1$.

i – **Inclination** The angle between the orbital plane and the reference plane, measured at the ascending node and taken positive along the direction of motion.

Ω – **Longitude of the Ascending Node** Angle enclosed between the chosen X–axis and the line of nodes towards the ascending node. When Ω describes an orbit **around the Earth**, is is usually called the *Right Ascension* of the ascending node – a term which originates from the concept of the celestial sphere used in astronomy.

ω – **Argument of Pericenter** Angle between the ascending node and the pericenter, both from the origin (central body at F).

θ_0 – **true anomaly** Angle from the pericenter to the position of the point mass in its orbit, both from the origin (central body at F). $\theta_0 = \theta(t_0)$ indicates the value of this angle at some initial time t_0, which is usually taken equal to the time of the last pericenter passage. For all other times t, the corresponding value for θ van be found via Equations A.11 and A.18.

Parametric Representation

Any ellipse in 3-space may be represented by a single parameter E. The parametric presentation of a 3-dimensional ellipse makes it possible to write its coordinates as a simple scalar function with a single argument E:

$$\begin{aligned} \boldsymbol{x} &= f(E) = \boldsymbol{C} + a\boldsymbol{u}\cos E + b\boldsymbol{v}\sin E \\ &= [\boldsymbol{C}\ a\boldsymbol{u}\ b\boldsymbol{v}][1\ \cos E\ \sin E]^T \end{aligned} \tag{A.24}$$

with $\boldsymbol{C}$ the Cartesian coordinates of the center, a the semi-major axis, b the semi-minor axis, $\boldsymbol{u}$ a unit vector towards pericenter and $\boldsymbol{v}$ a unit vector perpendicular to $\boldsymbol{u}$ along the direction of motion. Thus, filling in any value for E results in a vector of Cartesian coordinates without *any* intermediate steps, once the vectors $\boldsymbol{C}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are known.

The velocity in this representation is:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \dot{f}(E) = -a\boldsymbol{u}\dot{E}\sin E + b\boldsymbol{v}\dot{E}\cos E \\ &= [\boldsymbol{0}\ a\boldsymbol{u}\ b\boldsymbol{v}][1\ -\sin E\ \cos E]^T \cdot \dot{E} \end{aligned} \tag{A.25}$$

where

$$\dot{E} = \frac{\dot{M}}{1 - e\cos E} = \frac{n}{1 - e\cos E},$$

and $\dot{\boldsymbol{x}} = [\dot{x}\ \dot{y}\ \dot{z}] = [V_x\ V_y\ V_z]$, as before. Therefore, the whole Euclidian state vector can be represented by

$$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & a\boldsymbol{u} & b\boldsymbol{v} \\ \boldsymbol{0} & b\dot{E}\boldsymbol{v} & -a\dot{E}\boldsymbol{u} \end{bmatrix} \begin{bmatrix} 1 \\ \cos E \\ \sin E \end{bmatrix} \tag{A.26}$$

The parameter E in this representation is defined as the angle between $\boldsymbol{u}$ and the point of interest on the ellipse, both starting from the center $\boldsymbol{C}$. Thus it is the *central* angle and thus indeed the orbit's *eccentric* anomaly. This representation thus is a *coupling* between the previous two representations. One obvious benefit of this is that it avoids having to convert E to θ if only Cartesian coordinates are needed. Using this representation therefore improves computation time significantly when the spacecraft's Cartesian coordinates need to be known at many points in a given orbit.

The same idea, but then applied to a hyperbolic trajectory, results in

$$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & -a\boldsymbol{u} & b\boldsymbol{v} \\ \boldsymbol{0} & b\dot{F}\boldsymbol{v} & +a\dot{F}\boldsymbol{u} \end{bmatrix} \begin{bmatrix} 1 \\ \cosh F \\ \sinh F \end{bmatrix}, \tag{A.27}$$

where

$$\begin{aligned} b &= -a\sqrt{e^2-1} \\ \dot{F} &= \frac{n}{e\cosh F - 1} \end{aligned}$$

Conversions Between Representations

Some calculations are easily carried out in one representation, and much more difficult in another. Or, some properties of the dynamics are much more apparent when expressed in one representation than another. For these and many more reasons, these *three* different representations must be used interchangeably. Therefore, one representation must be converted into another quite frequently. These coordinate conversions are somewhat lengthy, so they have been included in appendix F for completeness.

A.2. Useful Definitions

A.2.1 Speed and Velocity

In most fields in physics, the *speed* V of a body with respect to a chosen reference is the *Euclidian distance* that body travels per unit of time. Because the Euclidian distance d is defined by the Pythagorean theorem by

$$d = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2},$$

where $[xyz]_1$ and $[xyz]_2$ are any two arbitrary coordinates, the speed V of a body does not depend on the direction in which it is moving. Quite often however it is also useful to have knowledge about the direction of motion of a body with nonzero speed. Therefore, a second quantity called *velocity* is used. The velocity of a body is just a measure of its speed in the three Euclidian directions:

$$\text{velocity} \equiv \begin{bmatrix} \text{speed in } x\text{-direction} \\ \text{speed in } y\text{-direction} \\ \text{speed in } z\text{-direction} \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \boldsymbol{V}.$$

Thus, velocity is by definition a *vectorial* quantity, whereas its speed is a *scalar* quantity. Translated to orbital dynamics, this means

$$\begin{aligned} V &= |\boldsymbol{V}| = |\dot{\boldsymbol{r}}| \\ &= \sqrt{(\dot{x}_2-\dot{x}_1)^2 + (\dot{x}_1-\dot{y}_1)^2 + (\dot{z}_2-\dot{z}_1)^2} \end{aligned} \tag{A.28}$$

In surprisingly much of the literature on subjects related to orbital dynamics, these quite basic concepts are used *interchangeably*, and thus *incorrectly*. Throughout this thesis these concepts are used as defined above.

A.2.2 Sphere of Influence

When a spacecraft travels from planet to planet frequently during a mission, the central body around which it orbits changes from planet to Sun to planet to Sun etc. It would be convenient to have a method to determine the approximate location in space where the equations of motion should be shifted from one central body to the next.

In first-order analyses, the concept of the SOI can be used to determine just that. The SOI is defined as an imaginary sphere around a body that delimits the region of space where that body's gravitational attraction is dominant, with respect to some further removed reference body. That is, inside the SOI, that body can be considered to be the central body, and all other bodies as mere perturbing forces. On the surface of the SOI, the gravitational pull from the central body is approximately equal to that from the reference body[4], which is how its equation is derived. Its equation can be stated as [Wakker 2007; adopted from equation 4.22]:

$$R_{\text{SOI}} \approx a_P \left(\frac{M_P}{M_S}\right)^{2/5} = a_P \left(\frac{\mu_P}{\mu_S}\right)^{2/5}, \tag{A.29}$$

where the subscripts P and S stand for *primary* and *secondary*, respectively ("primary" is the reference body and "secondary" the body around which the SOI is being defined), and M is the mass, μ the central body's standard gravitational parameter, and a the semi-major axis of the central body with respect to the primary body.

The SOI's of all the planets (and several larger minor planets) with respect to the Sun are listed in Table A.1 – in this table, the radii are expressed in both millions of km, and number of times the radius of the corresponding (minor) planet.

A.3. Tsiolkovskii's Equation

A fundamental problem in rocket science is how to find the final speed of a given rocket. This is difficult, because Newton's equations do not apply to an object with variable mass. Konstantin Eduardovich Tsiolkovskii (1857–1935), considered to be the founding father of spaceflight, was the first to correctly derive the relation between a rocket engine's parameters, and how much change in speed the engine can deliver. His now famous equation reads

$$\Delta V = V_e \ln \frac{m_0}{m_e}, \tag{A.30}$$

[4]The SOI is only approximately a sphere. In the derivation of the equation Equation A.29 there are a few assumptions and approximations that degenerate the true equations into those describing a perfect sphere. But since this is a method used only in first order analyses, those approximations introduce negligible errors compared to those made by general first-order trajectory analysis.

Table A.1 Spheres of Influence for the planets, and several of the larger minor planets expressed in both 10^6 km and number of times the body's (mean) radius.

Planet/MP	R_{SOI} (10^6 km)	R_{SOI} (R_{Body})
Mercury	0.112	46.075
Venus	0.616	101.83
Earth	0.924	144.89
Mars	0.577	169.65
Jupiter	48.24	674.73
Saturn	54.62	906.36
Uranus	51.70	2022.8
Neptune	86.40	3488.8
Ceres	0.075	895.79
Pallas	0.050	483.61
Juno	0.015	115.01
Vesta	0.039	487.99
Astraea	0.002	10.994
Hebe	0.012	87.211

where ΔV is the rocket's change in speed, V_e is the exhaust speed of the rocket's propellant, and m_0 and m_e are the initial and final masses of the rocket, that is, with and without propellant. These latter masses are usually also referred to as the *wet mass* and *dry mass*, respectively.

To avoid confusion when two different systems of units are used during the design of a single rocket, this equation is usually rewritten as

$$\Delta V = g_0 I_{\mathrm{sp}} \ln \Lambda, \tag{A.31}$$

where g_0 is the standard gravitational acceleration at sea level, I_{sp} is the *specific impulse*, and $\Lambda = m_0/m_e$ is the *mass ratio*. The major benefit of this notation is that the dimension of the parameter I_{sp} is seconds, and Λ is dimensionless. Seconds are used as a unit of time in *any* unit system (SI, Imperial units, ...), so that I_{sp} is a rocket performance measure that can be used throughout the world, without confusion[5].

A.4. Gravity-assist Manoeuver

Although there are many names for it[6], the technically most correct term is probably the GAM. This manoeuver allows a spacecraft to gain considerable amounts of orbital energy without expelling any propellant. It is in fact the only practical way to go to the outer

[5]The actual unit-conversion takes place when I_{sp} is multiplied by g_0, which is given in the units of preference, e.g., $g_0 = 9.80665$ m s^{-2} $= 32.17405$ ft s^{-2} $= \ldots$

[6]This concept is so common nowadays that it has a myriad of names in orbit designer's lingo. It almost seems like a sport to come up with the most original name for the concept. Frequently used (even in the literature) are *Swing-by* and *Gravitational Slingshot*, but in "everyday" conversation terms like *cosmic billiards*, *gravity-well exploit*, or even *planetary pool* are frequently heard.

planets, and has been used frequently since it was first demonstrated by the Mariner 14 spacecraft in 1974.

The spacecraft's gain in orbital energy comes from the exchange of angular momentum between the swingby-planet and the spacecraft. The spacecraft is directed around the planet such that it "steals" some of the planet's angular momentum from its orbit around the Sun. An oversimplified example is illustrated in Figure A.6. In this figure, all motion is assumed to occur in one dimension only. This oversimplified example nicely illustrates the principle of a GAM – simply because the planet has a velocity at any point in its orbit around the Sun can the spacecraft's speed be increased with *twice* the planet's orbital speed.

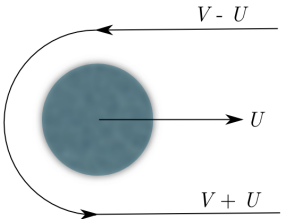
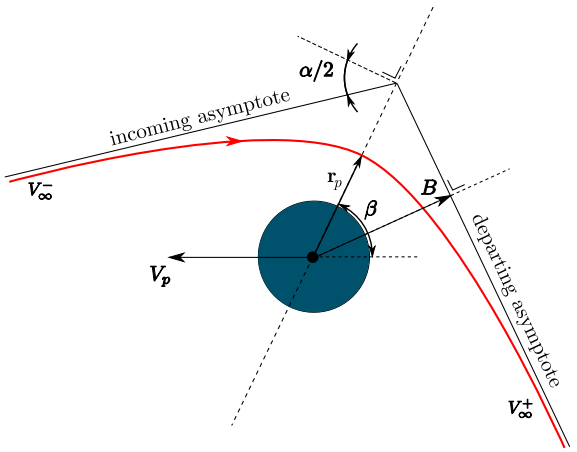


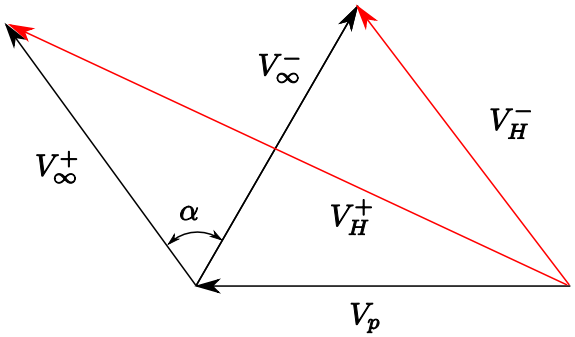
Figure A.6 The Gravity-Assist Manoeuver in one dimension.

A.4.1 Two Dimensions

Since the objective is to go from one planet to another, the trajectory the spacecraft follows around the planet can not be a closed orbit – it must be a hyperbolic escape trajectory. As such, the speeds indicated by $V+U$ and $V-U$ in Figure A.6 indeed indicate the body-centered hyperbolic excess velocities.



(a) Overview of a GAM in two dimensions. Since the hyperbolic trajectory completely lies in a plane, this diagram also serves as the basis for the three-dimensional GAM.



(b) Relevant vectors. Note that the quantity V_H^+ is considerably larger than V_H^-.

Figure A.7 Geometry of the General Gravity-Assist Manoeuver.

A more realistic picture for a GAM is shown in Figure A.7(a). The most important quantities indicated in this figure are the *turn angle* α, the pericenter distance r_p and the *impact parameter* B. All relevant velocities and angles may be abstracted into Figure A.7(b), which shows that the heliocentric velocity change ($\Delta V = V_H^+ - V_H^+$) only depends on the angle of rotation of the V_∞-vectors, α. When viewed from the Sun, it is as if the planet has induced an instantaneous rotation and magnitude change of the heliocentric velocities, resulting in an

orbit with significantly different energy than before the encounter with the planet. Note that both an energy *increment* and *decrement* are possible.

Usually, in back-of-the-envelope calculations, the minimum flyby distance r_p in Equation A.32 is simply taken equal to the planet's (mean) radius R_p, which is known and thus easy to insert into the equations. However, this can not be achieved in practice because the planetary radius R_P is a globally averaged quantity – in reality there are mountain ranges, atmospheres, hazardous magnetic fields etc. close to the planet's surface that make $r_p = R_p$ impossible. For those (and other) reasons, some reasonable and more realistic values for the minimum allowable flyby distances $r_p > R_p$ must be assumed.

To find an expression for the pericenter distance r_p, first note that

$$\cos\left(\frac{\alpha}{2}+\frac{\pi}{2}\right) = -\sin\left(\frac{\alpha}{2}\right) = \frac{-1}{e},$$

as follows from evaluating Equation A.2 at infinite distance. Then, the relation between the pericenter and e and a gives

$$e = 1 - \frac{r_p}{a},$$

Equation A.22 gives

$$V_{\text{peri}}^2 = V_\infty^2 + \frac{2\mu}{r_p},$$

and Equation A.4 gives

$$a = \frac{-1}{\frac{V_{\text{peri}}^2}{\mu} - \frac{2}{r_p}},$$

which can all be combined into

$$\sin\left(\frac{\alpha}{2}\right) = \frac{1}{1+\frac{r_p V_\infty^2}{\mu}} \rightarrow r_p = \frac{\mu}{V_\infty^2}\left(\csc\left(\frac{\alpha}{2}\right)-1\right). \tag{A.32}$$

which nicely expresses the pericenter r_p in terms of α and V_∞. This equation shows that if α is increased, the pericenter distance r_p *decreases*. But increasing α also *increases* the heliocentric ΔV, as shown before[7]. This implies that the minimum *flyable* pericenter distances limit the maximum ΔV that can be obtained from a GAM.

The minimum *flyable* pericenter distances that will be assumed for this mission are listed in Table A.2. The given values are both in [km] from the planet's center of mass, and in altitudes $h = r_p - R_P$, with R_P the corresponding body's (mean) radius. The values listed for the planets are copied from [Melman 2002; table 3.2], the minimum allowable Solar distance[8]

[7] This is also true if the planet is more massive; its value for μ is then larger.

[8] The appearance of the Sun in this table of planets may seem somewhat odd. However, as will be explained in detail in chapter 10, also the Sun will potentially be used as a body for GAM's.

Table A.2 Minimum allowable altitudes during GAM's, both in km and fraction of the body's radius. From [Maleki and Prestage 2001] (Sun) and [Melman 2002] (Planets).

Body	Mean radius R_P [km]	min. r_p [R_P]	min. h [km]
Sun	695,500	4	2,782,000
Mercury	2,440	1.082	200
Venus	6,052	1.047	284
Earth	6,378	1.048	306
Mars	3,397	1.076	257
Jupiter	71,492	1.600	42,895
Saturn	60,268	1.342	20,612
Uranus	25,559	4.190	81,533
Neptune	24,764	1.181	4,482

is from [Maleki and Prestage 2001]. These values are only indicative, since they are simply those values actually used by spacecraft in the past (see [Melman 2002] for details). Thus, they will only serve as the values to be used in first-order calculations, and even there they will be used as mild constraints at most. The body's (mean) radius however, *will* be used as a hard constraint in the penalty functions (see section 11.2.1).

Finally, to get an idea of what a GAM can accomplish, some rough estimates of the possible energy gain by virtue of a GAM are presented. It should be clear that the change in total energy per unit mass $\Delta\epsilon$ due to a GAM is equal to the change in Heliocentric kinetic energy,

$$\begin{aligned}
\Delta\epsilon &= \frac{1}{2}\left(\left(V_H^+\right)^2 - \left(V_H^-\right)^2\right) \\
&= \frac{1}{2}\left(\boldsymbol{V}_H^+ + \boldsymbol{V}_H^-\right)\cdot\left(\boldsymbol{V}_H^+ - \boldsymbol{V}_H^-\right) \\
&= \boldsymbol{V}_p\cdot\left(\boldsymbol{V}_\infty^+ - \boldsymbol{V}_\infty^-\right) \\
&= \frac{2V_pV_\infty\cos(\beta)}{1+\frac{r_pV_\infty^2}{\mu}},
\end{aligned} \tag{A.33}$$

where the last step follows from applying Equation A.32 in Figure A.7(a). When taking $\beta = 0$, $r_p = R_p$, and the different values for μ of all planets in the Solar system, this relation between V_∞ and the possible energy gain can be shown graphically (see Figure A.8). From this figure it is immediately obvious that Jupiter is the most attractive planet to be used for a GAM – it can induce the largest energy gain of all planets *by far*. Other attractive planets are Venus and the Earth, especially since their maximum energy gain is accomplished at relatively low values of V_∞ – this makes these planets ideally suited for an initial "boost" to get to the outer planets.

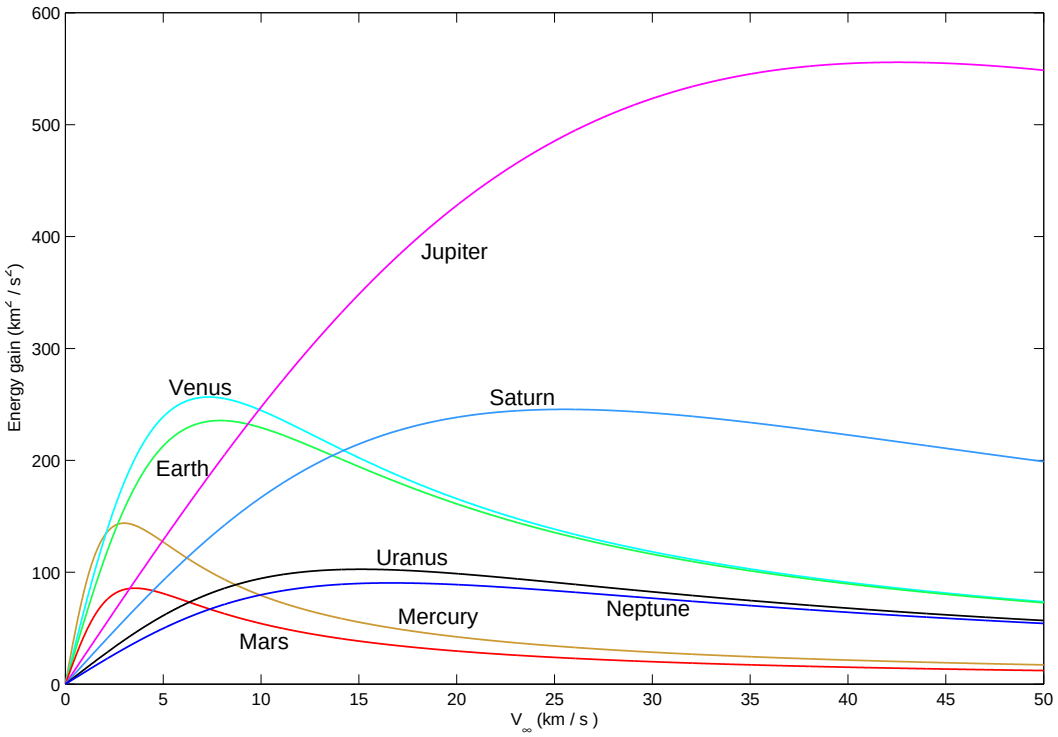


Figure A.8 Maximum possible energy gain for performing GAM's with all planets in the Solar system, as a function of the approach V_∞. In this plot, the pericenter distances were assumed equal to the planet's (mean) radius, and the alignment of the GAM was assumed to be ideal.

B

Basic Algorithms and Preliminary Pruning

B.1. Ephemerides

Although Johannes Kepler (1571-1630) discovered that the planets move along elliptic paths around the Sun, this proved to be only an approximation to the *actual* shape of the orbits. About half a century later, Isaac Newton (1643–1727) proved that elliptic orbits can only be valid in cases where there are only two bodies in gravitational interaction. In case there are more than two bodies (as is the case in the Solar system) the elliptic approximation is a zeroth-order approximation at best. In fact, Urbain Le Verrier (1811–1877) and John Couch Adams (1819–1892), both used the observed deviations of Uranus from its elliptic orbit, to predict the existence of another planet. This planet, which we now call Neptune, was first observed in 1846 by Johann Gottfried Galle (1812–1910), while pointing his telescope at the predicted coordinates. This story nicely shows how far a planet can deviate from its nominal elliptic orbit, especially when taking into account the observational inaccuracies associated with observations from that era.

Determining the positions of the planets *accurately* is no easy task. This becomes even more complicated when trying to determine where a much less massive body will be in the future, such as an asteroid or spacecraft. The only possible way is to take into account all forces acting on all bodies at all instants in time, and then calculating for every instant in time what the resulting motion will be. Needless to say, this is quite tedious and time-consuming work, even more so when the number of bodies to be taken into account is quite large.

Fortunately, many methods have been developed to facilitate this task. Generally, these methods reduce the resulting accuracy slightly, but greatly improve upon computational cost. Several of these methods will be discussed in this section.

B.1.1 First Order (Moderate Accuracy)

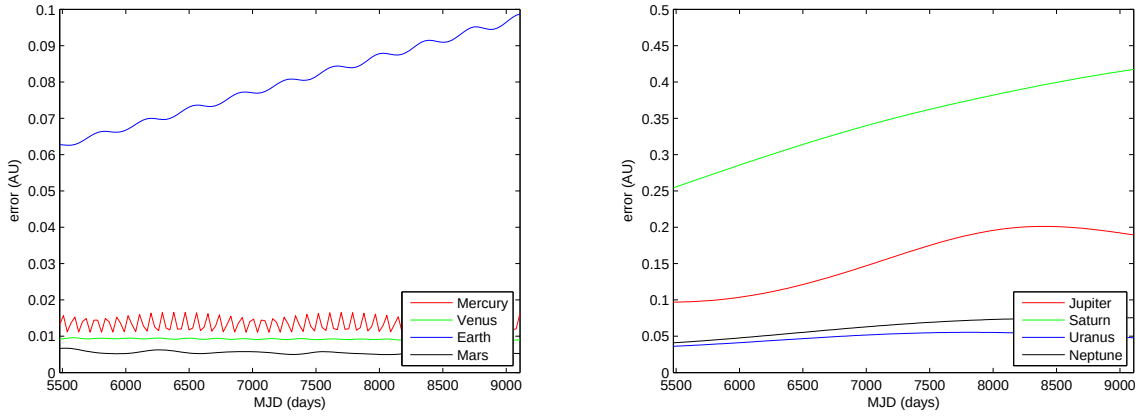
The simplest and fastest way to compute the position of a body at an arbitrary instant in time, is to simply ignore the existence of other bodies aside from the Sun and the body under consideration. Computing positions this way is usually referred to as *solving the two-body problem*.

In the absence of other gravitating bodies, the position may be computed by solving Kepler's equation,

$$M = M_0 + n(t - t_0) = E - e \sin E, \tag{B.1}$$

where $n = \sqrt{a^3/\mu}$ is the orbit's mean motion, M the mean anomaly, e the eccentricity, E the eccentric anomaly, and t_0 and M_0 some initial values at a certain initial epoch. This equation is a valid approximation for all bodies in the Solar system, since all their trajectories are closed, and ellipse-shaped in zeroth approximation. See appendix F for more details on how to solve Kepler's equation efficiently.

To get an idea of the accuracy of this method, the positions of all the planets in the time window [Jan. 1st, 2015, Dec. 31st, 2025] have been computed using Kepler's equation, with the orbital elements from J2000.0. These positions were then compared with very accurate ephemerides from the NASA/JPL HORIZONS System (see [Yeomans and Chamberlin 2009]) in the [Jan. 1st, 2015, Dec. 31st, 2025] time window. The magnitude of the difference vector between these two sets of coordinates is shown in Figure B.1. In these figures, the time step between data points was 30 days.



(a) Inner planets

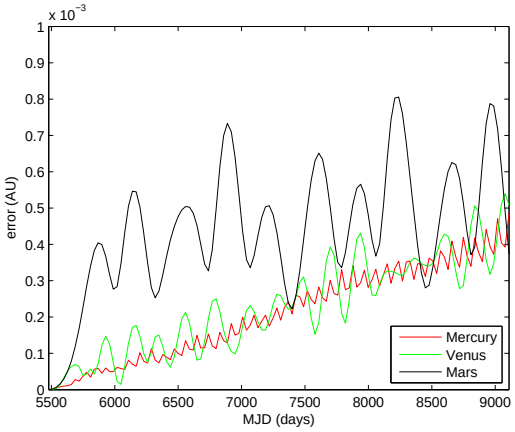
(b) Outer planets

Figure B.1 The magnitude of the difference vector between coordinates obtained from NASA/JPL and Kepler's method. Notice the relatively low accuracy in case of the Earth, due to ignoring the presence of the Moon. These figures were produced using a time step of 30 days, and the orbital elements from J2000.0.

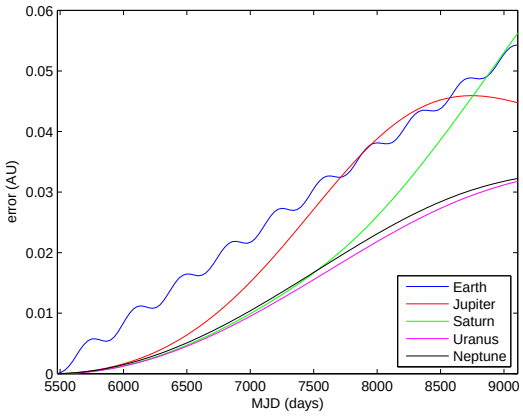
As can readily be concluded from these figures, the accuracy is quite low for some cases. Although the best case (Mars) is only off by ∼0.01 AU or ∼1.5 million km, the error in the

case of Saturn is ∼0.25 AU or ∼37.5 million km, which is unacceptably high. Note that this method does not take into account that the Moon and the Earth are separate bodies (the Earth-Moon *barycenter* was used), which explains the periodic pattern in Earth's position error.

This accuracy can be somewhat improved upon by using the orbital elements not from J2000.0, but as they are at the beginning of the aforementioned [Jan. 1st, 2015, Dec. 31st, 2025] time window. This results in Figure B.2. This indeed improves the accuracy of the positions significantly.



(a) Mercury, Venus, Mars



(b) Earth, Jupiter, Saturn, Uranus and Neptune

Figure B.2 The magnitude of the difference vector between coordinates obtained from NASA/JPL and Kepler's method. These figures were produced again by using a time step of 30 days, but now with the orbital elements as they are in Jan. 1st, 2015. The accuracy is much improved compared to the previous case.

B.1.2 Second Order (High Accuracy)

Planets

Much work on the planetary ephemerides has been done in the past by the planetary science group from NASA/JPL. The accumulated work of many years has resulted in the HORIZONS system [Yeomans and Chamberlin 2009], which can employ any of two elaborate force models (known as DE200 and DE405). Of these two, the DE405 is the most accurate. This model was developed using observational data from optical transit measurements of the Sun and planets since 1911, radar ranging to Mercury and Venus since 1964, tracking of deep space probes, planetary orbiters and landers since 1971, and lunar laser ranging since 1970 [Gill and Montenbrück 2000; chapter 3, page 76].

Both systems are built around taking all these data and performing an extremely accurate simulation of the Solar system backwards and forwards in time (the extended DE405 can give planetary ephemerides from the year 3000 BC to 3000 AD). These simulations take into account the mutual gravitational attraction of all planets and their (known) natural satellites

and several larger asteroids (Ceres, Juno and Vesta). In addition to that, it takes into account shifts in the J2000.0 coordinate system due to the precession of the Earth's nodes and first-order relativistic corrections to the equations of motion. The ephemerides resulting from these simulations are subsequently compressed by calculating the coefficients of Chebyshev-polynomials that most accurately represent the planet's positions. The HORIZONS system simply takes the appropriate coefficients and converts them efficiently to any desired coordinate system, and any desired system of units. As can be found in Yeomans and Chamberlin [2009], the maximum error made by this system lies in the order of 100 m (for the planet Neptune near the end of the year 2160), but is normally well below 10 m.

The representation of ephemerides in Chebyshev polynomials makes this system extremely compact (the actual data are not needed, just the coefficients), fast, very accurate and therefore extremely powerful. The HORIZONS system is the international standard used by scientists and engineers all over the world, to either obtain the coordinates of the planets or validate their own software.

The source code of the HORIZONS system can be downloaded free of charge from [Yeomans and Chamberlin 2009]. Alternatively, the online version offers automated retrieval of ephemerides via Telnet, e-mail, or simply manually via the web interface. Because HORIZONS is written in FORTRAN-77, is is well suited for use by GALOMUSIT and OPTIDUS (see section 4). However, most of this research will be carried out in MATLAB®, for which there are some conversion issues when implementing large amounts of FORTRAN-77 code. It is however indispensable to have data of the same accuracy of the base data in these separate simulations. For that reason, some other means must be used to compute the ephemerides in MATLAB®.

[Meeus 2005; chapter 31] describes a simple method to find the *mean orbital elements* of all the planets as a function of time. Taking the mean elements means giving up on the principal periodic terms of the osculating orbital elements, the most prominent of which is nutation. Basically, this method approximates the orbital elements by a third-order least-squares polynomial to the orbital elements over a long period of time. The equation thus used to predict the instantaneous value of an orbital element, is

$$\alpha = a_0 + a_1 T + a_2 T^2 + a_3 T^3, \tag{B.2}$$

where α is an arbitrary orbital element, a_0 through a_3 are coefficients that depend on α and the planet under consideration, and T is the time measured in Julian Centuries:

$$T = \frac{\text{JDE} - 2451545}{36525},$$

where JDE are Julian ephemeris days measured from J2000.0 (days of 86400 seconds). Note that the coefficient a_1 is *not* the centennial rate of the orbital element, as is often the case in similar methods. Note also that the numerator in fraction above (JDE - 2451545) describes the amount of days passed since *midnight* on the 1st of January, 2000 – this is different from the J2000.0 definition, for that is defined from *noon* on the same date (the associated number in J2000.0 is 2451544.5; half a day *less*).

As an example, the coefficients a_0 through a_3 of Mars and the Earth, as given in [Meeus 2005], are listed in table Table B.1. Note that these coefficients have been scaled such that all resulting angles will be in *degrees*, and all distances in *astronomical units*.

Adopting the notation used in [Meeus 2005]:

$$\begin{array}{rclrcl} L &=& \text{mean longitude,} & \Omega &=& \text{longitude of the ascending node,} \\ a &=& \text{semi-major axis,} & \omega &=& \text{argument of perihelion,} \\ e &=& \text{eccentricity,} & \pi &=& \text{longitude of perihelion,} \\ i &=& \text{inclination to the ecliptic,} & \nu &=& \text{true anomaly,} \end{array}$$

the relations between the orbital elements are then as follows:

$$\begin{aligned} \pi &= \Omega + \omega \\ M &= L - \pi = L - \Omega - \omega \\ \nu &= M + C, \end{aligned}$$

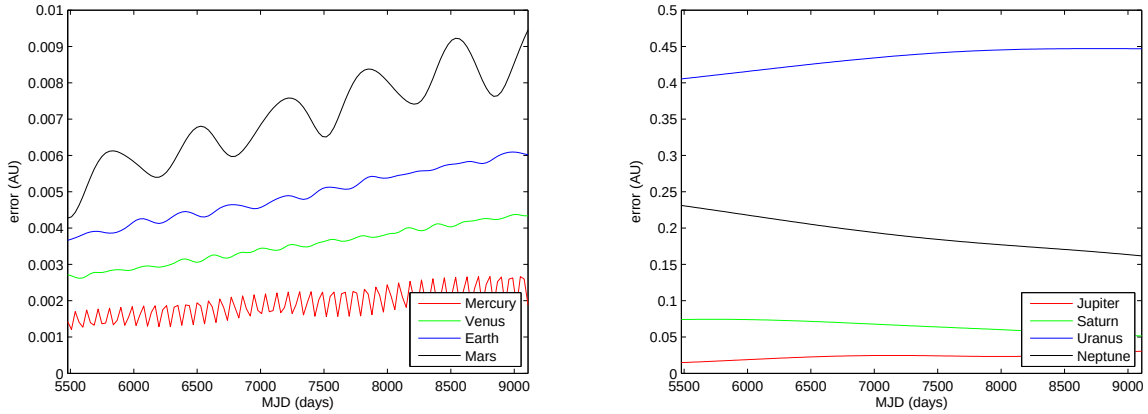
where C is given by the *equation of the center*[1] of the orbit,

$$\begin{aligned} C \;\cong\; & \left(2e - \frac{e^3}{4} + \frac{5}{96}e^5\right)\sin M + \left(\frac{5}{4}e^2 - \frac{11}{24}e^4\right)\sin 2M \\ + \; & \left(\frac{13}{12}e^3 - \frac{43}{64}e^5\right)\sin 3M + \frac{103}{96}e^4 \sin 4M + \frac{1097}{960}e^5 \sin 5M. \end{aligned}$$

which holds only if $e < 0.663$. This condition is surely met for all the planets – the maximum eccentricity is that of Mercury, $e_{\text{Mercury}} \simeq 0.206$. With all of the above, all the mean orbital elements are defined, and the position of the planets can be found from a simple coordinate transformation (see section F).

Again the accuracy has been compared with the high-accuracy ephemerides obtained from the NASA/JPL HORIZONS system. The results are plotted in Figures B.3. Notice that the accuracy is quite disappointing – although the overall accuracy compared to Figure B.1 has improved by an order of magnitude, the *inverse* is true when comparing to Figure B.2. Most importantly, it still implies introducing errors in the order of *millions* of km. In the case of Neptune and Uranus the situation is even worse – the accuracy of their positions is actually *less* than in Figure B.2. It could be argued that this is again due to the poor choice of the initial elements. This method also takes the initial orbital elements (a_0) from J2000.0. However, if these elements are again changed into those elements at the start of Jan. 1$^{\text{st}}$, 2015, the results are very similar to those of Figure B.3. This is to be expected; the cubic fit was done from J2000.0, so the fit would have to be *re-done* when shifting this basis, which, judging from Figure B.3, seems hardly worth the effort.

[1] The equation of the center is basically the Fourier series expansion required to write the true anomaly as a direct function of the mean anomaly ($\theta = F(M)$). See [Wakker 2007; section 6.5] for a more complete treatment on this.



(a) Inner planets

(b) Outer planets

Figure B.3 Difference in the positions resulting from the HORIZONS system and the 1st order method from [Meeus 2005]. Note the overall improvement compared to Kepler's method, except for the cases of Neptune and Uranus – their accuracy is *worse* compared to Kepler's method.

Earth				
	a_0	a_1	a_2	a_3
L	+100.46645700	+36000.769828	+0.0003032200	+0.00000002000
a	+1.0000010180	+0.0000000000	+0.0000000000	+0.00000000000
e	+0.0167086300	-0.0000420370	-0.0000001267	+0.00000000014
i	+0.0000000001	+0.0000000000	+0.0000000000	+0.00000000000
Ω	+0.0000000000	+0.0000000000	+0.0000000000	+0.00000000000
π	+102.93734800	+1.7195366000	+0.0004568800	-0.00000001800

Mars				
	a_0	a_1	a_2	a_3
L	+355.4330000	+19141.6964471	+0.0003105200	+0.00000001600
a	+1.523679342	+0.00000000000	+0.0000000000	+0.00000000000
e	+0.093400650	+0.00009048400	-0.0000000806	-0.00000000025
i	+1.849726000	-0.00060110000	+0.0000127600	-0.00000000700
Ω	+49.55809300	+0.77209590000	+0.0000155700	+0.00000226700
π	+336.0602340	+1.84104490000	+0.0001347700	+0.00000053600

Table B.1 All coefficients to determine the mean orbital elements for Mars and Earth [Meeus 2005].

To resolve this issue, an *ad-hoc* solution was devised. Instead of using the HORIZONS system directly, the *output* of the HORIZONS system for all the bodies in the DE405 model will be used instead. The state vectors of all the planets, as well as the bodies Ceres, Vesta, Pallas, and Pluto, are retrieved from the online HORIZONS web interface, with the settings such that the highest accuracy possible is attained. This means including the corrections for stellar aberration and precession of the nodes, a small interval size and a long time window. The resulting state vectors are then stored in a data file, and read in by MATLAB®. To allow an

integrator to find the positions and velocities of a planet also at times *not* included in this data set, the state vectors are interpolated by *cubic splines* – piecewise continuous and smooth cubic polynomials, which avoid problems like Runge's phenomenon (rapid oscillations near the endpoints of an interval) or data exclusion (only approximating the actual data points).

This is most easily accomplished by the standard MATLAB® function `spline()`, and the corresponding function-value lookup function `ppval()`. File sizes are still very reasonable (∼1 MB) for an interval length of 10 days, if the ephemerides are stored for the interval 2000 – 2150. Unfortunately, the ephemerides of the bodies Mars, Jupiter, Uranus, and Pluto have been calculated by both the DE405 and DE409 models, which resulted in (small) discontinuities around the epoch 01-01-2050. Therefore, the epoch window for each body was kept in the time frame [2000, 2050], which is fortunately sufficiently large for the current mission. The cubic-splines interpolation resulted in splines requiring a surprisingly small amount of memory (2.86 kB) for each of the states in the state vectors.

The accuracy of this *ad-hoc* method is of course (nearly) equal to that of the HORIZONS system, since the cubic splines polynomials go exactly *through* all the data points while preserving continuity in the derivatives. At the very least, the error does not accumulate (as with the other methods), but returns to zero with a period of 10 days. The calculation speed is however quite fast. For comparison, calculating the position and velocity of Jupiter at one million random epochs using the cubic splines required 32 seconds, whereas the same procedure for the two-body problem required no less than 110 seconds. Also, the code found in the `ppval()` function leaves room for further optimization by "de-generalizing" it. This decreased the computation time for one million epochs further down to ∼ 12 seconds. Therefore, when the positions of the planets need to be known, this method will *always* be used instead of the first-order Kepler-method.

Minor Planets

Unfortunately, applying the same method as described in section B.1.2 to MP's is not realistic – retrieving and storing all ephemerides for each of the ∼400.000 asteroids simply requires too much effort and storage. Thus, the only way to find the position of asteroids in second order approximations, is by numerical integration. Note that this argument holds for both the HORIZONS system and the MATLAB®-version – HORIZONS *does* offer asteroid ephemerides, but it determines these also by direct numerical integration.

There is however a simplification that can be made when accurate ephemerides of a rather large amount of MP's are required. Instead of taking into account *all* planets, large moons and large asteroids (as in the HORIZONS DE405 force model), only the body with the largest mass aside from the Sun can be taken into account. This leads to the so-called 3rd-body perturbation technique, which can be stated as (see for example, [Vallado 2001; pages 539–542])

$$\ddot{\boldsymbol{r}}_{\mathrm{MP}} = -\frac{\mu_S \boldsymbol{r}_{\mathrm{MP}}}{r_{\mathrm{MP}}^3} + \mu_3 \left(\frac{\boldsymbol{r}_{\mathrm{MP}_3}}{r_{\mathrm{MP}_3}^3} - \frac{\boldsymbol{r}_{\mathrm{MP}}}{r_{\mathrm{MP}}^3} \right), \tag{B.3}$$

where $\boldsymbol{r}_{\mathrm{MP}_3}$ is the instantaneous distance from the MP to the third body, and $\boldsymbol{r}_{\mathrm{MP}}$ that from the MP to the central body. Taking the 3rd body to be Jupiter (plus the masses of the

four Galilean satellites), the three-body numerical integrator gets the position for Jupiter at each time step from either the HORIZONS system, or the method described in section B.1.2, followed by calculating the instantaneous distance from the MP to Jupiter $\boldsymbol{r}_{\mathrm{MP}}^{\mathrm{Jupiter}}$ and integrating the MP's motion under the combined forces from Jupiter and the Sun by using Equation B.3.

When all the final mission targets have been selected from optimizations that use either of the aforementioned methods, high accuracy ephemerides for the selected MP's will be required to work out the details of the flight program. This implies performing a direct integration of all final mission targets using the complete DE405 force model. Although this is generally quite costly from a computational standpoint, when the amount of final targets has been reduced to only a few this is no longer an issue. A direct integration to determine the positions and velocities at of a large amount of gravitationally interacting bodies a certain epoch is usually referred to as *solving the N-body problem*. See section B.3.3 for a brief discussion on this subject.

B.2. Perturbative Forces

As mentioned before, the two-body model is quite limited when it comes to accuracy. This is due to the fact that the two-body formalism is constructed around the assumption that the central attractive force is the only force present. In reality many more forces are present, and although they are generally quite small compared to the central force, excluding them from the calculations would be unacceptable – they act continuously and therefore accumulate, especially for relatively small bodies such as (interplanetary) spacecraft.

Given their small magnitude, these additional forces are usually called *perturbative forces*. Some of these forces are discussed qualitatively in this section. The main purpose of this section is to show that, in the scope of the current mission, most of these perturbations can safely be left out of consideration.

B.2.1 Sources of Perturbations

The most prominent sources of perturbative forces are [Gill and Montenbrück 2000]:

Gravitational Forces from non-central bodies These forces arise from gravitating bodies other than the central body. Examples are the gravitational forces of the Sun and Moon in case a spacecraft orbits the Earth (the so-called *Luni-Solar* perturbation), or the gravitational attraction of Jupiter and Saturn that have large long-term effects on the orbits of the asteroids in the Asteroid Belt. In interplanetary space, these perturbations are by far the largest cause of deviation from a Keplerian trajectory. These perturbations must therefore be included in the force model to be used in this research if any reasonable degree of accuracy is to be achieved.

Solar Radiation Pressure The force due to Solar Radiation Pressure (SRP) is due to the constant flux of EM-radiation from the Sun (light, x-rays, ...) onto the spacecraft.

This radiation continuously transfers a small amount of momentum to the spacecraft according to the laws of quantum mechanics, effectively de-creasing the local Sun-central gravitational force. Famous examples of how this force influences an object's orbit, are the Poynting-Robertson effect, which causes very small bodies initially in Solar orbit to escape the Solar system altogether, or the Solar sail, which can serve as a propulsion system for spacecraft by itself. The magnitude of this force is hard to quantify since it depends on the surface area and surface properties (and velocity) of the irradiated body, and in case of a spacecraft these quantities can vary a great deal. In general though, this force is several orders of magnitude smaller than the aforementioned non-central gravitational perturbations. Therefore, the perturbations caused by SRP are entirely left out of consideration for this research.

Electromagnetic Forces In case a (small) body has some electric charge on it, it will experience a force when it is moving in the vicinity of a body that has a magnetic field. Alternatively, if some component on the spacecraft requires a relatively large current it will cause a Lorentz force to be exerted on the spacecraft. These effects are especially relevant for orbits around planets such as the Earth or Jupiter, since these have particularly strong magnetic fields. But also the interplanetary space is filled with the Sun's magnetic field, so that this perturbation is potentially always present. This type of perturbation particularly affects spacecraft equipped with ion engines – although measures can be taken to minimize the effects, these engines always leave some residual charge on the spacecraft and require a large current in some places. The magnitude of this force is again hard to quantify, but is usually also several orders of magnitude smaller than the third-body gravitational effects – this force can also be safely ignored for the purpose of this research.

Non-spherical Planets All planets are spheres only by approximation. In reality, most planets and natural satellites have some degree of non-homogeneous density variations in their interior, and shapes that deviate quite much from a perfect sphere. These variations give rise to (small) local perturbations in the central force when a satellite orbits around a given body. Although much more complicated in detail, such perturbations are mostly caused by the so-called J_2-effect, which comes from the planet's oblateness due to its own rotation. When a satellite is in orbit around a planet, this perturbation is usually the largest perturbation present, even larger than the aforementioned third body effects. However, since this mission will most likely not complete even one full orbit around any of the planets (aside from the Earth perhaps, directly after launch), the perturbations caused by the J_2-effects will probably be much smaller than the small errors introduced by even the second-order methods used for the trajectory optimizations. Therefore, the J_2-effects are initially ignored, but will be included if time permits.

Other Perturbations Of course, all of the above perturbations are only the most prominent forces. There are many others, such as atmospheric drag (when orbiting relatively low around a body with an atmosphere), small-body perturbations (the gravitational pull exerted by all of the asteroids, comets, and other Solar system "debris"), relativistic effects (which comes from the fact that the Newtonian theory used in the calculations is actually incorrect), thermal radiation (for example, the infrared radiation coming from a radiator to cool the spacecraft, also gives the spacecraft a small impulse in the

opposite direction), and many others. These perturbations are (usually) much smaller than the forces already mentioned, so none of these forces will be considered in this research.

An impression of the magnitudes of various perturbations is given in Figure B.4. Although this figure is more suited for satellites in Earth orbit, it does serve as a justification for the conclusions drawn above. The figure shows that the influence of the Sun, Moon, Jupiter, Venus (and by extension, all other large bodies in the Solar system) gradually increases with increasing distance from the Earth – in interplanetary space, all these forces become vastly greater than all the other perturbations. Naturally, far from the Earth, all J-components, dynamic solid tide etcetera also become vanishingly small compared to the GM-forces from other Solar system bodies.

B.3. Orbit Propagation

If the position and velocity of a body at some epoch t are known, one can predict quite accurately where that body is going to be at some later epoch $\hat{t} = t + \Delta t$. Doing such predictions is generally called *orbit propagation*, which is the subject of this section. As always, there is a trade-off between accuracy of the solution and computational speed. The first two methods described here revolve around computational speed, and the last method has accuracy as its primary concern.

B.3.1 Repeated Coordinate Transformations

The classical (and easiest) way to propagate bodies along their trajectories, is by repeatedly applying the correct coordinate transformations. In cases where the initial state is given in terms of the classical Keplerian elements, and the elements at some later epoch are required, the only required operation is solving the corresponding Kepler's equation. Such cases are quite straightforward to solve, and not much can be done to improve computational efficiency with such propagations.

If however the initial state at a given time t is given in terms of the Cartesian state vector $\boldsymbol{\chi}_1$, and a state vector $\boldsymbol{\chi}_2$ at a later time $\hat{t} = t + \Delta t$ is needed, the following transformations must be applied:

1. Convert the given state vector $\boldsymbol{\chi}_1$ to the corresponding Keplerian elements. This can be accomplished with the method presented in appendix F.3.

2. With the Keplerian elements, the mean anomaly at the later epoch is simply $M_2 = M_1 + n_1 \Delta t$, where M_1 is the mean anomaly at time t found with the previous coordinate transformation, and $n_1 = \sqrt{a_1^3/\mu}$ is the orbit's mean motion.

3. Solve the corresponding Kepler's equation to find E_2 and thus θ_2 at time $\hat{t}$. This is accomplished with the methods presented in appendix F.1.

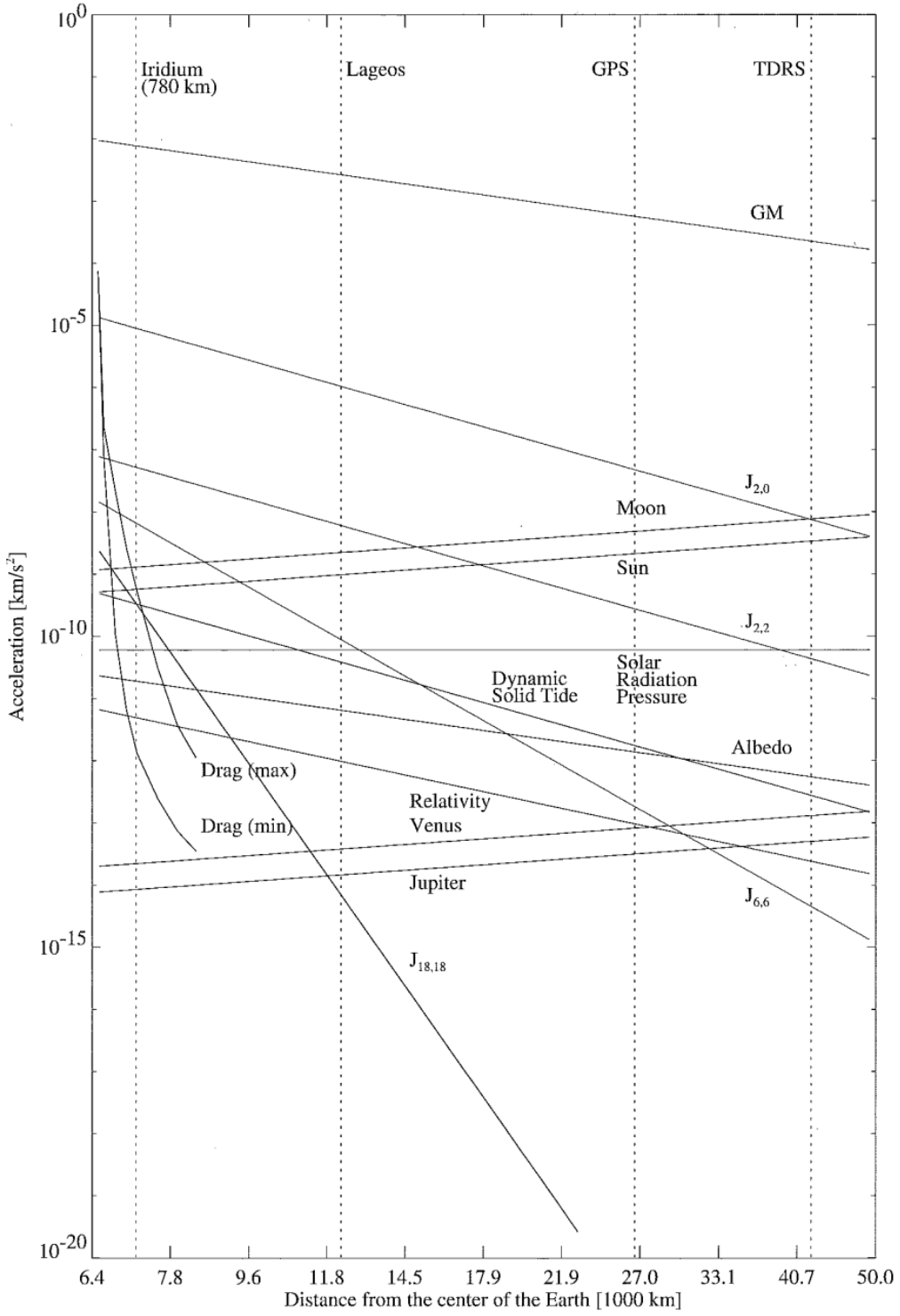


Figure B.4 Various perturbations for a satellite in Earth orbit, as a function of the satellite's altitude. Note that J_2 is the largest perturbation compared to the central GM-force, but still 3 to 4 orders of magnitude smaller. This is [Gill and Montenbrück 2000; figure 3.1].

4. Convert the new Keplerian elements back to Cartesian coordinates, which yields the required state vector $\boldsymbol{\chi}_2$. This is accomplished with the method described in appendix F, section F.2.

5. Alternatively, in problems where computational speed can really form a bottleneck, the conversion from E_2 to θ_2 can be skipped altogether by solving $\boldsymbol{\chi}_2$ with the parametric representation from section A.1.5. See appendix F for more details on this.

Although quite intuitive and straightforward, the number of operations to accomplish mentioned coordinate transformations is quite large. In situations where an orbit has to be propagated very often (for example, when using the method of patched micro conics, section 13.2), applying these transformations for every step is quite tedious and unnecessarily slow.

B.3.2 State Transition Matrix

Fortunately, there is a faster way to propagate a point mass along its trajectory. Shepperd [1984] found a completely general method to produce the STM Φ without the need for such triple coordinate conversions. The STM $\Phi = \Phi(\delta t)$ is the matrix such that

$$\boldsymbol{\chi}_2 = \Phi(\delta t) \cdot \boldsymbol{\chi}_1, \tag{B.4}$$

where $\boldsymbol{\chi}_1$ is some initial state vector, $\boldsymbol{\chi}_2$ the state vector an amount of time δt before or after the initial state vector $\boldsymbol{\chi}_1$. In this formulation, Kepler's equation can be solved by means of a *continued fraction*. Generally, continued fractions have very fast convergence rates, and this one is no exception.

First, initialize the following parameters:

$$r_0 = |\boldsymbol{x}_0| \tag{B.5}$$
$$\nu_0 = \boldsymbol{x}_0 \cdot \dot{\boldsymbol{x}}_0 \tag{B.6}$$
$$\beta = \frac{2\mu}{r_0} - (\dot{\boldsymbol{x}}_0 \cdot \dot{\boldsymbol{x}}_0) \tag{B.7}$$

The eccentricity of the orbit is captured in the quantity β. If $\beta < 0$, the orbit is elliptical, in case $\beta > 0$, it is hyperbolic. It is not known beforehand whether the given time step t would imply a complete revolution in case the orbit is elliptical. Therefore, when $\beta < 0$, an additional quantity ΔU must be initialized to take into account the effects of (possible) multiple revolutions:

$$P = \frac{2\pi\mu}{\beta^{3/2}} \tag{B.8}$$
$$n = \text{largest integer } \leq \frac{1}{P}\left(t + \frac{P}{2} - \frac{2\nu_0}{\beta}\right) \tag{B.9}$$
$$\Delta U = \frac{2n\pi}{\beta^{5/2}}. \tag{B.10}$$

The desired state vector $\boldsymbol{\chi}_2 = [\boldsymbol{x}_2,\, \dot{\boldsymbol{x}}_2]^T$ at time $t+\delta t$ can be written in terms of $\boldsymbol{\chi}_1 = [\boldsymbol{x}_1,\, \dot{\boldsymbol{x}}_1]^T$ as

$$\begin{aligned}
x_2 &= r_0 U_0 + \nu_0 U_1 + \mu U_2 && \text{(B.11)}\\
\hat{t} &= r_0 U_1 + \nu_0 U_2 + \mu U_3 && \text{(B.12)}\\
\boldsymbol{x}_2 &= f\boldsymbol{x}_0 + g\dot{\boldsymbol{x}}_0 \\
&= \left(1 - \frac{\mu U_2}{r_0}\right)\boldsymbol{x}_0 + (r_0 U_1 + \nu_0 U_2)\dot{\boldsymbol{x}}_0 && \text{(B.13)}\\
\dot{\boldsymbol{x}}_2 &= F\boldsymbol{x}_0 + G\dot{\boldsymbol{x}}_0 \\
&= -\frac{\mu U_1}{x_2 r_0}\boldsymbol{x}_0 + \left(1 - \frac{\mu U_2}{x_2}\right)\dot{\boldsymbol{x}}_0, && \text{(B.14)}
\end{aligned}$$

with $U_n = U_n(w,\, \beta)$ the n^{th}-order Stumpff function,

$$U_n(w,\, \beta) = \sum_{k=0}^{\infty} \frac{(-\beta)^k w^{n+2k}}{(n+2k)!},$$

and w comes from the Sundman transformation,

$$\frac{dt}{dw} = r.$$

It is known that the Stumpff functions have the derivative relationships

$$\begin{aligned}
\frac{\partial U_0}{\partial w} &= -\beta U_1, \\
\frac{\partial U_n}{\partial w} &= U_{n-1} \quad (n > 0), \\
\frac{\partial U_n}{\partial \beta} &= \frac{1}{2}\left(n U_{n+2} + w U_{n+1}\right).
\end{aligned}$$

Shepperd [1984] employs these relationships by introducing the independent variables

$$\begin{aligned}
u &= \frac{U_1(w/4,\, \beta)}{U_0(w/4,\, \beta)} && \text{(B.15)}\\
q &= \frac{\beta u^2}{1 + \beta u^2}, && \text{(B.16)}
\end{aligned}$$

for which it may be derived that

$$U_0(w/2,\,\beta) = 1-2q, \tag{B.17}$$
$$U_1(w/2,\,\beta) = 2u(1-q), \tag{B.18}$$

$$U_0(w,\,\beta) = 2U_0^2(w/2,\,\beta)-1, \tag{B.19}$$
$$U_1(w,\,\beta) = 2U_0(w/2,\,\beta)U_1(w/2,\,\beta), \tag{B.20}$$
$$U_2(w,\,\beta) = 2U_1^2(w/2,\,\beta), \tag{B.21}$$
$$U_3(w,\,\beta) = \beta U(w,\,\beta)+\frac{1}{3}U_1(w,\,\beta)U_2(w,\,\beta), \tag{B.22}$$
$$U(w,\,\beta) = \frac{16}{15}U_1^5(w/2,\,\beta)G_5(q)+\Delta U. \tag{B.23}$$

Taking $u_0 = 0$ and $t_0 = 0$ as initial values, the relations above can be shaped into an iterative scheme that brings the current time t closer to the desired time $\hat{t} = t+\delta t$. The update to r, t and u is given by

$$r \leftarrow r_0U_0(w,\,\beta)+\nu_0U_1(w,\,\beta)+\mu U_2(w,\,\beta) \tag{B.24}$$
$$t \leftarrow r_0U_1(w,\,\beta)+\nu_0U_2(w,\,\beta)+\mu U_3(w,\,\beta) \tag{B.25}$$
$$u \leftarrow u-\frac{t-\delta t}{4r(1-q)} \tag{B.26}$$

where the last step is a Newton-Raphson update to the assumed value for u. In this formulation, it holds that $q \leq 1/2$. This fact can be used to check the correctness of any implementation of the STM-procedure. The solution to Kepler's equation corresponding to this problem is embedded in the quantity $G_5(q) = F(5,\,1,\,7/2,\,q) = G(5,\,0,\,5/2,\,q)$, with $F = F(a,\,b,\,c,\,x)$ the general hypergeometric differential equation (see [Battin 1999; chapter 1]). By using Gautschi's method, the solutions to such equations may be written as

$$G(a,\,b,\,c,\,x) = \frac{F(a,\,b+1,\,c+1,\,x)}{F(a,\,b,\,c,\,x)} = \frac{1}{1-\frac{h_1x}{1-\frac{h_2x}{1-\cdots}}} \tag{B.27}$$

in which

$$h_{2n} = \frac{(n+b)(n+c-a)}{(2n+c-1)(2n+c)},$$
$$h_{2n+1} = \frac{(n+a)(n+c-b)}{(2n+c)(2n+c+1)}.$$

[Shepperd 1984] increases the efficiency of evaluating this continued fraction, by reworking

the modified-Lentz' formulation of the fraction into the following procedure. First, initialize

$$\begin{array}{llll} k & = -9 & A & = 0 \\ \ell & = 3 & B & = 0 \\ d & = 15 & G & = 1 \\ n & = 0. & & \end{array} \tag{B.28}$$

Then, repeat the following calculations until G converges:

$$\begin{aligned} k &\leftarrow -k \\ \ell &\leftarrow \ell + 2 \\ d &\leftarrow d + 4\ell \\ n &\leftarrow n + (1+k)\ell \\ & \\ A &\leftarrow \frac{d}{d - nAq} \\ B &\leftarrow (A-1)B \\ G &\leftarrow G + B. \end{aligned} \tag{B.29}$$

Note that the quantities k, ℓ, d and n all remain integers throughout the calculation, which helps increase the efficiency of the evaluation of the continued fraction. With this, all required operations are explicitly defined. In summary, the complete procedure is given by

1. Initialize all variables (Equations B.5 through B.7 and Equation B.28). Also set

$$u = 0, \tag{B.30}$$
$$\Delta U = 0, \tag{B.31}$$
$$t = 0. \tag{B.32}$$

2. Correct for possible multi-revolution cases by adjusting ΔU, depending on the value of β (Equation B.7)

3. Iterate Equations B.17 through B.26 until the current time t is equal to the given time δt within some tolerance.

4. In every iteration, the quantity $G_5(q)$ must be computed by the continued fraction evaluation (Equation B.29).

5. Once convergence has been achieved ($t = \hat{t}$), the values of the f, F, g and G functions are

$$\begin{array}{llll} f & = 1 - \frac{\mu U_2}{r_0} & F & = -\frac{\mu U_1}{r r_0} \\ g & = r_0 U_1 + \nu_0 U_2 & G & = 1 - \frac{\mu U_2}{r} \end{array}$$

6. The new state vector $\boldsymbol{\chi}_2$ at time t relative to $\boldsymbol{\chi}_1$ is given by Equations B.13 and B.14. (The actual matrix $\Phi(t)$ is rather lengthy (6×6), so it is omitted here for brevity.)

B.3.3 High Accuracy Propagation

Note that the previous two methods to propagate a trajectory only employ a two-body model, that is, they do not take into account the various perturbative forces. Although these forces are quite small compared to the central attractive force (see section B.2.1), in the long run they cause unacceptably large deviations from the two-body model.

Generally, during the last steps of the mission design process, the spacecraft's trajectory is propagated by *direct integration*. Such orbit integration takes into account more forces than just the radial central one (possibly *many* more). This method attempts to solve the nonlinear equations [Chobotov 2002]

$$\frac{d^2\boldsymbol{r}}{dt^2} + \mu\frac{\boldsymbol{r}}{r^3} = \boldsymbol{a}_p, \tag{B.33}$$

where $\boldsymbol{a}_p$ is the sum of all perturbative accelerations at position $\boldsymbol{r}$:

$$\boldsymbol{a}_p = \boldsymbol{a}_{\text{LuniSolar}} + \boldsymbol{a}_{\text{Atmospheric}} + \boldsymbol{a}_{J_2} + \boldsymbol{a}_{\text{Jupiter}} + \dots.$$

Naturally, the more perturbative forces are included in this term, the more accurately any obtained solutions will represent the spacecraft's future states. Of course, many of these perturbative forces do not only depend on the spacecraft's position, but also for example, its orientation (solar radiation pressure, thermal radiation, ...). Thus, to know each term in Equation B.33 usually requires a separate sub-calculation during the integration process. If one order of magnitude of improvement in accuracy is required, it frequently takes several orders of magnitude longer computation times.

One very important factor in both the accuracy and computational efficiency is the integration method used for the propagation. There is a myriad of such methods, each with their own virtues and pitfalls. Equations of the type Equation B.33 can be solved with the methods discussed in appendix E.2.2.

As stated before, for this thesis it suffices to only include the same force model as used by DE405 and neglect all the other perturbative forces, partly because they are too small to have a significant impact, and partly because they can not be determined or even estimated at this stage. Therefore, the focus will be on solving the N-body problem.

Third Body Perturbations

Mathematically, the acceleration of a body i in an N-body system can be written as

$$\ddot{\boldsymbol{r}}_i = -\sum_{j=1}^{N} \frac{\mu_j \boldsymbol{r}_{ij}}{r_{ij}^3}, \quad j \neq i \tag{B.34}$$

where N is the total number of bodies in the system, μ_j is the standard gravitational parameter of body j, $\boldsymbol{r}_{ij}$ is the vector spanned from body i to body j, and $\ddot{\boldsymbol{r}}_i$ is the resulting acceleration of body i.

The terms in this equation can be rearranged to separate the effects from the central body, the main perturbing body (the *third body*), and all the further removed bodies (the *perturbing bodies*). Simply separating the central body term and the third body term from the rest and rearranging gives ([Bond and Allman 1996; chapter 11] or [Wakker 2007])

$$\ddot{\boldsymbol{r}} = -G\frac{m_1 + m_2}{r^3} - Gm_3\left(\frac{\boldsymbol{d}_3}{d_3^3} - \frac{\boldsymbol{\rho}_3}{\rho_3^3}\right) - G\sum_{j=4}^{N} m_j\left(\frac{\boldsymbol{d}_j}{d_j^3} - \frac{\boldsymbol{\rho}_j}{\rho_j^3}\right). \tag{B.35}$$

This reformulation emphasizes the effects of the perturbing bodies relative to those of the central body more clearly. Here, m_j is the mass of body j, $\boldsymbol{d}_j$ is the vector from the perturbing body j to the spacecraft (and d_j its magnitude), and $\boldsymbol{\rho}_j$ is the vector from the central body to the perturbing body j (and ρ_j its magnitude), and $\boldsymbol{r}$ is the vector from the central body to the spacecraft (and r its magnitude). See also Figure B.5.

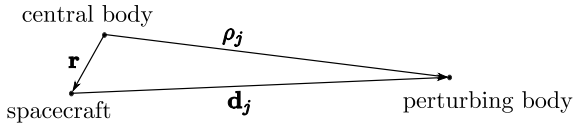


Figure B.5 Geometry that facilitates notation in the reformulation of the N-body equations when including third-body effects. Adopted from [Bond and Allman 1996; figure 11.2].

Battin's Reformulation

Equation B.34 can be shortened to [Bond and Allman 1996; chapter 11]

$$\ddot{\boldsymbol{r}} + \frac{\mu}{r^3} = -G\sum_{j=3}^{N} m_j\left(\frac{\boldsymbol{d}_j}{d_j^3} - \frac{\boldsymbol{\rho}_j}{\rho_j^3}\right) \tag{B.36}$$

Although this notation is quite elegant, there is a problem with this equation that gives some numerical problems in practical applications. In cases where $r \ll \rho_j$ and thus $\boldsymbol{d}_j \approx \boldsymbol{\rho}_j$ (the *interior problem*), the term

$$\frac{\boldsymbol{d}_j}{d_j^3} - \frac{\boldsymbol{\rho}_j}{\rho_j^3}$$

is essentially a subtraction of two large and nearly equal terms. This can give rise to relatively large numerical inaccuracies during the computation of this term. Battin [1999] proved that this expression can be rewritten as

$$\ddot{\boldsymbol{r}} + \frac{\mu}{r^3}\boldsymbol{r} = -G\sum_{j=3}^{N}\frac{m_j}{d_j^3}\left(\boldsymbol{r} + f(q_j)\boldsymbol{\rho}_j\right), \tag{B.37}$$

where

$$\begin{aligned} f(q_j) &= q_j\left(\frac{3 + 3q_j + q_j^2}{1 + (1+q_j)^{3/2}}\right), \\ q_j &= \frac{\boldsymbol{r}}{\rho_j^2}\cdot(\boldsymbol{r} - 2\rho_j). \end{aligned}$$

This reformulation completely does away with the numerical inaccuracies introduced by using Equation B.36. It can be proved that this reformulation also simplifies the *exterior problem* – $r \gg \rho_3$, and thus $\boldsymbol{d}_3 \approx \boldsymbol{r}$.

Encke's Method

Finally, a method to simplify the integration required to solve Equation B.34 is presented. This method, generally known as *Encke's Method*, can be formulated as follows.

Let the *true* position of the spacecraft in its orbit be given by $\boldsymbol{r}$, and $\boldsymbol{\rho}$ the spacecraft's position in the orbit given by the corresponding two-body approximation (the *reference orbit*). If the difference between the two is written as

$$\delta\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{\rho},$$

the following equation for the difference in the accelerations between the two orbits can be set up [Madonna 1997; section 7.3]:

$$\delta\ddot{\boldsymbol{r}} = \boldsymbol{a}_p + \frac{\mu}{\rho^3}\left[\left(1 - \frac{\rho^3}{r^3}\right)\boldsymbol{r} - \delta\boldsymbol{r}\right]. \tag{B.38}$$

Where the perturbing acceleration $\boldsymbol{a}_p$ can be found from Equation B.37. Introducing the notation

$$2q = 1 - \frac{r^3}{\rho^3},$$

this can be rewritten in the simpler form

$$\delta\ddot{\boldsymbol{r}} = \boldsymbol{a}_p + \frac{\mu}{\rho^3}\left[\left(1 - (1-2q)^{-3/2}\right)\boldsymbol{r} - \delta\boldsymbol{r}\right]. \tag{B.39}$$

Before this expression can be integrated to yield the difference between the true and reference orbits, the value of q must be known. Writing

$$r^2 = (\rho_x + \delta x)^2 + (\rho_y + \delta y)^2 + (\rho_z + \delta z)^2$$

for the true radial distance, q can be written as

$$q = -\frac{1}{\rho^2}\left[\delta x\left(\rho_x + \frac{1}{2}\delta x\right) + \delta y\left(\rho_y + \frac{1}{2}\delta y\right) + \delta z\left(\rho_z + \frac{1}{2}\delta z\right)\right] \tag{B.40}$$

so that the value of q can be computed at any point where $\boldsymbol{\rho}$ and $\delta\boldsymbol{r}$ are known.

Note that the term $1 - (1-2q)^{-3/2}$ is still the difference between two nearly equal terms, which might cause numerical inaccuracies during the integration process. To remedy this problem, the term can very well be approximated using the corresponding Taylor expansion:

$$1 - (1-2q)^{-3/2} = -3q - \frac{15}{2}q^2 - \frac{35}{2}q^3 - \ldots = -\sum_{n=1}^{\infty}\frac{\prod_{m=1}^{n} 2n+1}{n!}q^n$$

which converges quite quickly for small values of q.

The advantage of this method over a direct integration of the original equations of motion (e.g., Cowell's method) is that substantially larger step sizes can be used while guaranteeing the same level of accuracy, thus reducing the overall computation time. It has some disadvantages though, most notably the complexity of the implementation compared to Cowell's method. Also, when $\delta\boldsymbol{r}$ becomes too large, the reference orbit needs to be updated, which adds to the complexity of the implementation and gives an additional computational burden not experienced with a direct integration.

However, these are small prices to pay if the number of integrations that need to be carried out become quite large, as is usually the case in the last stages of trajectory optimization. In these cases, the reduction in computation time is well worth the effort of implementing it, and therefore this method is used in conjunction with Battin's reformulation given in Equation B.37.

B.4. Pruning the MPCORB Data Set

The number of minor planets in the database is so large that considering all of them in optimizations is simply too costly. But it does not make much sense to take into account MP's that are always beyond the reach of the spacecraft due of the assumed mission parameters, or MP's that remain out of reach as a consequence of the geometry of the spacecraft's instantaneous trajectory. Therefore, an important first step for all optimizations is *pruning* – removing those bodies from the data set that can safely be left out of consideration, without (significantly) altering the end result.

Accuracy of the Orbit Solution

Normally, when a new MP is discovered, the discovery must be verified by other observers before it is included in the MPCORB-database. Given the current rate of discovery, corroboration of this kind generally take weeks if not months. Also, to check whether the new body is actually *unknown*, the positions of all the other bodies must be known at any instant. For these reasons it is imperative to be able to determine *accurately* where a body will be in the sky at any given instant.

This is even more true when such a body is to be visited by a spacecraft. Angular accuracies which are sufficient to find the body in a telescope on Earth several years later, still have a spatial deviation of several thousand kilometers. A mission that is to visit a body must know *precisely* where that body is at any given time. Therefore, the accuracy of the orbit solution for such MP's must be very high.

The MPCORB-database includes a rough measure for the accuracy of the body's orbit solution; the total number of observations made of that body, and the time span these observations were made in. Generally speaking, the more observations, the better the accuracy; for the current mission it is demanded that an MP is observed at least 200 times. An important

problem is *dilution of precision*. If many measurements are taken relatively quickly after one another, they can not have included the time-varying components very accurately, so that inserting these measurements into the data set will only *reduce* the overall precision. This effect is especially relevant for far removed objects, which move only a few seconds of arc per decade as seen by an observer on Earth. Therefore, all observations are demanded to also have a good *spread* of the observations. For the current mission it is demanded that all MP's obey the rule

$$\frac{150 - 10 \cdot r_p}{t_{\text{last}} - t_{\text{first}}} \geq \frac{1 \text{ scaled observation}}{25 \text{ days}}, \tag{B.41}$$

were r_p is the MP's perihelion distance (in AU), and t_{first} and t_{last} (in days) the epochs the MP was first and last observed, respectively. Note that this method is rendered completely ineffective for all MP's for which $r_p > 15$ AU.

The full, unpruned data set is shown in red in Figure B.7. The pruning method outlined above successfully eliminated 302,400 MP's from the original set, so that the pruned set consists of 96,599 MP's. The MP's that remain after applying this pruning method are shown in green in Figure B.7.

Scope Of The Mission

In addition, the position of the SBS allows a large number of MP's to be excluded from consideration. Building on the assumption that the spacecraft will not perform a GAM that spans a large angle on the ecliptic at any planet beyond Jupiter (see section 3.4.2), it makes sense to exclude all minor planets that lie outside the prism-shaped region shown in Figure B.6. This region is defined as a part of the volume of a cone, which has its tip at the Sun and opening angle ϕ outside the orbit of Jupiter. The opening angle ϕ is a free parameter, which will be kept quite wide ($\phi = \pi/4$) to allow a broad search and prevent over-pruning the set.

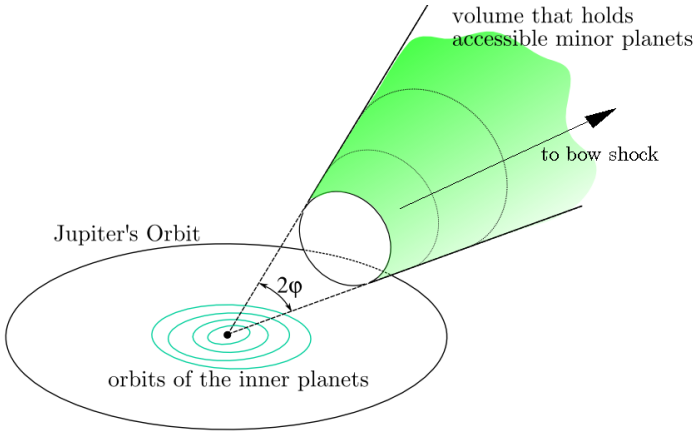


Figure B.6 Volume that holds all far-removed MP's that will be included in the data-set. The volume is defined as part of a cone with its tip at the Sun and opening angle ϕ, outside Jupiter's orbit.

To determine whether an MP will be inside this region during the mission lifetime, the

corresponding two-body problem will be solved at regular intervals in the mission window, and determine whether it travels inside the region at least once. Since any body which moves in an orbit that leads it far from the Sun has very low mean motion near this region of interest, it is sufficient to set this interval equal to one month (4 weeks). This pruning method eliminated an additional 442 bodies from the data set, so that the final set consists of 96,157 MP's[2]. The final set of MP's are colored blue in Figure B.7.

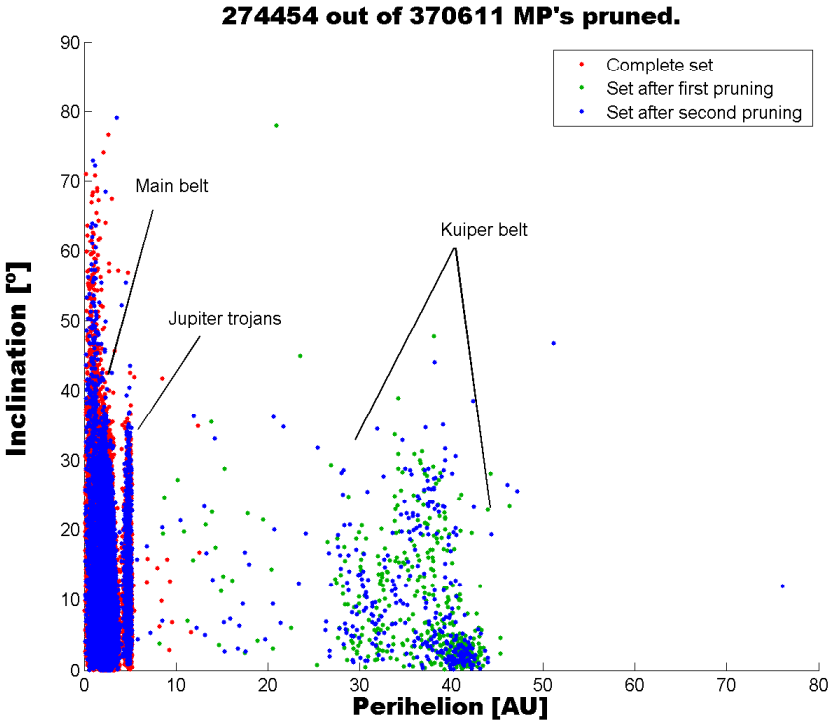


Figure B.7 Plot of the MP data set during all pruning stages. This plot shows the MP's semi-major axes versus inclination, which is done to show that an MP's inclination by itself is not suited to be used as a pruning criterion.The first pruning method was the most effective, resulting in the removal of 302,400 MP's. The second method was less successful; it only managed to reduce the set by 442 MP's.

[2]Note that the combination of these two pruning methods removed a satisfying amount of MP's from further consideration, while keeping the overall structure of the various orbital groupings intact. This was indeed aimed for during the design of these pruning methods; most other methods attempted resulted either in the complete removal of all bodies in the outer Solar system, or in severe under-pruning of the set.

C

Lambert Targeting Routines

C.1. High thrust: Lancaster & Blanchard's Algorithm

Since the equations governing the high-thrust orbital boundary value problem are transcendental, the unknown semi-major axis must be found iteratively. A large number of different iterative methods have been developed over time, each with their own advantages and disadvantages. Quite an efficient one is the method developed by Lancaster and Blanchard [1969], which reduces the problem to finding the root of the equation

$$T(x) = \frac{2(x - qz - d/y)}{E} - \hat{T}, \tag{C.1}$$

where

$$\begin{aligned}
x^2 &= 1 - \frac{s}{2a} & \qquad f &= y(z - qx) \\
q &= \frac{\sqrt{r_1 r_2}}{s} \cos\left(\frac{\Delta\theta}{2}\right) & g &= xz - qE \\
z &= \sqrt{1 + Eq^2} & y &= \sqrt{|E|} \\
d &= \begin{cases} \arctan\left(\frac{f}{g}\right) + m\pi & \text{if} \quad E < 0 \\ \ln(f + g) & \text{if} \quad E > 0 \end{cases} & E &= x^2 - 1 \\
& & \hat{T} &= t_f \cdot \sqrt{\frac{8\mu}{s^3}}
\end{aligned}$$

The derivative of Equation C.1 is [Lancaster and Blanchard 1969]:

$$\frac{dT}{dx} = \frac{4 - 4q^3 x/z - 3xT}{E}, \tag{C.2}$$

if x is *not* near 0 or 1. [1]In the above formulation, solutions for which $-1 < x < 1$ are ellipses,

[1]In cases where this can not be reasonably justified, see [Lancaster and Blanchard 1969] or [Gooding 1990]

$x > 1$ are hyperbolic, and $x = 1$ parabolic. The function T is not defined for $x < -1$, and care should be taken that this constraint on x is *always* met by the employed root-finding algorithm.

The end-result of the Lambert procedure above, are the terminal velocities $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ at the points $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, respectively. Expressions for these are also given by Lancaster and Blanchard [1969]:

$$\begin{array}{rclrcl} \boldsymbol{V}_{r_1} &=& +\gamma\frac{(qz-x)(1-\rho)}{r_1}\hat{\boldsymbol{r}}_1 & \boldsymbol{V}_{r_2} &=& -\gamma\frac{(qz-x)(1-\rho)}{r_1}\hat{\boldsymbol{r}}_2 \\ \boldsymbol{V}_{\theta_1} &=& \sigma\gamma\frac{z+qx}{r_1}\hat{\boldsymbol{\theta}}_1 & \boldsymbol{V}_{\theta_2} &=& \sigma\gamma\frac{z+qx}{r_2}\hat{\boldsymbol{\theta}}_2 \\ \boldsymbol{V}_1 &=& \boldsymbol{V}_{\theta_1} + \boldsymbol{V}_{r_1} & \boldsymbol{V}_2 &=& \boldsymbol{V}_{\theta_2} + \boldsymbol{V}_{r_2} \end{array} \tag{C.3}$$

with q and $z = z(x)$ as before, $\hat{\boldsymbol{\theta}}_{1,2}$ and $\hat{\boldsymbol{r}}_{1,2}$ unit vectors in the perpendicular- and radial directions for the position vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, respectively, and

$$\begin{aligned} \gamma &= \frac{s\mu_S}{2} \\ \sigma &= 2\sqrt{\frac{\boldsymbol{r}_1\boldsymbol{r}_2}{c^2}}\sin\left(\frac{\Delta\theta}{2}\right) \\ \rho &= \frac{\boldsymbol{r}_1 - \boldsymbol{r}_2}{c}. \end{aligned}$$

C.1.1 Initial Estimates

First estimates for x to find a root of Equation C.1, are provided by Gooding [1990]. The complete solution space is shown in Figure C.1. Observe that in case $m = 0$ there is *always* a solution, provided the initial values are such that $-1 < q < 1$. In such cases, the initial estimate x_0 is given by

$$\begin{aligned} \text{if} \quad T_0 - \hat{T} > 0 \rightarrow x_0 &= T_0\frac{T_0 - \hat{T}}{4\hat{T}} \\ \text{if} \quad T_0 - \hat{T} < 0 : x_{01} &= -\frac{\hat{T} - T_0}{\hat{T} - T_0 + 4} \\ x_{02} &= -\sqrt{\frac{\hat{T} - T_0}{\hat{T} + \frac{1}{2}T_0}} \\ \phi &= 2\texttt{atan2}\left(2q,\, 1 - q^2\right) \\ W &= x_{01} + 1.7\sqrt{2 - \frac{\phi}{\pi}} \\ x_{03} &= \begin{cases} x_{01} & \text{if} \quad W \geq 0 \\ x_{01} + (-W)^{1/16}\left(x_{02} - x_{01}\right) & \text{if} \quad W < 0 \end{cases} \\ \lambda &= 1 + 0.5x_{03}(1 + x_{01}) - 0.03x_{03}^2\sqrt{1 + x_{01}} \\ \rightarrow x_0 &= \lambda x_{03}, \end{aligned} \tag{C.4}$$

where $T_0 = T(0)$, and $\texttt{atan2}(x,\, y)$ is the four-quadrant arctangent function, available in most programming environments. These x_0 are based on approximating the $T(x)$-curves with a bilinear approximation, and a bit of "educated trial-and-error" – the more complicated x_0 (the one where $T_0 - \hat{T} < 0$), is basically the result from taking various approximations to $T(x)$, with some additional empirical "patching".

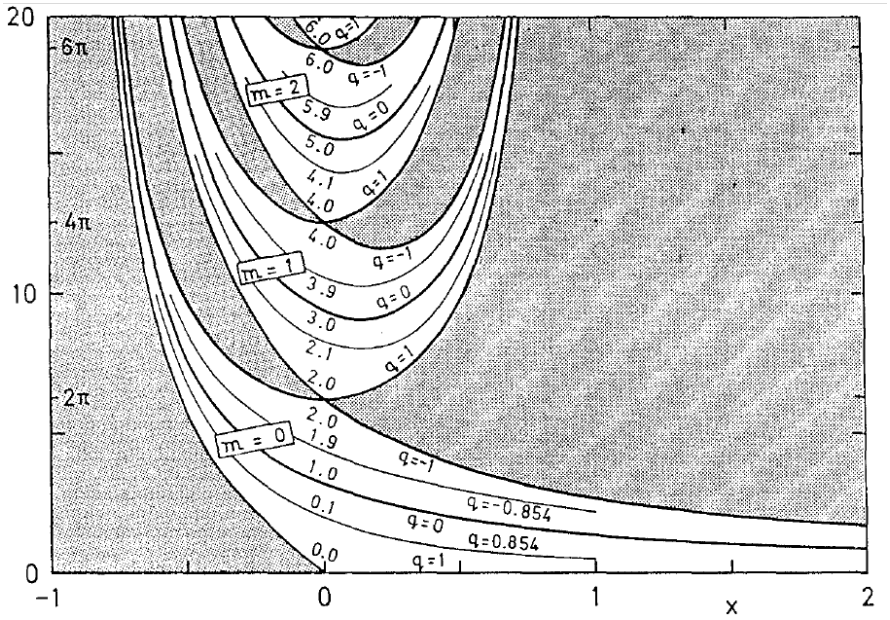


Figure C.1 The complete solution space for the function $T(x)$ invented by [Lancaster and Blanchard 1969] to solve the general high-thrust Lambert-problem. Valid solutions lie inside the white areas; no solutions exists in the shaded areas. This is Figure 1 from [Gooding 1990].

The case for $m \neq 0$ is somewhat more involved. When $m \neq 0$, the situation may arise that there are *no solutions* for a given set of boundary values. Therefore, it first has to be verified that there are indeed solutions for the given boundary conditions. To that end, the minimum T_M of the curve corresponding to the given m is first computed (see Figure C.1). Finding T_M must also be done iteratively, and thus an initial estimate x_M for the location of the minimum must be given. Good values were found to be

$$x_M = \begin{cases} x_{M,\pi} \left(\frac{\phi}{\pi}\right)^{1/8} & \text{if} \quad \phi < \pi \\ x_{M,\pi} \left(2 - \left[2 - \left(\frac{\phi}{\pi}\right)\right]^{1/8}\right) & \text{if} \quad \phi > \pi \end{cases} \tag{C.5}$$

where $x_{M,\pi} = 4/\left(3\pi\left[2m+1\right]\right)$ and $\phi = 2\texttt{atan2}(2q,\, 1-q^2)$, as before. With this initial value, T_M can be found. If its value turns out to be larger than the required $\hat{T}$, the problem has no solution, and further calculations are not required. If the problem turns out to have two solutions, *both* initial estimates from Equation C.4 are to be used to find them.

[Gooding 1990] also shows that using Halley's method (see appendix E.1.1) to find the root of Equation C.1 gives the most stable results. Also, finding the value for x_M in Equation C.5 can best be found with Halley's method. Extensive numerical experimentation showed that

Halley's method generally requires *more* iterations than Newton-Raphson's method (~ 5 versus ~ 3), but converged *always*, unlike the Newton-Raphson scheme.

One drawback for using Halley's method to find the correct x and x_M is that no less than *three* derivatives are required of the function $T(x)$. Luckily, the first of these is already given by Equation C.2, and the remaining two may be derived from it by repeatedly using the chain- and product rules from calculus:

$$\frac{d^2T}{dx^2} = \frac{\frac{-4q^3}{z}\left(1-\frac{q^2x^2}{z^2}\right) - 3T(x) - 3x\frac{dT}{dx}}{E} \tag{C.6}$$

$$\frac{d^3T}{dx^3} = \frac{\frac{4q^3}{z^2}\left(\left[1-\frac{q^2x^2}{z^2}\right] + \frac{2q^2x}{z^2}(z-x)\right) - 8\frac{dT}{dx} - 7x\frac{d^2T}{dx^2}}{E} \tag{C.7}$$

with symbols as before. It is repeated that these derivatives are not accurate near $x = 0$ or $x = 1$. For those cases, please refer to [Gooding 1990].

C.1.2 Long- or Short Way Solutions

Note that strictly speaking, there are two solutions not only to cases where $m > 0$, but to *every* Lambert-problem. This is due to the fact that in every geometry, the vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ define *two* values for the turn angle $\Delta\theta_1$ and $\Delta\theta_2 = 2\pi - \Delta\theta_1$. Of course, this also leads to two *different* possible trajectories, as shown in Figure C.2. In most literature, these two different paths are often referred to as the *long way* or *short way* solutions for the large- or small-angle, respectively[2]. Note that also for $m > 0$ the complementary turn angle $\Delta\theta_2$ may be used, so that there are *four* possible solutions in such cases.

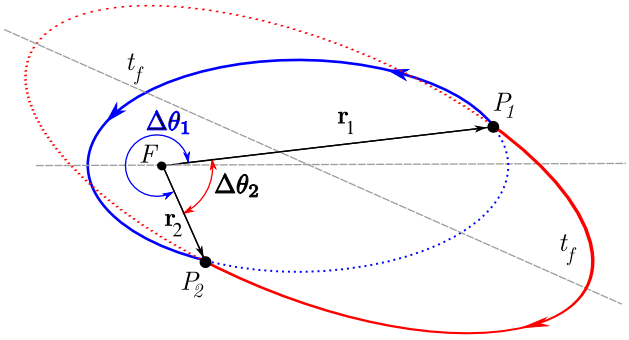


Figure C.2 The vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ cannot uniquely define the turn angle, giving rise to two different turn angles $\Delta\theta_1$ and $\Delta\theta_2$ in each Lambert problem. Therefore, two different trajectories also result, indicated here with the blue ($\Delta\theta_1$) and red ($\Delta\theta_2$) orbits.

In many MGA implementations only the prograde solutions are used. This gives both an

[2]Note that "short" or "long" have nothing to do with the time of flight – this is the same for both trajectories

easier implementation and prevents calculating (usually costly) retrograde solutions. However, this assumption is rather tenuous in the MPC-framework – after performing a GAM, the geometry for the next leg might be such that the retrograde solution resulting from the GAM is actually *more advantageous* than the prograde one. Such a scenario is illustrated in Figure C.3. It therefore seems necessary to make no such assumption, and analyze *both* solutions simultaneously.

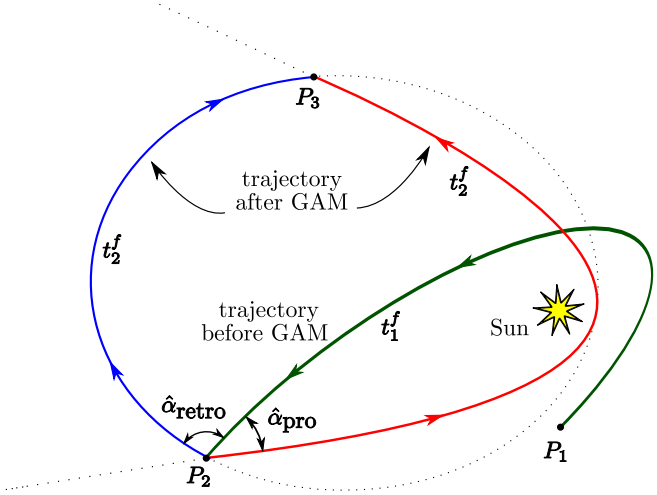


Figure C.3 In the geometry shown, the prograde solution would be a hyperbolic trajectory passing extremely close to the Sun, and would require a very large turn angle α at the flyby-planet P_2. The retrograde solution however, has none of these problems. Note that in this figure, $\hat{\alpha} \neq \alpha$; although the two angles are related, the angle $\hat{\alpha}$ actually describes the angle *complementary* to the turn angle α (see Figure 9.3).

Another argument in favor of using retrograde solutions in parallel with prograde ones, is that mission scenario 4 requires exploration of as many minor planets as possible. It can be expected that the largest amount of MP's will be encountered if the spacecraft travels on a retrograde trajectory – most MP's move on *prograde* orbits, so travelling in the *opposite* direction will increase the chances for an encounter. Indeed, this was also one of the results I had found in an earlier study (see [Oldenhuis 2009])[3]. Also, when the direction of the trajectory is reversed at some GAM, it will likely remain reversed throughout the mission. This leads to much larger V_∞ at the pericenters of all following swingby bodies, which in turn leads to a much higher possible energy gain for all following GAM's (see section 9.4 or Figure 9.6). For these reasons, *both* results must be considered during the optimizations. The penalty for doing so is the introduction of another $\mho$ dimensions to the solution space, which considerably increases the overall computational complexity.

The long way solution to Lambert's problem quite easily follows from Equations C.1 through C.3. It can readily be concluded that using either of the angles

[3] An argument *against* this would be that the relative velocity of the spacecraft with respect to the MP would be considerably higher, leading to much less contact time. However, the main objective of mission scenario 4 is *quantity*; the *quality* is a secondary consideration.

$$\Delta\theta = \arccos\left(\frac{\boldsymbol{r}_1 \cdot \boldsymbol{r}_2}{r_1 r_2}\right) \tag{C.8}$$

$$\Delta\tilde{\theta} = -(2\pi - \Delta\theta) = \Delta\theta - 2\pi \tag{C.9}$$

will automatically yield the correct magnitude and direction of the terminal velocity vectors. Note the minus sign in front of Equation C.9; this is necessary to account for travelling in the opposite direction the original algorithm had in mind.

C.2. Low thrust: Izzo's Algorithm for Exponential Sinusoids

C.2.1 Two Dimensions

[Petropoulos and Longuski 2004] used the equations from section 12.2 to design a low-thrust mission to Ceres, the largest body in the MAB. They used a relatively simple *forward-targeting* technique, which successfully yielded all the required parameters k_0 through φ for this mission. Although their approach was successful, it was nevertheless very tedious to apply to other mission concepts, as it is quite mission-specific. Consequently, Izzo [2006] outlined a more general method by solving the general Lambert-problem for ExpoSin's, using the equations given in section 12.2.

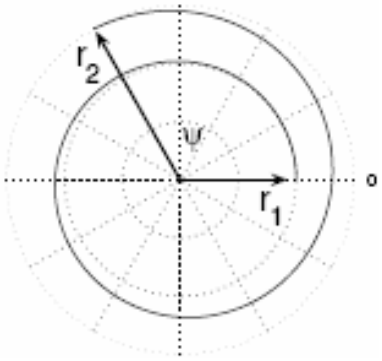


Figure C.4 Definition of parameters for solving Lambert's problem for spiral trajectories. Figure copied from [Petropoulos 2001].

His approach is as follows: choose the coordinate system such that the departure vector $\boldsymbol{r}_1$ is aligned with the x-axis, and the target vector $\boldsymbol{r}_2$ lies in the xy-plane, with a turn angle Ψ from $\boldsymbol{r}_1$ (see Figure C.4). It then follows that

$$\begin{aligned} \bar{\theta} &= \Psi + 2\pi m,\ m = 0,\ 1,\ 2, ... \\ r_1 &= k_0 e^{k_1 \sin(\phi)}, \\ r_2 &= k_0 e^{k_1 \sin(k_2\bar{\theta}+\phi)}. \end{aligned} \tag{C.10}$$

Note that the additional integer m takes care of multi-revolution cases. From Equation 12.12, the flight path angle γ_1 at the departure point $\boldsymbol{r}_1$ is

$$\tan\gamma_1 = k_1 k_2 \cos\varphi. \tag{C.11}$$

When values for k_2, γ_1 and m are assumed, Equations C.10 and C.11 constitute a set of 3 equations in the 3 unknowns k_0, k_1 and φ. These equations can be solved explicitly. The solution for the magnitude of k_1 is

$$k_1^2 = \left(\frac{\ln(r_1/r_2) + \frac{\tan\gamma_1}{k_2}\sin(k_2\bar{\theta})}{1-\cos(k_2\bar{\theta})}\right)^2 + \frac{\tan^2\gamma_1}{k_2^2}, \tag{C.12}$$

while its sign can be found from

$$\text{sign}(k_1) = \text{sign}\left(\ln\frac{r_1}{r_2} + \frac{\sin(k_2\bar{\theta})\tan(\gamma_1)}{k_2}\right), \tag{C.13}$$

with $\text{sign}(x) = -1$ if $x < 0$ and $\text{sign}(x) = 1$ if $x > 0$. The solutions for the other two variables are somewhat simpler:

$$\varphi = \arccos\left(\frac{\tan\gamma_1}{k_1 k_2}\right), \tag{C.14}$$

$$k_0 = r_1 e^{-k_1 \sin\varphi}. \tag{C.15}$$

From all of these parameters, the corresponding value for the time of flight may be calculated using Equation 12.7.

A Lambert *algorithm* built around these equations must first perform the simple check given in Equations 12.13 and 12.14 to see whether the selected value for k_2 can solve the problem. If so, the algorithm should continue by re-iterating all of the above for different values of γ_1, in order to find *that* value for γ_1 which satisfies the *required* time of flight (TOF_{req}). Doing this efficiently is no easy task, since the equation for the time of flight involves an integration that can only be carried out numerically. Moreover, although this was not included in the [Izzo 2006] paper, the value of the integral for different values of γ_1 does *not* simply increase monotonously with γ_1, which makes finding its root(s) rather tedious. This somewhat lengthy subject is discussed in section C.2.4.

However, when the correct value(s) for γ_1 has (have) been found, all the ExpoSin parameters k_0, k_1, k_2, ϕ, N and $\Delta\theta$ are known, the terminal velocities $\boldsymbol{V_1}$ and $\boldsymbol{V_2}$ can then be calculated by substituting the angles θ_1 and θ_2 into Equation 12.5, and the derivative of Equation 12.4 with respect to time:

$$\begin{aligned} \dot{r} &= k_0 k_1 k_2 c\dot{\theta} e^{k_1 s} \\ &= \tan\gamma \cdot r\dot{\theta} \end{aligned} \tag{C.16}$$

with s and c as in Equations 12.10 and 12.11. This may be written more clearly as

$$\begin{aligned} \boldsymbol{V}_1 &= V_{r_1}\hat{\boldsymbol{r}}_1 + V_{\theta_1}\hat{\boldsymbol{\theta}}_1, \\ \boldsymbol{V}_2 &= V_{r_2}\hat{\boldsymbol{r}}_2 + V_{\theta_2}\hat{\boldsymbol{\theta}}_2, \end{aligned}$$

where $\hat{\boldsymbol{\theta}}_{12}$ and $\hat{\boldsymbol{r}}_{12}$ are the corresponding unit vectors in radial and tangential direction for the initial and terminal points, respectively, and

$$\begin{aligned} V_{r_{12}} &= k_0 k_1 k_2 \dot{\theta}(\theta_{12}) c_{12} e^{k_1 s_{12}}, \\ V_{\theta_{12}} &= \dot{\theta}(\theta_{12}). \end{aligned}$$

The unit vectors $\hat{\boldsymbol{\theta}}_{12}$ and $\hat{\boldsymbol{r}}_{12}$ can be found from the (two dimensional) problem geometry. Since $\boldsymbol{r}_1$ is aligned with the x-axis,

$$\begin{aligned} \hat{\boldsymbol{\theta}}_1 &= [0,\, 1,\, 0]\,, \\ \hat{\boldsymbol{r}}_1 &= [1,\, 0,\, 0]\,. \end{aligned}$$

and for $\boldsymbol{r}_2$,

$$\begin{aligned} \boldsymbol{r}_2 &= [r_2 \cos\Psi,\, r_2 \sin\Psi,\, 0]\,, \\ \hat{\boldsymbol{\theta}}_2 &= [-\boldsymbol{r}_2^y,\, \boldsymbol{r}_2^x,\, 0]\,/r_2, \\ \hat{\boldsymbol{r}}_2 &= \boldsymbol{r}_2/r_2, \end{aligned}$$

with r_2 the magnitude of $\boldsymbol{r}_2$. To check whether the final results are correct, the relation between the terminal flight path angles may be used:

$$\tan\gamma_2 = \tan\gamma_{1,\min} + \tan\gamma_{1,\max} - \tan\gamma_1. \tag{C.17}$$

When all of this is done, the value found for γ_1 must be substituted in Equation 12.15 to see whether the given combination of γ_1 and k_2 give a physically realistic solution.

C.2.2 Long-way and Short-way Solutions

Just as in the high-thrust Lambert problem, there is an ambiguity in the definition of the turn angle $\Delta\theta$. The two angles $\Delta\theta$ and $\Delta\tilde{\theta} = 2\pi - \Delta\theta$ each give rise to a valid solution. Unlike the high-thrust case, Equations 12.4 through C.16 show that it does not suffice to simply use the negative of the complementary angle $-\Delta\tilde{\theta}$ to obtain the long-way solution. This solution must be found by using the *positive* angle $\Delta\tilde{\theta}$ in all of the equations, and then negating *only the tangential component* of the terminal velocity vectors:

$$\begin{aligned} \tilde{\boldsymbol{V}}_1 &= V_{r_1}\hat{\boldsymbol{r}}_1 - V_{\theta_1}\hat{\boldsymbol{\theta}}_1, \\ \tilde{\boldsymbol{V}}_2 &= V_{r_2}\hat{\boldsymbol{r}}_2 - V_{\theta_2}\hat{\boldsymbol{\theta}}_2, \end{aligned}$$

with $\tilde{\boldsymbol{V}}_{12}$ the terminal velocities for the second solution, and $\hat{\boldsymbol{\theta}}_{12}$ and $\hat{\boldsymbol{r}}_{12}$ the corresponding unit vectors.

C.2.3 Three Dimensions

The coordinate system chosen by Izzo [2006] in the above procedure is such that $\theta_1 = 0$ ($\boldsymbol{r}_1$ lies along the x-axis), and all motion takes place in the xy-plane. Also, [Petropoulos 2001] used the notion of reference planes to keep all motion in two dimensions. This is not a very convenient assumption when the objective is to visit asteroids or other bodies that lie well outside this reference plane. Also, after launch, the spacecraft's V_∞-vector can have any direction and magnitude (within the limits of the launch vehicle of course), so that taking the

reference plane to be the Earth's orbital plane also seems somewhat inappropriate.

However, since this method will only be used as a first-order approximation to get a rough idea about the feasibility of a certain configuration, it seems a good idea to simply transform the reference plane from the orbital plane of the departure planet, to the plane defined by the departure planet, target planet and the Sun – the same as is done in the high-thrust Lambert procedure.

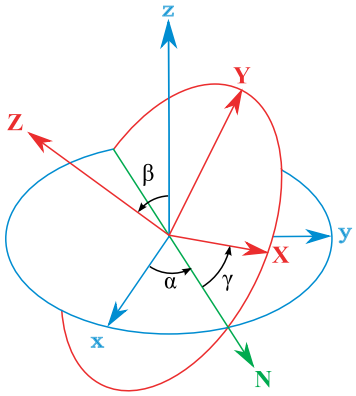


Figure C.5 Definition of the Euler-rotation angles. Here, N is the unit vector along the line of nodes. From [Wikipedia 2009].

To do so, only a simple coordinate transformation is required to transform the results from the two-dimensional case back to the original three-dimensional geometry. This transformation is most easily carried out using the Euler-angles formulation. Note that this transformation only applies to the terminal velocities $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ resulting from the Lambert procedure, since the ExpoSin will only contain the polar coordinates that already lie in the new (two dimensional) transfer plane.

A general Euler transformation from one Cartesian frame into another with the same origin, may be carried out by performing three consecutive rotations of the original frame. These rotations are defined by three angles, and three axes about which the frame is rotated by each of the three angles. Unfortunately, there is no unique way to define these angles and axes – the order in which these rotations are performed changes the definition of the axes and angles.

There are no less than 12 unique ways to define three rotations that have the same outcome. Here, the $Z-X-Z$ rotation-order will be used. With this definition, a transformation from a reference frame $[X,\, Y,\, Z]$ to $[x,\, y,\, z]$ may be performed by the multiplication

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}^T \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.18}$$

where the angles α, β and γ are defined as in Figure C.5.

However, for the required transformation, both reference frames are known, but the angles α, β and γ are not. To find the angles from the known orientation, the following relations may be derived (see Figure C.5):

$$\begin{aligned} \alpha &= \texttt{atan2}\left(Z_1,\, -Z_2\right), \\ \beta &= \texttt{atan2}\left(\sqrt{Z_1^2 + Z_2^2},\, Z_3\right) \\ \gamma &= \texttt{atan2}\left(X_3,\, Y_3\right) \end{aligned}$$

where `atan2`(y, x) is the four quadrant arctangent function, and the subscripts $_{1,2,3}$ denote the individual components of the corresponding vectors. With these angles now known, the required coordinate transformation may be carried out by Equation C.18.

It should be emphasized that performing such a transformation completely abandons the assumptions that [Petropoulos 2001] and [Izzo 2006] made in the design of the ExpoSin method. Therefore, it may be expected that the assumption of "low"-thrust will not be reasonable at many points along the MGA trajectory optimized with this Lambert-routine. However, in the MGA framework, it seems a reasonable assumption to make, since the large rotations from one frame to the next can be accounted for by the generalized three-dimensional GAM's executed along the trajectory. Thus, this somewhat inappropriate usage of the method can be well corrected by a proper design of the cost functions to be used for the MGA method, which is the subject of chapter 14.

C.2.4 Routine to find γ_1

[Izzo 2006] concluded after extensive numerical experimentation that the function $F(\gamma_1)$ increases monotonically for all choices of the parameters m and k_2, as shown in Figure C.6. Although he never proved this monotonicity, he continued by finding the (single) value for γ_1 that yields the required time of flight TOF_{req} using a simple Regula Falsi root-finding algorithm. Assuming monotonicity, this root would be known to lie in the interval

$$\tan \gamma_{m,M} = \frac{k_2}{2}\left[-\ln\frac{r_1}{r_2}\cot\frac{k_2}{2} \pm \sqrt{\Delta}\right], \tag{C.19}$$

where

$$\Delta = \frac{2(1-\cos k_2\bar{\theta})}{k_2^4} - \ln^2\frac{r_1}{r_2}.$$

Trajectories for which $\Delta < 0$ are infeasible, and may thus be skipped without further consideration. For feasible solutions, the correct value for γ_1 must be found by computing the root of the function

$$F(\gamma_1) = \sqrt{\frac{1}{\mu}}\int_0^{\Delta\theta}\sqrt{\frac{r^3(\theta)}{\Gamma(\theta,\gamma_1)}}d\theta - TOF_{\text{req}}, \tag{C.20}$$

with $\Gamma(\theta,\gamma_1)$ as in Equation 12.7, and TOF_{req} the required time of flight. However, as found by Corradini [2009], for certain combinations of the parameters k_2 and m, the value of the time-of-flight integral $F(\gamma_1)$ does *not* increase monotonically. Instead, it can assume one of three distinct shapes: monotonically *in*creasing, monotonically *de*creasing, or a "bathtub"; first decreasing, then increasing. See Figure C.7 for an example of these different shapes.

To detect which shape the time-of-flight integral will have for given initial values and m and k_2, it is most convenient to have derivative information of the function $F(\gamma_1)$. This derivative

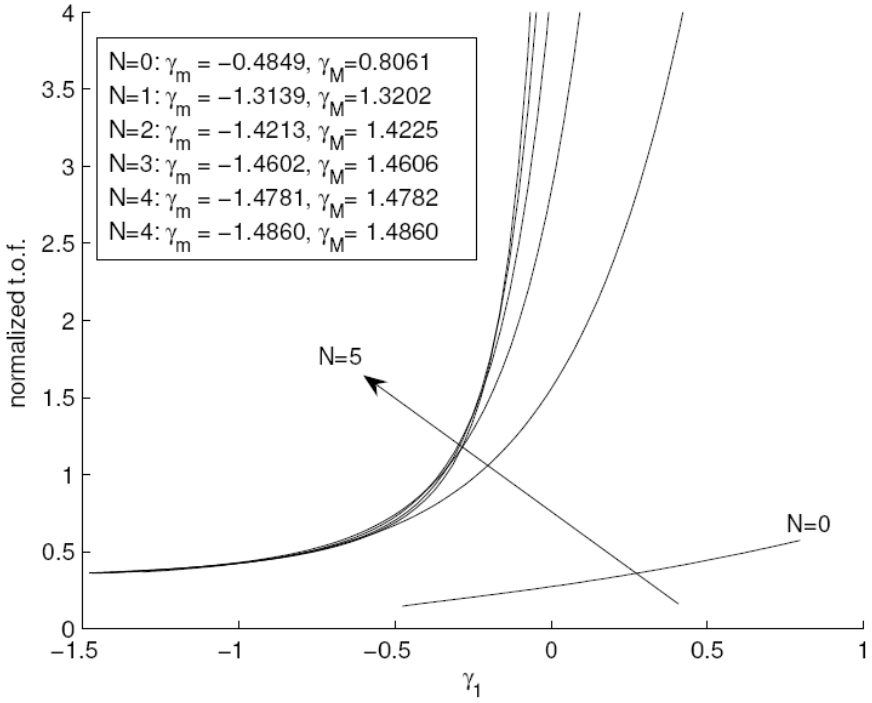
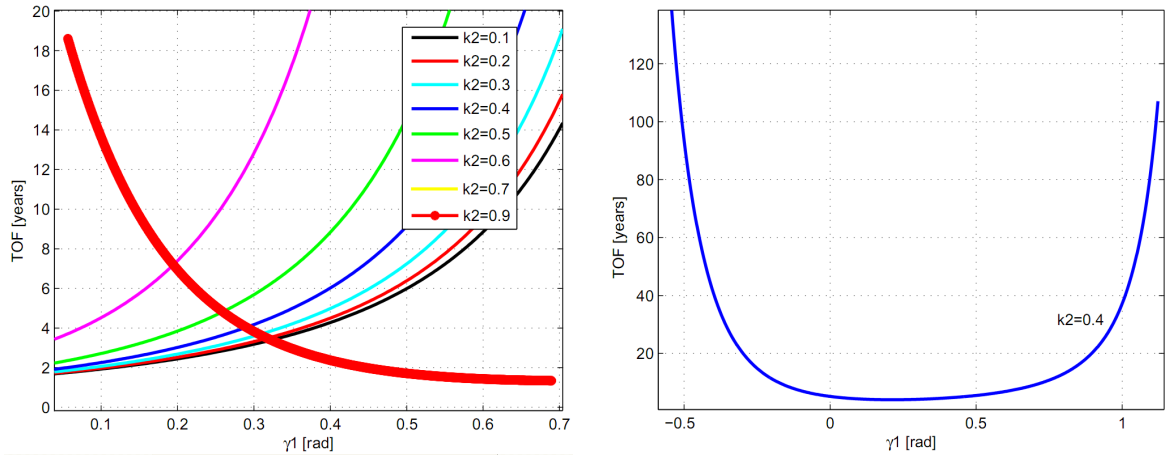


Figure C.6 Time of flight (normalized to years) as a function of γ_1 for the ExpoSin's Lambert-problem. This is Figure 2 from [Izzo 2006].



(a) Example of a monotonously *de*creasing time-of-flight

(b) Example of a "bathtub"; first decreasing, then increasing

Figure C.7 Some examples of the non-monotonicity of the value of the time-of-flight integral $F(\gamma_1)$. The value of the integral can be monotonously de– or increasing, or even first decrease and then increase again. Adopted from Figures 7.10 and 7.20 from [Corradini 2009].

may be found (although rather tediously) by employing the Leibniz integral rule, better known as *differentiation under the integral sign*. Using this technique, the derivative is found to be

$$
\begin{aligned}
F'(\gamma_1) = \frac{dt_f}{d\gamma_1} &= \int_{\theta_0}^{\theta_f} \frac{\partial}{\partial \gamma_1} \sqrt{\frac{r^3\Gamma}{\mu}} d\theta \\
&= \frac{1}{2\sqrt{\mu}} \int_{\theta_0}^{\theta_f} \frac{3\Gamma r^2 \frac{\partial r}{\partial \gamma_1} + r^3 \frac{\partial \Gamma}{\partial \gamma_1}}{\sqrt{r^3\Gamma}} \\
&= \frac{1}{2\sqrt{\mu}} \int_{\theta_0}^{\theta_f} \frac{\frac{3r^3\Gamma}{k_0}\left(\frac{\partial k_0}{\partial \gamma_1} + k_0\Phi\right) + r^3\left(2k_1k_2\frac{\partial \Phi}{\partial \theta}\cos(k_2\theta+\phi) + k_2^2\Phi\right)}{\sqrt{r^3\Gamma}}
\end{aligned}
\tag{C.21}
$$

In this expression,

$$
\begin{aligned}
\Gamma &= \tan^2\gamma + k_1k_2^2 \sin\left(k_2\theta + \phi\right) + 1 \\
\Phi &= \frac{\partial k_1}{\partial \gamma_1}\sin\left(k_2\theta+\phi\right) + k_1\frac{\partial \phi}{\partial \gamma_1}\cos\left(k_2\theta+\phi\right)
\end{aligned}
$$

and

$$
\begin{aligned}
\frac{\partial k_1}{\partial \gamma_1} &= \frac{1}{2k_1}\left\{2\left(\frac{\ln\frac{r_1}{r_2} + \frac{\tan\gamma_1 \sin k_2\bar{\theta}}{k_2}}{1-\cos k_2\bar{\theta}}\right)\left(\frac{\left(1-\cos k_2\bar{\theta}\right)\left(\frac{\sin k_2\bar{\theta}\sec^2\gamma_1}{k_2}\right)}{\left(1-\cos k_2\bar{\theta}\right)^2}\right) + \frac{2\tan\gamma_1\sec^2\gamma_1}{k_2^2}\right\} \\
&= \frac{\sec^2\gamma_1\tan\gamma_1}{k_1k_2^2}\left\{\frac{\sin k_2\bar{\theta}}{\left(1-\cos k_2\bar{\theta}\right)^2}\left[\frac{k_2\ln\frac{r_1}{r_2}}{\tan\gamma_1} + \sin k_2\bar{\theta}\right] + 1\right\},
\end{aligned}
$$

$$
\frac{\partial \varphi}{\partial \gamma_1} = \frac{-\left(k_1k_2\sec^2\gamma_1 - \frac{\partial k_1}{\partial \gamma_1}k_2\tan\gamma_1\right)}{k_1^2k_2^2\sqrt{1-\frac{\tan^2\gamma_1}{k_1^2k_2^2}}} = \frac{\frac{\partial k_1}{\partial \gamma_1}\tan\gamma_1 - k_1\sec^2\gamma_1}{|k_1|\sqrt{k_2^2k_1^2 - \tan^2\gamma_1}},
$$

$$
\frac{\partial k_0}{\partial \gamma_1} = -k_0\left(\frac{\partial k_1}{\partial \gamma_1}\sin\varphi + \frac{\partial \varphi}{\partial \gamma_1}k_1\cos\varphi\right).
$$

Note that the usage of $\partial\Phi/\partial\theta$ is used for brevity; the fact that the outcomes are the same is purely coincidental. Note also that many terms appear in both the derivative and the function itself, so that the implementation can easily be made most efficient by recycling these values in cases where both values are needed simultaneously. Evaluating Equation C.21 at the points γ_m and γ_M will give a certain combination of positive and/or negative values, from which the correct shape easily follows.

To find the actual root, it is most convenient to first perform some tests on the function $F(\gamma_1)$. Extensive numerical experimentation showed that the values for γ_m and γ_M given by Equation C.19 do not constrain the γ_1–search space strongly enough; using these values can lead to numerical overflow or underflow. This is because Equation C.19 does not take into account the constraint equation Equation 12.16. As an example, consider the case

$$\begin{aligned} k_2 &= 1/12 \rightarrow |k_1| \leq 144 \quad \text{(from Equation 12.16)} \\ \rightarrow k_0 &= r_1 \cdot e^{\pm 144 \cdot \sin\varphi} \quad \text{(from Equation C.15)} \end{aligned}$$

which involves the computation of $\exp(\sim 144)$. Of course this will lead to extremely large (or small) values of k_0. This is important for the computation of the time-of-flight integral, since explicit evaluation of (at least) $k_0^{3/2}$ is required, or $\exp(1.5 \times 144) = \exp(216)$ in the example. This can lead to equally large inaccuracies in the evaluation of the integral (not to mention questionable physical reality). For smaller values of k_2 the integral can indeed be rendered incalculable altogether, which is not beneficial for the overall robustness of the Lambert routine. So instead of using γ_m and γ_M as the delimiters of the search space for γ_1, it is better to introduce a constraint that limits the value of k_1 so that all steps in the calculation are guaranteed to remain fully representable in `double`-precision. The largest value that can occur throughout the calculation is that for $k_0^{3/2}$, so a suitable constraint is

$$|k_1(\gamma_m,\, \gamma_M)| \leq \min\left(450, \frac{1}{k_2^2}\right), \tag{C.22}$$

where the value 450 is based on

$$\frac{1}{1.05}\frac{\ln(\texttt{realmax})}{1.5} \approx 450,$$

where the factor 1.05 is included as a safety factor, and `realmax` is the maximum numeric value representable by double-precision variables in a computer, $\texttt{realmax} \approx 1.8 \times 10^{308}$. If the value for k_1 at the initial values γ_m and/or γ_M (calculated with Equations C.12 and C.13) violates this constraint, the values for the boundaries γ_m and/or γ_M must be shifted towards each other until the constraint is met at both ends. All further checks on the feasibility must be carried out with these stricter values for γ_m and/or γ_M.

When $F(\gamma_1)$ is evaluated at the (new) points γ_m and γ_M, and the value of the derivative at these points is also known, it can easily be determined whether the function will ever reach as high (or as low) as the required value TOF_{req}. If this is not the case, the routine can be terminated and some null-result should be returned. This test is however somewhat more involved in case the function is a "bathtub". In those cases, it may be that the function values at both the end-points lie above the required value TOF_{req}, but this is *not* sufficient to conclude that the function will pass through TOF_{req}. To determine if that scenario applies, the minimum of the bathtub needs to be found at the expense of a few extra FE's. To do this, the root of the derivative from Equation C.21 must be found. A Newton-Raphson routine (using simple forward-differences to approximate the second derivative) with a starting value halfway between γ_m and γ_M was found to perform quite satisfactorily. The value of

Equation C.20 at this minimum should then be negative in order for solutions to exist. In case it is positive, a default null-result should be returned.

The Regula-Falsi method (used by Izzo [2006]) is not the best choice to find the actual root. This is because after these tests, the only values for γ_1 of which it is known that the corresponding function values lie above and/or below zero, are $F(\gamma_m)$ and $F(\gamma_M)$. For many realistic cases (especially for $m > 0$), $F(\gamma_m)$ and/or $F(\gamma_m)$ yield rather large values (order $\sim 10^{40}$ or higher is not uncommon). This behavior is also clearly shown in Figure C.7. Therefore, any line drawn from $F(\gamma_M)$ to $F(\gamma_m)$ will in many cases be *nearly vertical*, resulting in *extremely* small step sizes in the Regula-Falsi routine. Also, now that derivative information is available from Equation C.21, it seems rather wasteful not to use this in the root finding routine.

Notice that the time-of-flight integral curve behaves very much like an exponential function. This fact, and the fact that the derivative can now be computed, allows for a much more clever root-finding method. Assume that the curve can be reasonably approximated by an exponential equation of the form

$$F(\gamma_1) \approx Ae^{B\gamma_1} + C, \tag{C.23}$$
$$F'(\gamma_1) \approx ABe^{B\gamma_1} = De^{B\gamma_1}. \tag{C.24}$$

After performing the initial validity tests, both the function values and their derivatives at the end-points γ_m and γ_M are already known. From these values, the coefficients A, B and C easily follow:

$$B = \frac{\ln\left(F'(\gamma_m)/F'(\gamma_M)\right)}{\gamma_m - \gamma_M} \tag{C.25}$$
$$D = F'(\gamma_m)e^{-B\gamma_m} = F'(\gamma_M)e^{-B\gamma_M} \tag{C.26}$$
$$A = \frac{F(\gamma_m) - F(\gamma_M)}{e^{B\gamma_m} - e^{B\gamma_M}} = \frac{D}{B} \tag{C.27}$$
$$C = F(\gamma_m) - Ae^{B\gamma_m} = F(\gamma_M) - Ae^{B\gamma_M}. \tag{C.28}$$

As can be seen, the value for A can be found in two different ways. Since the function is not a true exponential, these two values will generally be different from one another. It is very tempting to use this difference to estimate the error made with this approximation, and extrapolate it away to yield a more accurate initial estimate. However, this would require additional FE's, so for now, simply the average of these two values will be used. With these coefficients, an initial estimate for the root-finder follows:

$$\gamma_1^0 = \frac{\ln\left(-C/A\right)}{B}. \tag{C.29}$$

When the function and its derivative are evaluated at this initial point, the whole procedure can be repeated to fit a new exponential function through this new point, and either γ_m or γ_M. Repeating this procedure over and over again, while consistently taking the previous leftmost or rightmost point to make the new fit, the root will quickly be found. Since

the Newton-Raphson method interpolates every new iterate *linearly* instead of this problem-specific *exponential* fit, it can be expected that convergence will be achieved faster than with the Newton-Raphson routine.

Unfortunately, this particular tactic to find the root only applies if $F(\gamma_1)$ is monotonous (either increasing of decreasing). If it is found however that the function is a bathtub, a slight modification is necessary. Bathtubs will generally yield *two* solutions (if one hasn't already been eliminated in the initial testing phase), which *both* require a solution. Generating initial estimates using only the endpoints is no longer possible, since the associated problems are under-determined. However, if the function value and its derivative would be known at two additional points, it would be possible to create two separate exponential approximations, to both sides of the bathtub. Therefore, for bathtubs, *four* additional FE's (twice both $F(\gamma_1)$ and $F'(\gamma_1)$) are necessary, which will be performed at the quarter-points of the interval:

$$\gamma_1^{0,1} = \gamma_m + \frac{\gamma_M + \gamma_m}{4} \tag{C.30}$$

$$\gamma_1^{0,2} = \gamma_m + 3\frac{\gamma_M + \gamma_m}{4}. \tag{C.31}$$

The complete procedure to find γ_1 can be summarized as follows:

1. Compute γ_m and γ_M, and the value for k_1 at these points. If Equation C.22 is not met, adjust the interval $[\gamma_m, \gamma_M]$ so that the constraint is met at both end points. In cases where the interval $[\gamma_m, \gamma_M]$ obtains zero length in order to satisfy the constraint ($\gamma_m \geq \gamma_M$), terminate the calculation.

2. Evaluate $F(\gamma_1)$ at the points γ_m and γ_M, with $F(\gamma_1)$ as in Equation C.20. If both values are negative, there is no solution and the procedure can be terminated.

3. If one is positive and the other negative, *or* if both are positive, evaluate $F'(\gamma_1)$ at the points γ_m and γ_M, with $F'(\gamma_1)$ as in Equation C.21.

4. If the values of the derivative at the end-points have *equal* sign, $F(\gamma_1)$ is monotonous. Use Equations C.23 through C.28 to find the root of $F(\gamma_1)$, and thus the desired value for γ_1. Terminate the procedure.

5. If the values of the derivative have *opposite* sign, $F(\gamma_1)$ follows a bathtub-shape. If a specific branch is desired (left or right branch), and the sign of the function value at the associated end-point is negative, no solution is possible; terminate the procedure in that case. If the the sign of the function value at the correct end-point is positive, find the minimum of the bathtub by finding the root of $F'(\gamma_1)$. If the value of $F(\gamma_1)$ at this root is positive, there are no solutions; terminate the procedure in that case. If the value is negative however, evaluate both $F(\gamma_1)$ and $F'(\gamma_1)$ at the associated quarter-point (Equation C.30 or Equation C.31). With this additional point, obtain an initial exponential fit using Equations C.23 through C.28, and use the exponential-fit root finding procedure to find the proper root of the function. Terminate the procedure.

D

Validation of Skipping Stone

D.1. Validating Basic Algorithms

D.1.1 STM / Coordinate Conversions

The ability to quickly and accurately calculate the positions of bodies that undergo motion in a central gravitational field is a crucial element of any trajectory optimizer; calculations that either progress the trajectory for some time or convert the representation of the body's statevector between different conventions must be executed by nearly every subroutine. The coordinate conversion routines used by Skipping Stone are already implicitly validated on numerous occasions (see appendix B for example). However, the STM-routine has not yet been validated, and because it is somewhat involved to implement it correctly it seems a good idea to at least compare the STM-routine against the coordinate conversion routines.

Such a test was carried out for three types of conic sections; circular, elliptic and hyperbolic. The rectilinear and parabolic cases were not tested both to keep this test small and simple, and because these cases are never encountered in practice. The cases considered are:

$$\begin{aligned}
&\text{Case 1:} \quad \boldsymbol{\chi}_0 = \left[1,\, 0,\, 0,\, 0,\, -\sqrt{\mu},\, 0\right] && \text{(Circular orbit)} \\
&\text{Case 2:} \quad \boldsymbol{\chi}_0 = \left[1,\, 0,\, 0,\, 0,\, -\sqrt{\tfrac{3}{2}\cdot\mu},\, 0\right] && \text{(Elliptic orbit)} \\
&\text{Case 3:} \quad \boldsymbol{\chi}_0 = \left[1,\, 0,\, 0,\, 0,\, -\sqrt{4\cdot\mu},\, 0\right] && \text{(Hyperbolic Trajectory)}
\end{aligned}$$

These trajectories were all progressed for 5 complete years in 183 steps by the STM-routine and with the triple coordinate conversion method. The vectorial difference between the results of both methods are plotted in Figure D.1. As can be readily concluded from these plots, the difference between the methods is marginal even after the full 5 years, so it can be concluded that the method is implemented correctly[1].

[1] The methods are actually quite the same – the different mathematics however completely occlude their

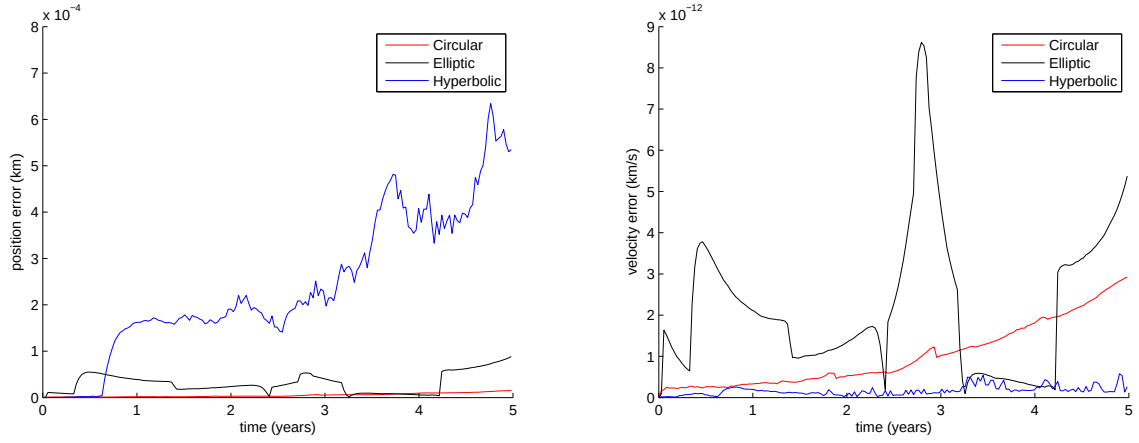
One remark should be made here: during these tests (and some further experimentation) it was found that, especially for multi-revolution elliptic orbits or highly-eccentric hyperbolic trajectories, the corrective Newton-Raphson step for u used by Shepperd [1984] (see Equation B.26):

$$u \leftarrow u - \frac{\delta t - \hat{t}}{4(1-q)r}$$

makes the method quite unstable; very often (roughly 10% of all cases) the method fails to keep $q < 1$ which results in a divergent continued fraction G, so the routine is forced to resort to the triple coordinate conversion. Using a *Halley* update instead of a Newton-Raphson update (see appendix E.1.1) proved to give a much more stable STM-routine; the method *never* resorted anymore to its fail-safe triple-conversion during all tests conducted. When re-substituting all substitutions made and collecting terms, the resulting Halley-step for u takes the shape

$$u \leftarrow u - \frac{\delta t - \hat{t}}{(1-q)(4r + \beta u(\delta t - \hat{t}))}, \tag{D.1}$$

with symbols as in section B.3.2. This update was indeed used for the test. Profiling records kept during these tests also showed that using the Kepler state transition matrix is an order of magnitude faster than the triple coordinate conversion method (which can be ascribed to the use of a continued fraction to solve Kepler's equation), so it is recommendable to use this method whenever an initial statevector needs to be progressed to a later epoch.



(a) Absolute errors in the positions

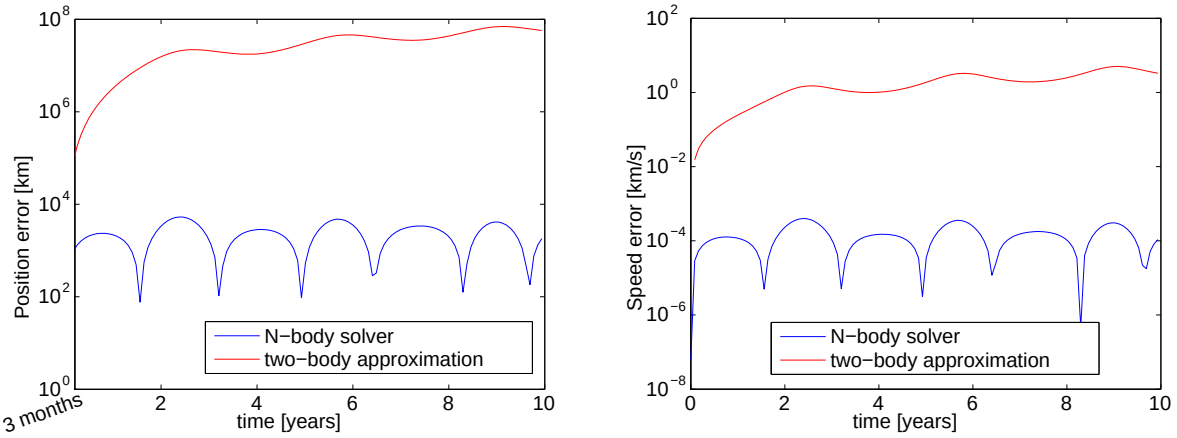
(b) Absolute errors in the velocities

Figure D.1 Comparison of the Kepler STM routine with the "brute-force" triple coordinate transformation method. The absolute errors between the two methods indicate that these methods agree with each other with a very acceptable accuracy. Because both methods calculate the same thing in a dissimilar fashion this can be regarded as corroboration to the correct functioning of both methods.

equality. The differences that *are* present most likely arise from the different tolerances used in the embedded methods to solve Kepler's equation.

D.1.2 Ephemerides and Integrator

Validating the ephemerides calculator has already (partly) been covered in appendix B. However, since the numeric integration of the state vectors of bodies in the Solar system also depends on the accuracy of the ephemerides of all its perturbing bodies, a numeric integration of asteroid 951 Gaspra was carried out to validate both the planetary ephemerides and the correctness of the implementation of the N-body integrator (using Encke's method and Battin's reformulation of the second derivative). The state vectors of this asteroid were integrated over a 10 year interval to demonstrate the error accumulation over longer intervals. The resulting state vectors were compared against Gaspra's ephemerides as generated by the JPL/HORIZONS system. For a better picture of the achieved accuracy, also a two-body approximation has been included. The results are shown in Figure D.2.



(a) Absolute position error

(b) Absolute speed error

Figure D.2 Validation of the N-body integrator using Encke's method, with Battin's reformulation of the second derivative. This figure shows the differences between the HORIZONS ephemerides and an integration/two-body approximation of asteroid 951 Gaspra over a 10-year interval.

The leftmost figure (position error) has been cropped slightly for better scaling; the first three months gave almost *no* error for both the two-body approximation and the integrator, which would overly compress the rest of the logarithmic plot if it were included. Note also that the position error for the N-body integrator does *not* increase over time – the calculated position seems to hover around the "true" position with an amplitude of $\sim 8.5 \times 10^3$ km. The remaining error can possibly be explained by taking too large an initial step, or by the fact that the integrator used by JPL/HORIZONS also takes into account several other perturbations (relativistic effects, precession of nodes, stellar aberration,...) The point is that the N-body integrator gives very reasonable accuracies, even over the 10 year interval considered.

In light of the position error reported by the two-body approximation one might suspect that the methods used in this research to calculate minimum distances to MP's (which employs two-body approximations extensively) will yield vastly inaccurate estimates, and possibly report nearby MP's that do not actually come close to the spacecraft in second-order analyses. However, keep in mind that the figure above displays a *10 year* interval; vastly longer than

any of the legs the spacecraft's trajectory will take (except of course, the leg to the SBS). The errors in the first half-year of the figure certainly *were* negligible, which is why they were excluded from the figure. Moreover, the possibility of large error with two-body approximations was kept in mind during the construction of the minimum-distance methods, which was *why* the position-dependent threshold was introduced (see Equation 11.6). Also, Gaspra is a member of the MAB, so its mean motion (and its error accumulation) is relatively large compared to outer Solar system bodies, which are to be visited in the last legs of the trajectory.

D.1.3 Lambert-targeter for Exponential Sinusoids

Also a central part of Skipping Stone are the various Lambert-targeting routines. Validation of the high-thrust Lambert-targeter (the one by Lancaster & Blanchard) is postponed to section D.2, where it is used to generate trajectories flown by actual spacecraft. Because locating the global optima for *low-thrust* trajectories follows largely the same lines as the high-thrust procedure, the only thing that requires additional validation is the ExpoSin Lambert-targeter.

The ExpoSin's-Lambert-targeter should at the very least be capable of reproducing the same results as found by its inventor ([Izzo 2006]). A small addition of about 25 lines of `M`-code sufficed to generate very similar results (see Figure D.3). However, note that the two curves for the cases $N = 0$ differ slightly; both their maximum values (~ 0.6 for Figure D.3(a) opposed to ~ 0.75 for Figure D.3(b)) and their curvature are different. The curve in Figure D.3(b) is consistent with that in [Corradini 2009; Fig. 6.7], which corroborates to its correctness. Moreover, replacing the term

$$\tan^2\gamma + k_1 k_2^2 \sin(k_2\theta + \varphi) + 1$$

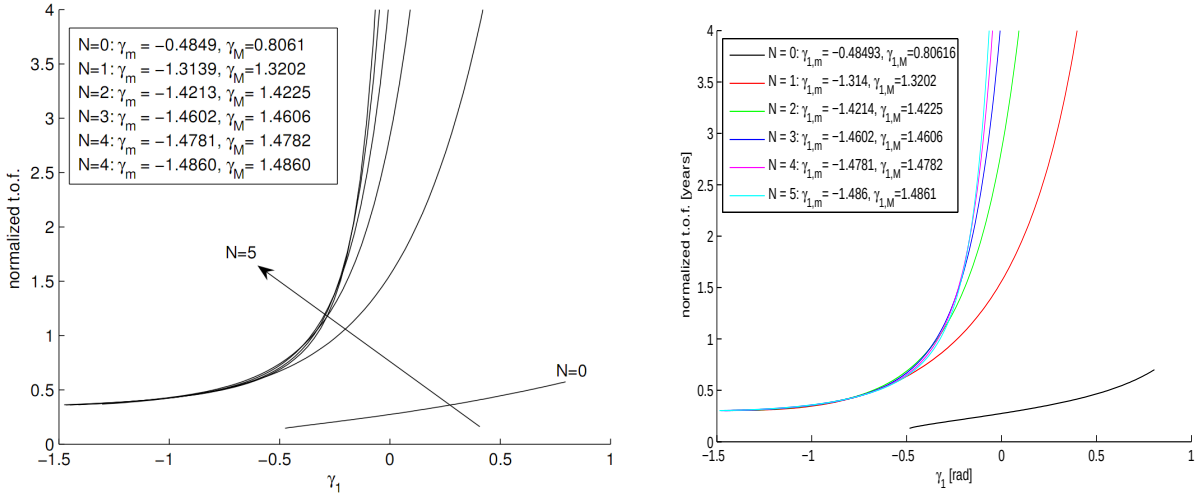
with

$$\tan^2\gamma_1 + k_1 k_2^2 \sin(k_2\theta + \varphi) + 1$$

in the time-of-flight integral (note the different γ's) *did* result in a more accurate reproduction of Izzo [2006; Figure 2]. However, at this point it must remain uncertain whether the two independent authors ([Corradini 2009] and the author of this thesis) indeed made the same mistake, but the main point is that the two figures agree to an acceptable level of accuracy.

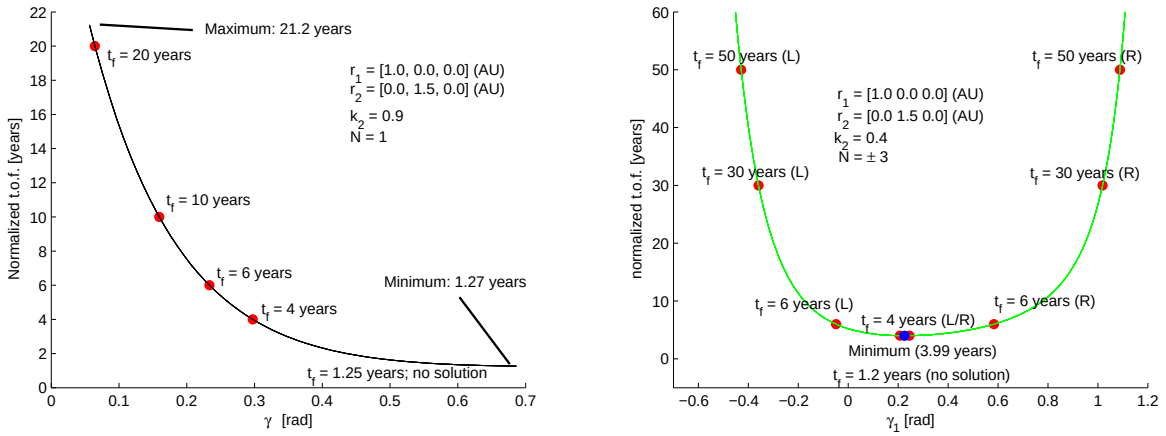
The ExpoSin-Lambert targeter should also be able to produce the correct ExpoSin in "problematic" cases, as those discussed by Corradini [2009]. To demonstrate the correct functioning of the whole procedure, two such cases were tested by "blindly" using the method for different values of t_f and plotting the reported values of γ_1. These results are shown in Figure D.4. As can be concluded from these figures, the values of γ_1 reported by the procedure indeed agree with the given times-of-flight. The corresponding number of iterations required were [5, 3, 3, 2, 0] for t_f = [20, 10, 6, 4, 1.25] years for the monotonously decreasing case, and [5, 5, 4, 2, 0] for t_f = [50, 30, 6, 4, 1.25] years for both branches in the bathtub case. The minimum of the bathtub was located in 7 iterations.

For the correct functioning of the method as outlined in section C.2.4 it is essential that the



(a) Figure 2 from [Izzo 2006] (b) The same Figure, generated by Skipping Stone

Figure D.3 Validation of the ExpoSin's Lambert-targeter; a reproduction of the figure published by its inventor.



(a) Monotonously decreasing case (b) Bathtub case

Figure D.4 Validation of the ExpoSin's Lambert-targeter; blindly finding the correct value of γ_1 for two different "problematic" cases.

lengthy and complicated derivative from Equation C.21 is indeed implemented *correctly*. To that end, the values for this derivative at all values for γ_1 for all the cases considered by Izzo [2006; Fig. 2] are compared against taking central-differences of the time-of-flight integral itself (see appendix E.2). The results of this small test are shown in Figure D.5.

Although not shown in this figure, experience leans that calculating the derivative given by Equation C.21 is *much* more sensitive to the type and settings of the quadrature method used. This is especially noticeable near the lower limit $\gamma_1 = \gamma_m$; the curves shown were all generated with an adaptive $7^{\text{th}}/15^{\text{th}}$-order Gauß/Kronrod method with ∼50 intervals (changed adaptively), giving a maximum error of ∼10% (black curve, $N = 0$). Re-doing this test with a

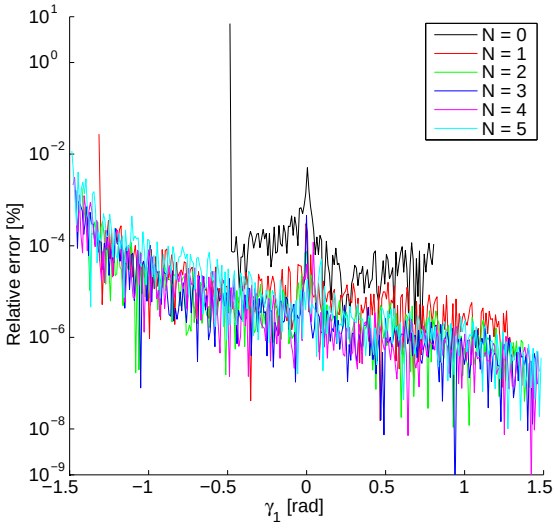


Figure D.5 A comparison of the derivative from Equation C.21 versus finite-differences. Note that the derivative is much more sensitive to the quadrature method used.

10-point Newton-cotes method with 400 intervals gave errors in the order of 50% (or more) in that particular region. This is a rather surprising result, because the accuracy of the second method *should* be higher than that of the first – the number of intervals does *not* cause additional roundoff error (as opposed to the order of the method), so more intervals *should* always give higher accuracy. Nevertheless, the results of this test prove that it is better to use the Gaußian quadrature method for all further calculations.

D.2. Reproducing Well-Known Trajectories

It is imperative that at the very least, Skipping Stone is able to reproduce the trajectories flown by well-known spacecraft in the past. Their trajectories are known to be (near) optimal, and because they were actually flown, all data on them have been published in an abundance of journals; their correctness is unquestionable. If Skipping Stone is indeed able to find their trajectories from a "blind" optimization, it would prove that the coordinate conversion routines, the optimizer, the Lambert-targeter, the ephemerides generator, the plotting routines, the patched-conics procedure, basically *everything*, works correctly. To this end, Skipping Stone will attempt to reproduce the Voyager 1 & 2 spacecraft's trajectories, the Cassini/Huygens spacecraft's trajectory, and the Galileo spacecraft's trajectory.

D.2.1 Voyager 1 & 2

A large number of initial trials and subsequent experimentation showed that the configuration of all the outer planets around the time the Voyager 1 & 2 probes were launched provided a window of opportunity *so wide* that it is not very difficult to find trajectories that visit all of the outer planets without using any propellant. In fact, this window of opportunity is *so large* that there is about a three-month time window around both Voyager's launch dates

Table D.1 Validation of Voyager 1 trajectory – detailed data.

(a) Real Voyager 1

Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Sep 5$^{\text{th}}$, 1977	–
Jupiter flyby	Mar 5$^{\text{th}}$, 1979	349,000 km
Saturn flyby	Nov 12$^{\text{th}}$, 1980	124,000 km
Totals		
ΔV		$\sim$0 km/s
travel time		3.19 years

(b) Voyager 1 according to Skipping Stone

Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Sep 3$^{\text{rd}}$, 1977	–
Jupiter flyby	Mar 2$^{\text{nd}}$, 1979	233,010 km
Saturn (arrival)	Nov 10$^{\text{th}}$, 1980	–
Totals		
ΔV		0.10 km/s
travel time		3.19 years

(and about two-month windows around the transfer times) in which nearly *every* trajectory following the Voyager's swingby-sequence can be flown with zero propellant usage; it is *very fortunate* that the mission designers of the time recognized this tremendous opportunity far enough in advance to use it to its fullest potential.

The ease with which the Voyager's trajectories can be reproduced necessitated using very small time windows around the actual transfer times (as found in [Angrum 2010]) in Skipping Stone to keep the solutions from "escaping" to one of the numerous local minima. Using a 5 day window around the transfer times produced Tables D.1 and D.2 and the trajectories shown in Figure D.6. As can be concluded from these data, constraining Skipping Stone to use the real transfer times resulted in a solution that requires non-zero propellant expenditure (albeit a small amount) and slightly different closest approach distances at the swingby planets. This can be explained (at least partially) by the inaccuracies with which the planetary ephemerides are calculated on all times prior to the year 2000, as discussed in section B. Regardless of these small differences, Skipping Stone's solutions agree very well with the real trajectories, so that Skipping Stoneseems to work correctly at least for these two cases.

D.2.2 Cassini/Huygens

NASA's/ESA's enormously successful Cassini/Huygens mission to Saturn provides an excellent opportunity to validate Skipping Stone. The Cassini/Huygens spacecraft used the GAM-sequence `seq` = [Earth, Venus, Venus, Earth, Jupiter, Saturn] to push the required amount of ΔV down to about 1.6 km/s[2]. The spacecraft's launch mass was a massive 5712 kg, while

[2]This value is mentioned in various publications on the Cassini/Huygens mission. However, it remains uncertain whether or not this ΔV-budget includes the ΔV required for Saturn orbit acquisition and manoeu-

Table D.2 Validation of Voyager 2 trajectory – detailed data.

(a) Real Voyager 2

Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Aug 20$^{\text{th}}$, 1977	–
Jupiter flyby	Jul 9$^{\text{th}}$, 1979	570,000 km
Saturn flyby	Aug 25$^{\text{th}}$, 1981	100,800 km
Uranus flyby	Jan 24$^{\text{th}}$, 1986	81,500 km
Neptune flyby	Aug 25$^{\text{th}}$, 1989	4400 km
Totals		
ΔV		$\sim$0 km/s
travel time		12.01 years

(b) Voyager 2 according to Skipping Stone

Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Aug 23$^{\text{rd}}$, 1977	–
Jupiter flyby	Jun 20$^{\text{th}}$, 1979	539,850 km
Saturn flyby	Aug 28$^{\text{th}}$, 1981	112,540 km
Uranus flyby	Jan 19$^{\text{th}}$, 1986	84,503 km
Neptune flyby	Aug 28$^{\text{th}}$, 1989	–
Totals		
ΔV		0.37 km/s
travel time		12.02 years

the empty orbiter/Huygens-probe together were 2445 kg; this implies the spacecraft had a total of 3123 kg of propellant at its disposal[3]. All these data can be found on [NASA/JPL 2009a], and deep-linked sources therein.

These data (plus the encounter dates) were inserted in Skipping Stone to see how close it can come to the trajectory actually flown (or perhaps, to see "how optimal" the real trajectory really was). For the first three optimizations, the lower and upper limits on the launch date and the times-of-flight between the planets were set 60 days lower and higher than the actual transfer times (upholding a minimum of 10 days). For the latter two optimizations, a window of 30 days was used. The outcomes in all five optimizations were (more or less) the same. The best results among these five optimizations are listed in Table D.3, and the corresponding trajectory is shown in Figure D.7.

It can readily be concluded from these results that, aside from some differences in the encounter times of at most one month (Venus I swingby), the results are quite similar. This is rather unexpected, since the actual Cassini/Huygens mission also used a DSM in between

vering inside the Saturnian system. Therefore, this value should be taken loosely; it it only to be used as an indication.

[3]Inserting these data in Tsiolkovskii's equation gives $\Delta V \simeq 1.78$ km/s, which indicates that the mentioned 1.6 km/s *included* all manoeuvres in the Saturnian system. However, the issue remains speculative since the actual data seem to be unavailable.

Table D.3 Validation of Cassini/Huygens trajectory – detailed data.

(a) Real Cassini/Hugens

Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Oct 15$^{\text{th}}$, 1997	–
Venus flyby I	Apr 26$^{\text{th}}$, 1998	234
Venus flyby II	Jun 24$^{\text{th}}$, 1999	600
Earth flyby	Aug 18$^{\text{th}}$, 1999	1171
Jupiter flyby	Dec 30$^{\text{th}}$, 2000	10.00×10^6
Saturn (arrival)	Jul 1$^{\text{st}}$, 2004	–
Totals		
ΔV		1.6 km/s (?)
mass prior to Saturn injection		3315.9 kg (?)
travel time		6.77 years

(b) Cassini/Hugens according to Skipping Stone

Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Oct 23$^{\text{rd}}$, 1997	–
Venus flyby I	May 21$^{\text{st}}$, 1998	333.33
Venus flyby II	Jun 26$^{\text{th}}$, 1999	625.1
Earth flyby	Aug 18$^{\text{th}}$, 1999	1003.3
Jupiter flyby	Jan 6$^{\text{th}}$, 2001	10.08×10^6
Saturn (arrival)	Jun 17$^{\text{th}}$, 2004	–
Totals		
ΔV		0.59 km/s
mass prior to Saturn injection		4576.9 kg
travel time		6.65 years

the two Venus swingby's (albeit one of relatively small magnitude) to allow the spacecraft to perform a flyby with the main belt asteroid 2685 Masursky. In conclusion: the differences between the real trajectory and that found by Skipping Stone are small enough to be considered the *same* solution; Skipping Stone's results thus agree with reality. This conclusion certainly corroborates to the correct functioning of Skipping Stone.

One remark is in place here: because the Cassini/Huygens mission flew prior to 1/1/2000, the planetary ephemerides must be computed with Kepler's method instead of the JPL/HORIZONS system – the reason for this was described in appendix B. However, as was also shown in appendix B, for the given interval violation of ∼6 years these errors are entirely negligible and do not contribute a significant component to the overall error. However, they might account (at least for a small part) for the differences in encounter dates reported by Skipping Stone.

D.2.3 Galileo

NASA's Galileo mission to Jupiter is expected to be more difficult to reproduce. Galileo was designed to be launched with the Space Shuttle Atlantis and use its powerful Centaur upper

stage booster rocket to *directly* insert it into a Hohmann-transfer orbit towards Jupiter. However, new regulations put in effect after the Challenger disaster forbade all following Shuttle missions to carry "bombs" into space, and the Centaur upper stage booster (liquid propellant) was considered to fall in this category. Therefore, the Galileo mission was forced to use a much less powerful booster, which necessitated a considerably delayed launch and the use of planetary swingby's to reduce the required ΔV to $\sim$1 km/s. However, as it was not part of the original mission design, it can be expected that the [Earth, Venus, Earth, Earth, Jupiter] sequence that it finally used is not a very strong attractor in the search space; if it was, the original design could have incorporated it to allow for greater expenditure of propellant inside the Jovian system, resulting in an overall more flexible and extendable mission.

Precise data on the Galileo spacecraft and its trajectory is listed on [NASA/JPL 2009b]. Its launch mass was 2223 kg, which included 925 kg of propellants. It also carried an atmospheric probe (which was successfully dropped into Jupiter's atmosphere) with a mass of 339 kg, so the total mass of the Galileo spacecraft adds up to 2562 kg. These data (plus the encounter dates) were again inserted in Skipping Stone to see how close it can come to the trajectory actually flown. A total of 189 optimizations were carried out, each with different values for the upper and lower limits on the times of flight and launch date, the number of complete revolutions, different values for the maximum launch C_3, different settings for the short/long way solutions, minimal approach distances, launch and dry masses, even different Lambert targeting routines and ephemerides generators, but all to no avail – Skipping Stone simply cannot reproduce the trajectory flown by the Galileo spacecraft. The best result found in all these optimizations is listed in Table D.4, and the associated trajectory in Figure D.8.

As expected, Galileo's trajectory is indeed quite hard to find; the encounter dates do seem to be an attractor[4], because even with a 75 day tolerance on the encounter dates, Skipping Stone manages to come reasonably close to the real dates. However, the *actual* dates could only be reproduced when the windows around the travel times were set to a tight 8 days; in that case, the required ΔV was in the order of $\sim$7 km/s. Only when these windows were broadened to 50 days (or more) could the solution listed in Table D.4(b) be found.

All ephemerides have been verified on numerous occasions with the JPL/HORIZONS web service, and they always agree to within $\sim$ 2000 km / $\sim$ 0.8 km/s. The main problem seems to lie in the Earth-Earth leg; As can be concluded from Figure D.8, the apocenter of the orbit flown during the Earth-Earth leg (as found by Skipping Stone) points to the "lower left", while in the real trajectory it points to the "lower right". This difference necessitates very large turn angles during both Earth swingby's in the Skipping Stone version, and consequently, very large ΔVs. It remains unknown what the cause of this phenomenon is. In the real trajectory, this particular leg is an orbit with a period slightly longer than 2 years so that the start and endpoints are nearly the same. This might confuse the optimizer when it comes close to the solution, on whether it should use 1 or 0 complete revolutions. However, this possibility has indeed been tested (by manually forcing $m = 0$ and $m = 1$) but does not seem to solve the problem. Multi-revolution solutions as reported by 3 different Lambert routines were found to agree for any starting values. Results from all Lambert routines for

[4]Albeit a *weak* attractor; there are *much* stronger ones when launching 11 months earlier or 5 months later

Table D.4 Validation of Galileo trajectory – detailed data.

(a) Real Galileo

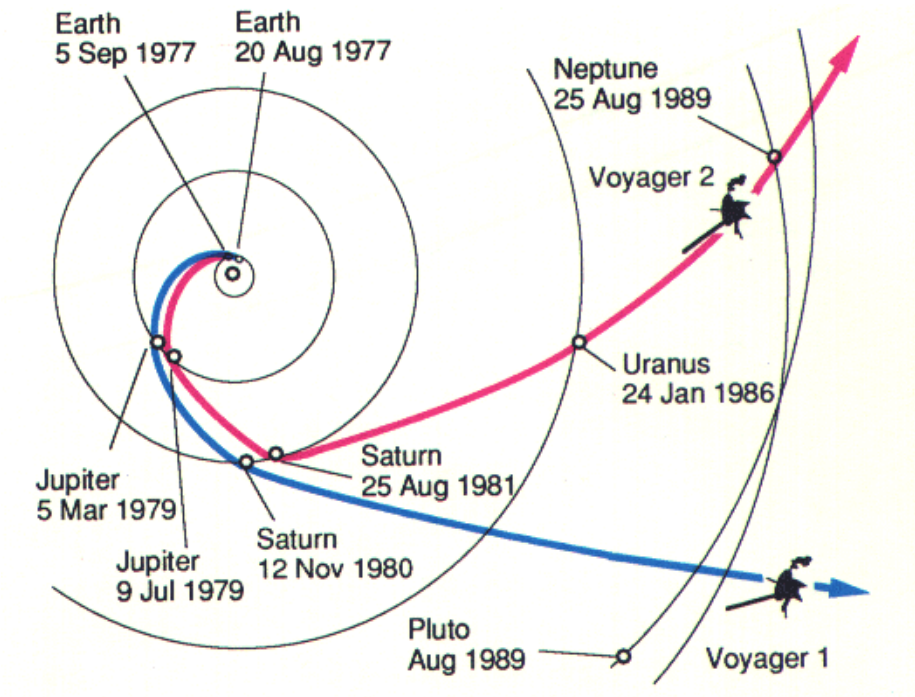
Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Oct 18th, 1989	–
Venus flyby	Feb 10th, 1990	16000
Earth flyby I	Oct 8th, 1990	960
Earth flyby II	Dec 8th, 1992	305
Jupiter (arrival)	Dec 7th, 1995	–
Totals		
ΔV		1.32 km/s (?)
mass prior to Jupiter injection		1673 kg (?)
travel time		6.14 years

(b) Galileo according to Skipping Stone

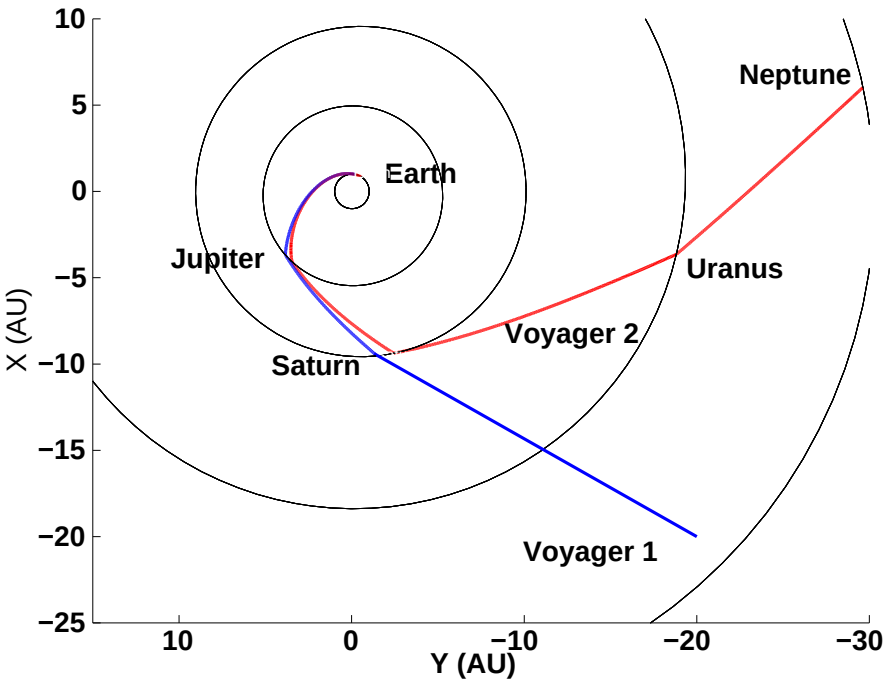
Body	**Date**	$r_{p,\min}[km]$
Earth (launch)	Oct 16th, 1989	–
Venus flyby	Feb 18th, 1990	384
Earth flyby I	Oct 20th, 1990	1798.4
Earth flyby II	Dec 28th, 1992	12466
Jupiter (arrival)	Jan 6th, 1996	–
Totals		
ΔV		2.86 km/s
mass prior to Jupiter injection		967.86 kg
travel time		6.22 years

the situation where $\boldsymbol{r}_1$ lies *just* before $\boldsymbol{r}_2$ and $m = 0$, or vice versa and $m = 1$, give very similar results; these situations seem to smoothly transform into each another.

At this point the possibilities on what the problem might be seemed exhausted. It was decided that it is no longer fruitful to try and find the cause of the disagreement. The main thing to focus on here is that optimizations carried out with Skipping Stone successfully approximate the real encounter dates (only not the ΔVs), and taking into account that both the Voyager trajectories and the Cassini/Huygens trajectory could be reproduced without problems, it can be concluded that Skipping Stone produces correct results in *most* cases. The remainder of this research will be carried out with this problem intact, which does render all results found somewhat uncertain. This issue is definitely something that deserves further attention if this mission is to enter Phase A, so it will be left as a recommendation.

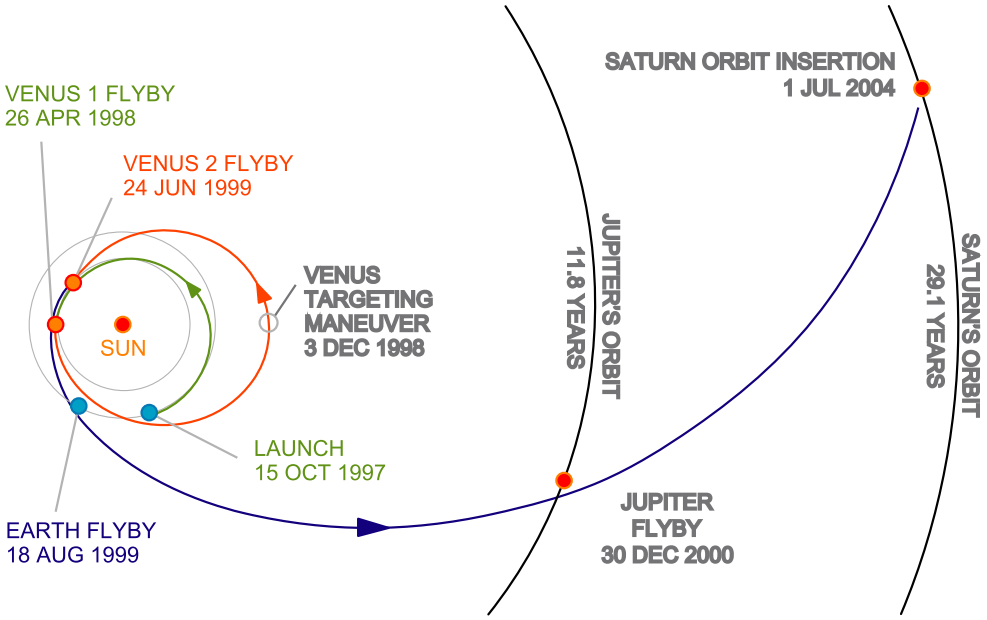


(a) Trajectories according to Decker et al. [2010]

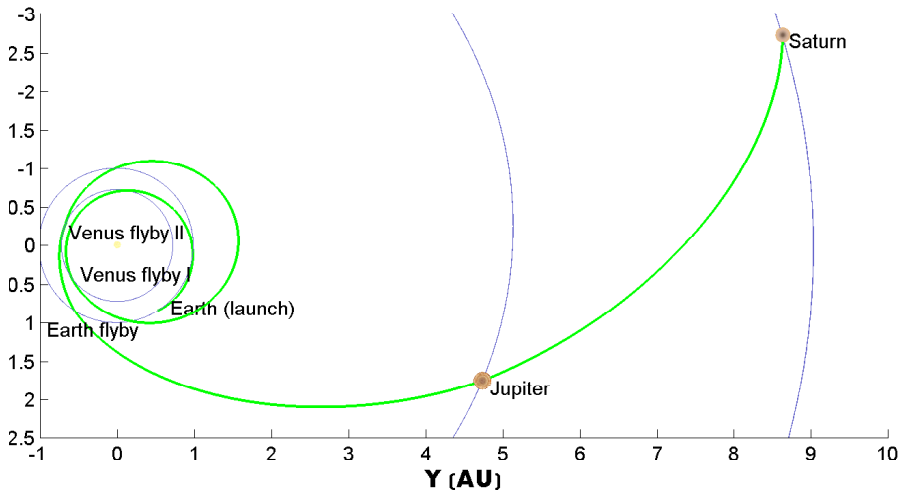


(b) Trajectories generated with Skipping Stone

Figure D.6 Validation of Voyager 1 & 2 trajectories – trajectories.

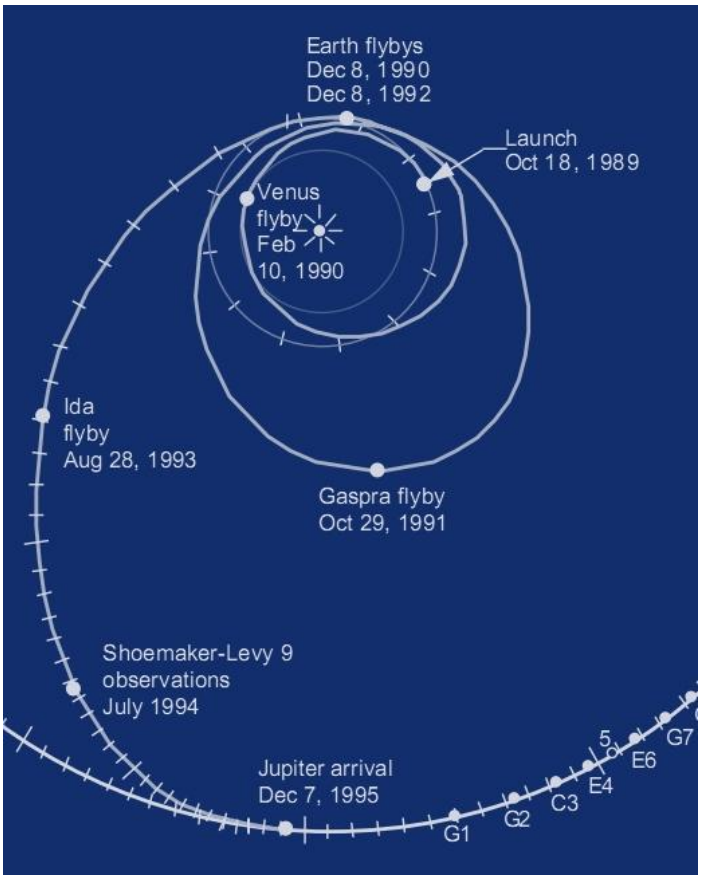


(a) Trajectory according to [NASA/JPL 2009a]

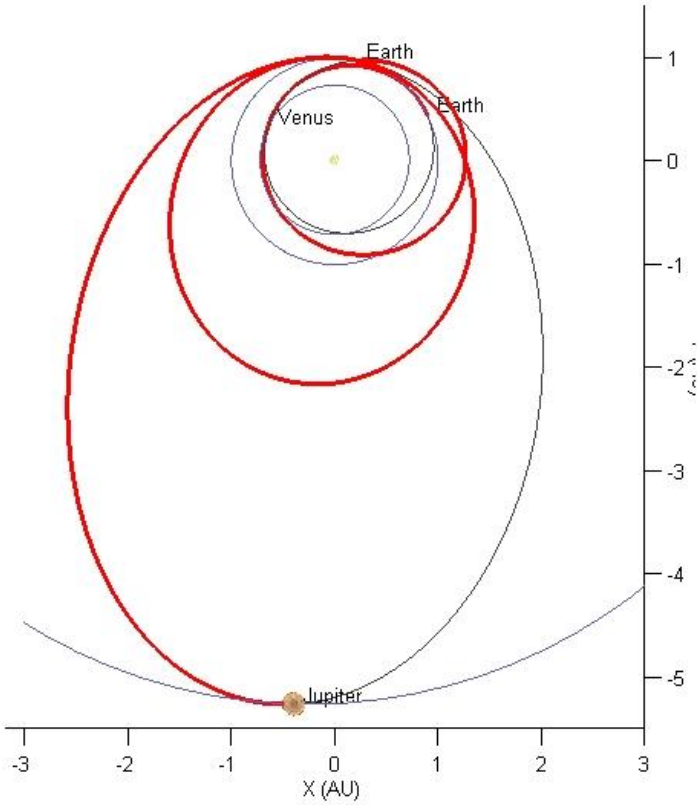


(b) Trajectory generated with Skipping Stone

Figure D.7 Validation of Cassini/Huygens trajectory – trajectories.



(a) Trajectory according to [NASA/JPL 2009b]



(b) Trajectory generated with Skipping Stone

Figure D.8 Validation of Galileo trajectory – trajectories.

E

Numerical Tools

Mathematical analysis can give us a deep insight into the nature of most problems. It allows us to understand and quickly assess the interactions that govern complex and un-intuitive problems we encounter, and allows us to generalize abstractions from one problem, into another, *completely unrelated* problem.

But that is only useful to grasp the *general idea* about the problem. When actual *numbers* are required (especially when dealing with problems from physics), analysis can not generate them *stante pede* as it can with theorems. For many types of problems (often even simple ones), a complex framework of interrelated theorems is required in order to find that specific number.

As the problems get more complicated, so do these frameworks. Today, providing only the set of required theorems is not enough – the number usually will have to be be found by an automatic implementation on a computer, which makes it necessary to also consider the *efficiency* with which the number is calculated. Therefore, the pros and cons of a select few numerical tools will be discussed here.

E.1. Root-finding Algorithms

One standard (and tedious) challenge in mathematics, is to find the *zero-passages* or *roots* of mathematical functions. For simply related quantities this is a fairly trivial task, but for most other problems this is not true – a so-called *root finding algorithm* must be used, which can approximate the root to arbitrary precision.

E.1.1 Householder's methods

The class of so called *Householder methods* are generalized by the equation

$$x_{i+1} = x_i + n\frac{(1/f)^{(n-1)}(x_i)}{(1/f)^{(n)}(x_i)}, \tag{E.1}$$

where the order of the method is indicated by n. An n^{th}-order Householder method requires knowledge on the first n derivatives of $f(x)$, but has a rate of convergence of $n+1$. Since this generalization by Alston Scott Householder (1904-1993) was made long after some of the lower-order methods were discovered, these are generally given different names. The first two orders of Householder's methods are called the *Newton-Raphson method*, and *Halley's method*.

Newton-Raphson Method

The Newton-Raphson root-finding iteration scheme is probably the most widely used root-finding algorithm in the world. It is usually the first algorithm to try, and usually also the last. It is obtained by substituting $n = 1$ in Equation E.1:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

where x_0 is assumed to be reasonably close to the root. Thus, intuitively, it estimates the new approximation by assuming the function $f(x)$ can locally be approximated by a linear function, which is obtained from the current slope $f'(x_i)$. The convergence rate for the Newton-Raphson scheme is usually quadratic [Press et al. 2007].

The main advantages of this method are of course the quick convergence rate versus the simple implementation. The major disadvantages are that the derivative of the function $f(x)$ needs to be known analytically (which can become very tedious for more complicated functions), and that this derivative may become very small or zero, leading to very slow convergence or non-numeric results. Also, if the initial estimate is located far from the root, the algorithm will simply never converge.

The derivative may also be computed numerically when derivatives are too complicated or impossible to find analytically, at the cost of extra FE at each iteration. In this case, due to the associated approximation error, convergence rates usually are slightly lower and reduce to superlinear.

Halley's Method

This method was originally used by Edmond Halley (1656–1742) to predict the next epoch the comet named after him would return to the Sun. This method was invented solely to reduce the amount of calculations involved in such problems. His method generally has cubic convergence rate, since it also uses the second derivative of the function $f(x)$ [Sebah and Gourdon 2001]:

$$x_{i+1} = x_i - \frac{2F(x_i)F'(x_i)}{2\left[F'(x_i)\right]^2 - F(x_i)F''(x_i)},$$

where x_0 is assumed to be reasonably close to the root. This formula can also be obtained by substituting $n = 2$ in Equation E.1, which indeed gives a cubic convergence rate. The advantage for Halley's method is that it requires less iterations due to its cubic convergence rate. As for any higher-order method, this does not automatically mean less calculations, since for many functions the derivatives involve many more terms than the original function. However, one of the main disadvantages encountered with the Newton-Raphson method (locally zero-valued derivatives) does not cause problems in Halley's method (except in isolated cases where *both* derivatives are zero simultaneously). Therefore, Halley's method is considerably more robust than the Newton-Raphson method. Disadvantages are of course that the second derivative needs to be known, and that implementation may be more involved due to this fact.

Also for Halley's method, the first- and second derivatives may be computed numerically when derivatives are too complicated or impossible to find analytically. Naturally, the computational cost of such an implementation will be much higher. In this case, convergence rates are much lower than cubic, and the method reduces to superlinear convergence [Sebah and Gourdon 2001].

E.1.2 Other methods

Regula-Falsi Method

The so-called *Regula-Falsi* method, also known as the *method of false position* or the *inverse linear interpolation method*, is one of the simplest root-finding algorithms. It requires no knowledge about –and makes no demands on– the smoothness of the derivatives of the function F. All it requires is two initial values a_0 and b_0, not necessarily close to the real root c_N, of which the function values $F(a_0)$ and $F(b_0)$ are known to be of opposite sign. The algorithm finds the root c_{i+1} of the line from $F(a_i)$ to $F(b_i)$, and uses c_{i+1} as the new estimate for the root of F. Mathematically [Press et al. 2007, Ford 1995],

$$c_{i+1} = \frac{F(b_i)a_i - F(a_i)b_i}{F(b_i) - F(a_i)}.$$

Then, if the sign of the function value $F(c_{i+1})$ is equal to that of $F(a_i)$, the value of a_{i+1} is set to c_{i+1}. If it is equal to that of $F(b_i)$, then b_{i+1} is set to c_{i+1}, and everything is then repeated until the function value $F(c_N)$ is acceptably close to 0.

The convergence rate of this method can be shown to be superlinear, but reduces to linear for many cases. However, [Ford 1995] showed that with the simple patch

$$c_{i+1} = \frac{\frac{1}{2}F(b_i)a_i - F(a_i)b_i}{\frac{1}{2}F(b_i) - F(a_i)}.$$

the convergence rate is *guaranteed* to be superlinear. [Ford 1995] treated several other patches that give even better superlinear rates, but they are much more complicated and make the method into an opposite to the elegant and simple original method. It seems more obvious

to use a different method in case higher convergence rates are demanded.

The major advantages of this method are the simple implementation, it is derivative-free, and only *one* FE is required in every iteration. The obvious disadvantage is that *two* initial values need to be given, of which is it known *beforehand* what the sign of their function values are. Finding initial values with guaranteed opposite sign is often difficult, which makes this method harder to use in fully automated computations.

Root finding with Chebyshev-polynomials

The Chebyshev-polynomials

$$T_n(x) = \cos(n \arccos(x)) \tag{E.2}$$

have the very useful property that when used for interpolation, they closely approximate the *minimax-polynomial* for any $f(x)$. That is, the maximum deviation between a function $f(x)$ and its interpolating Chebyshev polynomial is close to minimal. This property can be used as a method that reduces the process of finding all roots of $f(x)$ to finding the roots of only the interpolating Chebyshev-polynomial. This is extremely useful in cases where $f(x)$ is known to be smooth and reasonably well behaved, but rather expensive to evaluate. If an approximation to *all* roots of $f(x)$ is required, this method will greatly reduce the number of required evaluations of both $f(x)$ and/or its derivative $f'(x)$.

The first step is to find the coefficients for the Chebyshev polynomial. Since the Chebyshev polynomials are usually defined on the interval $-1 \leq x \leq +1$, the first step is to map the interval $[a, b]$ on which $f(x)$ is defined to $[-1, 1]$. This is accomplished by the change of variable

$$y = \frac{x - \frac{1}{2}(b + a)}{\frac{1}{2}(b - a)}. \tag{E.3}$$

Then, assuming a n^{th}-order polynomial is used for the interpolation, n coefficients c_j need to be calculated:

$$c_j = \frac{2}{n} \sum_{k=0}^{n-1} f\left[\frac{\pi(k + \frac{1}{2}}{n}\right] \cos\left(\frac{\pi j(k + \frac{1}{2}}{n}\right) \tag{E.4}$$

so that the function is approximated by

$$f(y) \approx \left[\sum_{k=0}^{n-1} c_k T_k(y)\right] - \frac{1}{2} c_0. \tag{E.5}$$

This is described in more detail in [Press et al. 2007; Section 5.8]. Now that the Chebyshev polynomial is known, the next step is to find its roots. Note that this is *not* equal to locating the n-roots from a regular Chebyshev-polynomial

$$y = \cos\left(\frac{\pi(k + \frac{1}{2})}{n}\right), \quad k = 1, 2, \ldots n,$$

and re-scaling these to the interval $[a, b]$, since the polynomial is now translated and scaled differently from the canonical polynomial in $[-1, 1]$. The most efficient way to find its n roots is by computing the eigenvalues of the Chebyshev-Frobenius companion matrix A [Boyd 2006]:

$$A = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & \\ 1 & 0 & \frac{1}{2} & 0 & \\ 0 & 1 & 0 & \frac{1}{2} & \cdots \\ & & \vdots & & \\ \frac{-c_1}{2c_n} & \frac{-c_2}{2c_n} & \frac{-c_3}{2c_n} & \cdots & \frac{-c_{n-1}}{2c_n} + \frac{1}{2} \end{bmatrix} \tag{E.6}$$

Finding the eigenvalues is a process requiring $O(n^2)$ time. The total computation time for this root-finding method is then $O(n^2)$ plus the n FE's, so using this method is only recommended for relatively small n. However, the great advantage of this method is that *all* roots of any transcendental function $f(x)$ can be found in one run, taking away the need to find appropriate initial values as is the case with an iterative method. Moreover, this method also does not require derivatives, making it (once implemented) very easy to use. If more accuracy is absolutely required, the method can either be used with a higher value for n (in case derivatives are unavailable) or the roots thus found can serve as accurate initial values for some iterative procedure.

E.2. Numerical Integration and Differentiation

E.2.1 Numerical Differentiation

Many algorithms used in numerical analysis will require knowledge on the derivatives of some function $f(x)$. Unfortunately, for many problems encountered in practice these derivatives are either very hard to find analytically, or simply cannot be found altogether. Also an important factor that cannot be ignored, is that even though explicit expressions for derivatives *can* be found for some function $f(x)$, they are usually complicated, large, and tedious to derive. And when found, they also need to be correctly translated into some computer language. Taken all together, this normally leads to mistakes that are hard to trace down and correct, and may even go by completely unnoticed. For these and many other reasons it is much better to have a way to accurately and efficiently compute derivatives *numerically*, which is the subject of this section.

The definition of the derivative of a one-dimensional function is

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

This definition may be extended to higher dimensions via

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, x_2, \ldots, x_i + h, \ldots, x_N) - f(x_1, x_2, \ldots, x_N)}{h}, \tag{E.7}$$

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_N}, \right). \tag{E.8}$$

This definition allows one to approximate the value of the derivative, simply by taking a *finitely* small value for h. This leads to the collection of *finite-difference* methods.

Finite Differences

Forward Differences The forward differences method make a small step h in the positive x-direction, and compute the value of the fraction Equation E.7:

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x)}{h}. \tag{E.9}$$

This method most closely resembles the exact definition from Equation E.7.

Backward Differences Taking a small step in the *negative* x-direction gives another, different, estimate for the derivative. This gives

$$\frac{df}{dx} \approx \frac{f(x) - f(x-h)}{h}. \tag{E.10}$$

This leads to an approximation of the first derivative that is quite similar to Equation E.9. In fact, when observed from the point $f(x-h)$, the method is exactly the same as Equation E.9 – only the function value at the forward projected point is now known, instead of that on the current point. In practice, the values computed by Equation E.9 or Equation E.10 are indeed quite close, but not the same.

Central Differences A much more accurate approximation to the first derivative may be expected by taking the average of the estimates from both the forward and backward difference methods. Averaging these estimates leads to the *central differences* approximation of the first derivative. Adding Equations E.9 and E.10 and collecting terms gives

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{E.11}$$

By application of Taylor's theorem it can be shown that the error made in the approximation of the derivative by both the forward and backward difference methods is $O(h)$, while the error made with central differences is $O(h^2)$. So as expected, central differences indeed give a much better estimate of the function's derivative at a point x.

As suggested by D'Errico [2009; and references therein], some simple improvements can be made to these "basic" formulas, that give accuracies of $O(h^4)$, $O(h^6)$ or even better. One of these improvements is *odd and even function expansion.* Another improvement is mentioned in [Hoffman 2001; chapter 5.6], which is *extrapolation with error estimates.*

Odd and Even Functions

Consider the derived functions $f_{\text{odd}}(x)$, $f_{\text{even}}(x)$, and their respective Taylor expansions around x_0 :

$$f_{\text{odd}} = \frac{f(x-x_0)-f(-x-x_0)}{2} \tag{E.12}$$

$$= (x-x_0)f^{(1)}(x_0) + \frac{(x-x_0)^3}{6}f^{(3)}(x_0) + \frac{(x-x_0)^5}{120}f^{(5)}(x_0) + \ldots \tag{E.13}$$

$$f_{\text{even}} = \frac{f(-x-x_0)-2f(x_0)+f(x-x_0)}{2} \tag{E.14}$$

$$= \frac{(x-x_0)^2}{2}f^{(2)}(x_0) + \frac{(x-x_0)^4}{24}f^{(4)}(x_0) + \frac{(x-x_0)^6}{720}f^{(6)}(x_0) + \ldots \tag{E.15}$$

Compare these with the Taylor expansion around x_0 of the original function $f(x)$:

$$f(x) = f(x_0)+(x-x_0)f^{(1)}(x_0)+\frac{(x-x_0)^2}{2}f^{(2)}(x_0)+\frac{(x-x_0)^3}{6}f^{(3)}(x_0)+\frac{(x-x_0)^4}{24}f^{(4)}(x_0)+\ldots$$

Note that the function $f_{\text{odd}}(x)$ only has the *odd*-numbered derivatives of $f(x)$ in its Taylor expansion, and $f_{\text{even}}(x)$ only the *even*-numbered derivatives. These functions simplify the process of finding higher-order approximations to derivatives. For example, substituting both $x_1 = x_0 + h$ and $x_2 = x_0 + 2h$ into Equation E.13, adding and rearranging both results, and isolating $f'(x_0)$ gives

$$f'(x_0) = \frac{8f_{\text{odd}}(x_0+h) - f_{\text{odd}}(x_0+2h)}{6h} + h^4 c_1 f^{(5)}(x_0) + \ldots, \tag{E.16}$$

which is clearly of $O(h^4)$.

Extrapolation With Error Estimates

From analysis of the Taylor functions, it can be shown that the error made for different step sizes can be described by the relations

$$\text{Error}_h = Ah^n + \ldots \approx \frac{R^n}{R^n-1}\left(f_{h/R} - f_h\right) \tag{E.17}$$

$$\text{Error}_{h/R} = A\left(\frac{h}{R}\right)^n + \ldots \approx \frac{1}{R^n-1}\left(f_{h/R} - f_h\right) \tag{E.18}$$

$$\tag{E.19}$$

where a subscript $_h$ indicates an n^{th}-order method with step size h. The estimated errors Error_h and $\text{Error}_{h/R}$ can be combined and added to the original estimate of the derivative to give an improved estimate:

$$\frac{df}{dx} \approx f_{h/R} + \frac{1}{R^n-1}\left(f_{h/R} - f_h\right), \tag{E.20}$$

which can be shown to be of $O(h^{n+m})$, where m is the increment in the powers of h in the Taylor expansion. For central differences, this is simply 1, but for the fourth– and sixth–order methods, it is equal to 2.

Accuracy

As a simple test for all the aforementioned methods, derivatives were computed for the simple test function $f(x) = e^x$. Since $d(e^x)/dx = e^x$, testing is greatly facilitated. Table E.1 shows the accuracy achieved versus the number of FE's required to achieve it (once the implementation is optimized). In this simple test, a constant step size of $h = 1 \times 10^{-2}$ was maintained.

Table E.1 Accuracy attained in computing the first two derivatives for the simple test case $f(x) = e^x$, with a constant (non-optimal) step size of $h = 1 \times 10^{-2}$.

Method	Error ($f'(0)$)	Fun. Eval
Fwd. Diffs.	$+5.017\times10^{-3}$	1
Fwd. Diffs. w/ extrap.	-4.180×10^{-6}	2
Bwd. Diffs.	-4.983×10^{-3}	1
Bwd. Diffs. w/ extrap.	-4.155×10^{-6}	2
Centr. Diffs.	1.667×10^{-5}	2
Centr. Diffs. w/ extrap.	-5.180×10^{-12}	4
4^{th}order	-3.333×10^{-10}	4
4^{th}order w/ extrap.	-6.550×10^{-16}	6

In conclusion, the best trade-off of number of FE's versus accuracy seems to be the forward or backward differences, with extrapolation of the error term. Indeed, further testing (with many different trial functions) showed that this method at least equals, but usually outperforms the central differences method, using the same number of FE's. This is because for many functions (especially those containing trigonometric functions), the increment in the powers of the Taylor expansion m is *two* instead of one. Therefore, the performance is *at least* equal to that of the central differences method, but often better.

Naturally, the step size h can not arbitrarily be reduced to extremely small magnitudes – this usually leads to considerable roundoff error. As shown in [Press et al. 2007; Sec. 5.7] an *optimum* step size h can be found that leads to the least amount of roundoff error. This step size is

$$h_{\texttt{opt}} \approx x\sqrt{\epsilon},$$

where ϵ can usually be assumed equal to the machine accuracy (`eps`, the spacing between doubles). Further testing also showed that setting

$$R = 4$$

seems to give the best results for the most functions.

E.2.2 Integration

The term *integration* can generally mean two different things in the literature. One is the calculation of the hypervolume under some function $f(\boldsymbol{x})$, the other is the numerical ap-

proximation of the solution to a (partial) differential equation[1]. Although dissimilar, their numerical solution does require a similar set of operations, hence the naming convention. Both integration methods are briefly elaborated here.

Definite Integrals

These methods attempt to approximate the value of the integral

$$I = \int_a^b f(x)\,dx \tag{E.21}$$

for some function $f(x)$ with an anti-derivative that is non-analytical. The simplest way to calculate I is by employing the *Newton-Cotes formulas*. A somewhat more elaborate method (but many times more efficient), is *Gaußian quadrature*.

Newton-Cotes Formulae A central idea in all quadrature methods is to subdivide the interval $[a,\, b]$ into n smaller segments, and calculating the integral of an p^{th}-order polynomial interpolated through the function. Mathematically,

$$I = \int_a^b f(x)\,dx = \sum_{i=0}^{b-a}\left(\int_a^{a+i} P_p(x)\,dx\right) \tag{E.22}$$

$$= \sum_{j=a}^{b-a} h \sum_{i=1}^{M} H_{M,i} f(j+ih), \tag{E.23}$$

where

$$H_{n,r+1} = \frac{(-1)^{n-r}}{r!(M-r)!}\int_0^M t(t-1)(\ldots)(t-M)\,dt,$$

and $h = (b-a)/M$ – see [Hoffman 2001; chapter 6.3] or [Weisstein 2009] for details. These is the generalized Newton-Cotes formula, which again has numerous names for its lower order derived formulas.

Equation E.23 gives rise to the tableaux listed in Table E.2. As an example, the first line in Table E.2 gives the *trapezoidal rule*,

$$I = \frac{h}{2}\sum_{x_j=a}^{b-a}\sum_{i=1}^{n}\left(f(x_j) + f(x_j+ih)\right), \tag{E.24}$$

the second line *Simpson's rule*,

$$I = \frac{h}{3}\sum_{x_j=a}^{b-a}\sum_{i=1}^{n}\left(f(x_j) + 4f(x_j+\frac{ih}{2}) + f\left(x_j+ih\right)\right), \tag{E.25}$$

[1]The first of these is more often referred to as *quadrature*, to distinguish it from the other.

Table E.2 Tableau for the n^{th}-order Newton-Cotes integration method. Adopted from [Weisstein 2009].

n \ r	1	2	3	4
1	$\frac{1}{2}$	$\frac{1}{2}$		
2	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{1}{3}$	
3	$\frac{3}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	$\frac{3}{8}$

etc. It can be shown that the approximation error (per interval) is of $O(h^{p+1})$ if p is even, and $O(h^{p+2})$ if p is odd. Therefore, the approximation per interval becomes better for a finer interval subdivision (since $h \downarrow$ when $n \uparrow$), and improves globally if the order of the method is increased (larger p).

Gaußian Quadrature The Newton-Cotes formulas all evaluate the function $f(x)$ at equally spaced points. Thus, for an p^{th}-order method, p points are pre-specified, and a $p+1$-dimensional polynomial can be used for the interpolation. This can be written as

$$I = \int_a^b f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i),$$

where the constants w_i follow from the Newton-Cotes formulas, and the x_i are equally spaced. However, when allowing the locations of the x_i to be *free*, p additional degrees of freedom are introduced. Therefore, a $(2p-1)^{\text{th}}$-degree polynomial can be fit through the data, increasing the accuracy of the approximation to the integral significantly for the same computational cost.

The points t_i where the function is evaluated, and the corresponding weights w_i, can be optimized such that the $2n-1$-degree polynomial is constructed of all polynomials of degree smaller than $2n-1$ that reproduce the exact value of the integral. When solved generally, the optimal points t_i are found to lie symmetrically around the mid-point of the interval of integration. They are the i^{th} root of the (normalized) polynomial $P_n(t_i)$, and the associated weights w_i are

$$\frac{2}{(1-t_i^2)(P_n'(t_i))^2},$$

where $P_n(x)$ is the n^{th}-order Legendre polynomial. The first few values of t_i and w_i are listed in Table E.3.

In summary,

$$I = \int_{-1}^{+1} f(x)\,dx$$

can thus be approximated by

$$I \approx \sum_{i=0}^{n} \left(w_{-i} f(t_{-i}) + w_{+i} f(t_{+i})\right).$$

Table E.3 Tableaux for n-point Gaußian quadrature methods. The order of the method is $2n-1$. From [Hoffman 2001]

n	w_i	t_i
2	1	$\pm\sqrt{1/3}$
3	$^8/_9$	0
	$^5/_9$	$\pm\sqrt{3/5}$
4	$\pm\sqrt{3\mp 2\sqrt{6/5}}$	$\frac{18\pm\sqrt{(30)}}{36}$
5	0	$^{128}/_{225}$
	$\pm^1/_3\sqrt{5-2\sqrt{10/7}}$	$\frac{322\pm 13\sqrt{70}}{900}$

which has accuracy of $O(2n-1)$. To shift this integral to the proper interval $a \le x \le b$, the following coordinate transformation must be applied [Hoffman 2001; Example 6.6 and further]:

$$I = \int_a^b f(x)\,dx \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}t_i + \frac{a+b}{2}\right). \tag{E.26}$$

The correct values for $w_{\pm i}$ and $t_{\pm i}$ are given in the tableaux in Table E.3.

Integrating Chebyshev polynomials A very convenient way to find the integral of a function is by integrating its Chebyshev polynomial. Since this is a regular polynomial, its anti-derivative is explicitly known so an integration is straightforward. Suppose some function $f(x)$ is interpolated by its Chebyshev polynomial over an interval $[a, b]$. The coefficients c_j are then given by Equation E.4, and the value of the function can be found with Equation E.5, which is repeated here for clarity:

$$f(x) \approx \left[\sum_{k=0}^{n-1} c_k T_k(x)\right] - \frac{1}{2}c_0. \tag{E.27}$$

For an $(n+1)^{\text{th}}$-order Chebyshev polynomial, the following recursion relation holds:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1} &= 2xT_n(x) - T_{n-1}(x) \end{aligned} \tag{E.28}$$

The integral of $f(x)$ can then be written as

$$\int f(x)dx \approx \int \left(\left[\sum_{k=0}^{n-1} c_k T_k(x)\right] - \frac{1}{2}c_0\right) dx = \left[\sum_{k=0}^{n-1} c_k \int T_k(x)dx\right] - \frac{1}{2}xc_0. \tag{E.29}$$

It can be shown with the recursion relation from Equation E.28 that the integral of a Chebyshev polynomial $T_n(x)$ can be written as

$$\int T_n(x)dx = \frac{nT_{n+1}(x)}{n^2-1} - \frac{xT_n(x)}{n-1}, \quad n \neq 1,$$

and for $n = 1$ the expression

$$\int T_1(x) = \frac{1}{2}x^2$$

follows trivially from Equation E.28. These expressions are exceptionally well-suited to integrate the function $f(x)$ over the interval $[-1, 1]$, because

$$\begin{aligned} T_n(+1) &= 1 \\ T_n(-1) &= (-1)^{n+1} \end{aligned}$$

for all n. With this, an approximation to the integral of the function over the interval $[-1, 1]$ is then

$$\int_{-1}^{+1} f(x)dx \approx \sum_{\substack{k=0 \\ k\neq 1}}^{n-1} \left[c_k \left(\frac{k}{k^2-1} - \frac{1}{k-1} \right) \right] - \sum_{\substack{k=0 \\ k\neq 1}}^{n-1} \left[(-1)^k c_k \left(\frac{1}{k-1} - \frac{k}{k^2-1} \right) \right] + \frac{c_2}{2}. \quad \text{(E.30)}$$

Naturally, the coordinate transformation from Equation E.3 can be used to map any interval $[a, b]$ to this canonical interval $[-1, 1]$. Note that in that case, the integral given above should be corrected when mapping it back to $[a, b]$:

$$\int_a^b f(x)dx = \frac{b-a}{2} \cdot \int_{-1}^{+1} f(y)dy.$$

Note that all the terms in both sums in Equation E.30 are *equal*; the only difference is the alternating minus sign in the second sum. This fact can improve the efficiency of an implementation of such a quadrature method, since all the terms only need to be calculated once per iteration. Since the terms are also *independent* from the function $f(x)$ and its variable x (aside from the coefficients c_j), it is is even more efficient to *pre-compute* a large number of terms so that these need not be re-computed every time a function is integrated.

Differential Equations

An Ordinary Differential Equation (ODE) is an equation of the form

$$\begin{aligned} \dot{\boldsymbol{y}} &= \boldsymbol{f}(t, \boldsymbol{y}, \dot{\boldsymbol{y}}), \\ \boldsymbol{y}(t_0) &= y_0, \end{aligned} \quad \text{(E.31)}$$

in other words, the only thing that is known about some unknown function $\boldsymbol{y}(t)$ is its relationship with its own derivative. It has been shown that there is no *general* method to explicitly write the function $\boldsymbol{y}(t)$ as some combination of elementary functions. Moreover, for many functions $\boldsymbol{f}$ it holds that this is *impossible*. Most ODE's encountered in practice can indeed not be solved analytically, which necessitates the use of a numerical method to solve ODE's. This section discusses what integration techniques are best suited for the purposes of this theses.

Numerical Integrators Methods that solve ODE's work by evaluating the function $\boldsymbol{f}$ at small time steps h away from the initial value, and adding these result to the current estimate: $\boldsymbol{y}(t_0+h) \approx h \cdot \boldsymbol{y}(t_0)$. These steps are then repeated until the desired final time t_{end} is reached. The accuracy of this solution depends on a great number of factors, most notably the time step taken (h) and the "order" of the method (q).

There are many methods to integrate (partial) differential equations. The most prominent categories are *single-step* integrators, *multi-step* integrators, and integrators that use *extrapolation*. The most notable single-step and multi-step methods are:

- Runge-Kutta (single step)
- Runge-Kutta-Nyström (single step)
- Adams-Bashforth (multi-step)
- Gauß-Jackson (multi-step)

A detailed description of integration methods is too lengthy to include here. The interested reader is referred to [Gill and Montenbrück 2000; chapter 4]. In [Gill and Montenbrück 2000; section 4.4], the authors conclude that in cases where the differential equation is independent of the first-order derivative $\dot{y}$, the class of higher-order RKN methods are most efficient in terms of accuracy, number of FE's and computation time required by the method itself. Since in this thesis, the integration of the equations of motion of bodies under their mutual Newtonian gravitational attraction is of interest, this is indeed the case. Therefore, this class of integrators is briefly elaborated here.

The class of RKN methods differ from RK methods in the sense that the RKN methods work directly on a system of second-order differential equations

$$\begin{aligned} \ddot{\boldsymbol{y}} &= \boldsymbol{f}(t, \boldsymbol{y}), \\ \boldsymbol{y}(t_0) &= y_0, \qquad \dot{\boldsymbol{y}}(t_0) = \dot{y}_0, \end{aligned} \tag{E.32}$$

instead of the more common equations in Equation E.32. The advantage of doing so is that for this special case, equations for an integrator can be derived, which requires less FE's than a classical RK method of the same order. The equations for a generic q^{th}-order RKN-integrator are:

$$\boldsymbol{f}_i = \boldsymbol{f}\left(t_n + c_i h_n,\, \boldsymbol{y}_n + c_i h_n \dot{\boldsymbol{y}}_n + h_n^2 \sum_{j=1}^{i-1} A_{ij} \boldsymbol{f}_j\right) \tag{E.33}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h_n \dot{\boldsymbol{y}}_n + h_n^2 \sum_{i=1}^{q} b_i \boldsymbol{f}_i \tag{E.34}$$

$$\dot{\boldsymbol{y}}_{n+1} = \dot{\boldsymbol{y}}_n + h_n \sum_{i=1}^{q} \hat{b}_i \boldsymbol{f}_i \tag{E.35}$$

Executing these steps for all epochs $t \in [t_0,\, t_0 + h_n,\, \ldots,\, t_{\text{end}}]$ will then yield estimates for both $\boldsymbol{y}$ and $\dot{\boldsymbol{y}}$ at all of these epochs. While this procedure with given constants $\boldsymbol{c}$, $\boldsymbol{b}$, $\hat{\boldsymbol{b}}$ and matrix A is already suited as an integrator, it can be shown that the *local truncation error* e made at each step n strongly depends on the local step size h_n, indicated by writing $e(h_n)$. Therefore, it would be beneficial to adjust the step size h_n at each iteration n such that it gives the least overall computational effort, while guaranteeing that the error $e(h_n)$ always stays below an acceptable limit at each step.

This is accomplished by *embedding* a method of order $q-2$ into the procedure above. By using the combination of orders $q(q-2)$ it can be shown that the local truncation error made is $O(h^{q+1})$, e.g., one order of magnitude higher than q. The actual step is then made by the q^{th}-order method, while the $(q-2)^{\text{th}}$-order method serves to make an estimate of the error e[2]. The error $e(h_n)$ can then be used to adjust the next step size h_{n+1}. In mathematical terms:

$$e(h_n) = \lambda h_n^2 \cdot \max\left(\left|\hat{\boldsymbol{y}} - \boldsymbol{y}\right|,\, \left|\dot{\hat{\boldsymbol{y}}} - \dot{\boldsymbol{y}}\right|\right),$$

where λ is a free design parameter, taken equal to $\lambda = 0.9$ for many high-order methods. The error can be used to construct a new h_{n+1}:

$$h_{n+1} = h_n \left(\frac{\texttt{tol}}{e(h_n)}\right)^{1/q}. \tag{E.36}$$

In the above, $\hat{\boldsymbol{y}}$ denotes the solution obtained with the q^{th}-order method and $\boldsymbol{y}$ that of the $(q-2)^{\text{th}}$-order method, and $0 < \texttt{tol} \ll 1$ is a user-defined demand on the minimum accuracy per step. The embedded scheme can be formulated as follows:

$$\boldsymbol{f}_i = \boldsymbol{f}\left(t_n + c_i h_n,\, \boldsymbol{y}_n + c_i h_n \dot{\boldsymbol{y}}_n + h_n^2 \sum_{j=1}^{i-1} A_{ij} \boldsymbol{g}_j\right),$$

$$\begin{array}{rclrcl} \hat{\boldsymbol{y}}_{n+1} &=& h_n \hat{\Phi} & \boldsymbol{y}_{n+1} &=& h_n \Phi \\ \dot{\hat{\boldsymbol{y}}}_{n+1} &=& h_n \dot{\hat{\Phi}} & \dot{\boldsymbol{y}}_{n+1} &=& h_n \dot{\Phi} \end{array} \tag{E.37}$$

where

$$\begin{array}{rclrcl} \hat{\Phi} &=& \hat{\dot{\boldsymbol{y}}}_n + h_n \sum_{i=1}^{q} \hat{b}_i \boldsymbol{f}_i & \Phi &=& \dot{\boldsymbol{y}}_n + h_n \sum_{i=1}^{q} b_i \boldsymbol{f}_i \\ \dot{\hat{\Phi}} &=& \sum_{i=1}^{q} \dot{\hat{b}}_i \boldsymbol{f}_i & \dot{\Phi} &=& \sum_{i=1}^{q} \dot{b}_i \boldsymbol{f}_i \end{array}$$

The coefficients $\boldsymbol{c}$, $\boldsymbol{b}$, $\hat{\boldsymbol{b}}$ and matrix A for higher-order methods are quite tedious to derive, especially for higher-order methods. Coefficients for an RKN8(6) method can be found in [Papakostas and Tsitouras 1999], and those for an RKN7(6) integrator are listed in [Gill and Montenbrück 2000; table 4.2]. Coefficients for an RKN12(10) method can be requested from [Dormand et al. 1991].

[2]The complete method with two embedded orders is usually denoted RKN$q(q-2)$, as in RKN8(6) or RKN12(10).

Trade-off The standard `ODE45()`-function in MATLAB® is an RK4(5) method. It should have accuracy of $O(h^4)$ per step. This would mean an error of $\sim 1 \times 10^{-4}$ AU ≈ 1500 km per step (in case $h = 0.1$), which is certainly not acceptable for longer integration times. The RKN8(6) has an accuracy of $O(h^9)$, giving $e(0.1) \approx 150$ m per step, at the cost of 12 FE's per step. This already seems quite acceptable; only on very long time intervals would this accumulate to unacceptable errors, but it is quite acceptable on the intervals used in the optimizations for this research.

The accuracy would be even better with an RKN12(10) method, which has an accuracy of $O(h^{13})$, at 17 FE's per step. This would mean $e(0.1) \approx 1.5$ cm per step. Of course, this amount of accuracy is far more than what is justifiable for the DE405 force model; since this model is already a simplification, it would be exaggerated to try and achieve accuracies at cm-level. Therefore it seems best to use the RKN8(6) integrator for all direct integrations.

F

Coordinate Transformations

F.1. Solving Kepler's Equation

Arguably the best known equation in all disciplines dealing with bodies in orbit, is *Kepler's equation*. It is the relation between time and the position of the body in its orbit, which is usually expressed as

$$\begin{aligned} M &= \sqrt{\tfrac{\mu}{a^3}}(t-t_0) = E - e\sin E \qquad \text{if } e<1 \\ N &= \sqrt{\tfrac{\mu}{-a^3}}(t-t_0) = e\sinh F - F \quad \text{if } e>1 \end{aligned} \tag{F.1}$$

with M the mean anomaly, E the eccentric anomaly, N and F the hyperbolic counterparts of these, t the current time, t_0 the time of last pericenter passage, a (or $-a$) the orbit's semi major axis, μ the central body's standard gravitational parameter, and e the orbit's eccentricity.

Since the eccentric anomaly is related to the true anomaly via

$$\tan\frac{\theta}{2} = \begin{cases} \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} & \text{if } e<1 \\ \sqrt{\frac{e+1}{e-1}}\tanh\frac{F}{2} & \text{if } e>1 \end{cases} \tag{F.2}$$

determining the mean anomaly when θ, a and e are given (and thus also the instant in time this configuration will occur) is a trivial exercise. However, the inverse operation – determining θ when the times t, t_0 and the other orbital parameters are known – is much less trivial.

Fortunately, many methods have been developed over the years to solve this problem. Given that Equation F.1 is transcendental, all of these methods must determine the value for E

iteratively. As shown by Meeus [2005; pages 199–205], it is very effective to use a modified-Newton-Raphson approach. For the elliptic case this takes the following form:

$$\begin{aligned} E_0 &= M \\ \text{correction} &= \frac{M + e\sin E_i - E_i}{1 - e\cos E_i} \end{aligned}$$

```
if |correction| > 0.5
        correction = sign(correction) * 0.5
end if
```

$$E_{i+1} = E_i - \text{correction}.$$

This modified-Newton-Raphson approach usually requires a few more iterations than many simpler schemes, but such schemes often only work for low eccentricities ($0 < e <\sim 0.5$), or have trouble converging for some values of M. In case the eccentricity is higher ($\sim 0.5 < e < \infty$), these simpler methods all become numerically unstable and require many more iterations than this modified Newton-Raphson approach to converge.

The additional restriction imposed on the absolute value of the correction at every iteration prevents the intermediate solutions from oscillating with an unreasonable amplitude around the correct solution at every iteration. Imposing this restriction on the correction factor can be justified by noting that $-1 < e\sin E < 1$. The value of 0.5 (the average of 0 and 1), was found to work best in a numerical study in [Meeus 2005; pages 199–205].

Usually, assuming a starting value of E equal to $E_0 = M$ works quite satisfactory. However, better initial estimates for E will result in faster convergence of the Newton-Raphson scheme. Good values are derived in [Battin 1999] for example; they read

$$\begin{aligned} E_0 &= M + \tfrac{e\sin M}{1-\sin(M+e)+\sin M}, && \text{if } e < 1 \\ H_0 &= \tfrac{N^2}{e(e-1)\sinh\frac{N}{e-1} - N} && \text{if } e > 1 \end{aligned} \tag{F.3}$$

These follow from performing a single Regula-Falsi iteration on Kepler's equation in both forms. However, after a bit of research, it was found to be far more efficient to solve the cubic part of a series approximation to Kepler's equation, as shown in [Mikkola 1987]. This results in the following form for the initial value:

$$\begin{aligned}
\alpha &= \frac{1-e}{4e+\frac{1}{2}} \\
\beta &= \frac{M}{2(4e+\frac{1}{2})} \\
z &= \left(\beta \pm \sqrt{\beta^2+\alpha^3}\right)^{1/3} \\
s_0 &= z - \frac{\alpha}{z} \\
s &= s_0 - \frac{0.078 s_0^5}{1+e} \\
E_0 &= M + e(3s - 4s^3)
\end{aligned} \tag{F.4}$$

where the sign of the square root, is taken to be the sign of β, and $-\pi \leq M < \pi$. This process leads to an initial guess with a *maximum* error of $|E_0 - E| < 0.002$, for all values of $e < 1$, $M = \pi$. Opposed to $|E_0 - E| <\sim 0.4$ as was found numerically by using Equation F.3 from [Battin 1999], this significantly improves the convergence rate of the iterative scheme. An accuracy of better than 10^{-10} is achieved for most values of M and e, after only *one* iteration of the Newton-Raphson scheme.

For the hyperbolic form of Kepler's equation, the Newton-Raphson scheme must be changed into

$$F_{i+1} = F_i - \frac{N - e\sinh F_i + F_i}{e\cosh F_i - 1}.$$

The argument for imposing a limitation on the correction factor no longer holds in this case, since $e\sinh F$ is unconstrained. So in this case the correction term must be left as is.

[Mikkola 1987] also provides initial conditions for the hyperbolic case:

$$\begin{aligned}
\alpha &= \frac{e-1}{4e+\frac{1}{2}} \\
\beta &= \frac{N}{2(4e+\frac{1}{2})} \\
z &= \left(\beta \pm \sqrt{\beta^2+\alpha^3}\right)^{1/3} \\
s_0 &= z - \frac{\alpha}{z} \\
s &= s_0 + \frac{0.071 s_0^5}{((1+0.45s_0^2)(1+4s_0^2)e)} \\
F_0 &= 3\ln(s + \sqrt{1+s^2})
\end{aligned} \tag{F.5}$$

again with the sign of the root equal to that of β. This method leads to accuracies very close to those of the elliptic case – for most values of N and e, *one* iteration of the modified Newton-Raphson scheme suffices to obtain accuracies better than 10^{-8}.

F.2. Kepler Elements to Cartesian coordinates

The transformation from given Keplerian elements $[a,\, e,\, i,\, \omega,\, \Omega,\, \theta]$ to cartesian coordinates $[x,\, y,\, z,\, \dot{x},\, \dot{y},\, \dot{z}]$ may be carried out as follows (see [Wakker 2007; pages 11-19 through 11-24] for a more complete derivation). Let the 2-dimensional intermediate reference frame ξ, η define coordinates in the plane of the orbit. Given the orbital elements a, e and θ, this ξ and η can be written as

$$\begin{aligned} \xi &= r\cos\theta, \qquad \text{(F.6)} \\ \eta &= r\sin\theta, \qquad \text{(F.7)} \end{aligned}$$

where

$$r = \frac{a(1-e^2)}{1+e\cos\theta}.$$

Then, applying the rules of spherical trigonometry, the Cartesian coordinates x, y, z are found to be

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} l_1 & l_2 \\ m_1 & m_2 \\ n_1 & n_2 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix}, \qquad \text{(F.8)}$$

in which

$$\begin{aligned} l_1 &= \cos\omega\cos\Omega - \sin\omega\sin\Omega\cos i \\ m_1 &= \cos\omega\sin\Omega - \sin\omega\cos\Omega\cos i \\ n_1 &= \sin\omega\sin i \\ l_2 &= -\sin\omega\cos\Omega - \cos\omega\sin\Omega\cos i \\ m_2 &= -\sin\omega\sin\Omega + \cos\omega\cos\Omega\cos i \\ n_2 &= \cos\omega\sin i. \end{aligned}$$

The velocities $\dot{x}$, $\dot{y}$ and $\dot{z}$ may be found from taking the derivative with respect to time from Equations F.6:

$$\begin{aligned} \dot{\xi} &= \dot{r}\cos\theta - r\dot{\theta}\sin\theta \\ \dot{\eta} &= \dot{r}\sin\theta + r\dot{\theta}\cos\theta. \end{aligned}$$

Substituting

$$\begin{aligned} r\dot{\theta} &= \frac{\mu}{H}(1+e\cos\theta) \\ \dot{r} &= \frac{\mu}{H}e\sin\theta \end{aligned}$$

and the previous relations l_1 through n_2 found from spherical trigonometry, results in

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \frac{\mu}{H} \begin{bmatrix} -l_1 & l_2 \\ -m_1 & m_2 \\ -n_1 & n_2 \end{bmatrix} \begin{bmatrix} \sin\theta \\ e + \cos\theta \end{bmatrix}. \tag{F.9}$$

F.3. Cartesian Coordinates to Kepler Elements

The transformation from given Cartesian coordinates $[x,\ y,\ z,\ \dot{x},\ \dot{y},\ \dot{z}]$ to Keplerian elements $[a,\ e,\ i,\ \omega,\ \Omega,\ \theta]$ may be carried out as follows (see [Wakker 2007; pages 11-26 through 11-27] for a more complete derivation). Define

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2}, && \text{(F.10)} \\ V &= \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}, && \text{(F.11)} \\ \boldsymbol{H} &= \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} = \begin{bmatrix} y\dot{z} - z\dot{y} \\ x\dot{z} - z\dot{x} \\ x\dot{y} - y\dot{x} \end{bmatrix}, && \text{(F.12)} \\ H &= \sqrt{H_x^2 + H_y^2 + H_z^2}, && \text{(F.13)} \end{aligned}$$

where $\boldsymbol{H}$ is the associated angular momentum. Then,

$$\begin{aligned} a &= \frac{r}{2 - rV^2/\mu} \\ i &= \arccos\left(\frac{H_z}{H}\right) \\ \Omega &= \texttt{atan2}(\sin\Omega,\ \cos\Omega), \\ \omega &= \texttt{atan2}(\sin(\omega+\theta),\ \cos(\omega+\theta)) - \theta, \end{aligned}$$

and θ and e follow from

$$\left.\begin{aligned} e\cos E &= 1 - \tfrac{r}{a} \\ e\sin E &= \sqrt{\tfrac{1}{\mu a}}(\boldsymbol{r}\cdot\boldsymbol{V}) \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} E &= \texttt{atan2}(e\sin E,\ e\cos E), \\ e &= \tfrac{1-r/a}{\cos E} = \tfrac{\sqrt{1/a\mu}(\boldsymbol{r}\cdot\boldsymbol{V})}{\sin E} \end{aligned}\right.$$

and Equations F.2. In these equations,

$$\sin\Omega = \frac{H_x}{H\sin i}, \qquad \cos\Omega = \frac{H_y}{H\sin i}, \tag{F.14}$$

$$\sin(\theta+\omega) = \frac{x}{r}\cos\Omega + \frac{y}{r}\sin\Omega, \qquad \sin(\omega+\theta) = \frac{z}{r\sin i}. \tag{F.15}$$

F.4. Keplerian Elements or Cartesian Coordinates to Parametric Representation

The general parametric representation looks like

$$\boldsymbol{\chi} = \begin{cases} \boldsymbol{C} + a\boldsymbol{u}\cos E + b\boldsymbol{v}\sin E & \text{if } e < 1 \\ \boldsymbol{C} - a\boldsymbol{u}\cosh F + b\boldsymbol{v}\sinh F & \text{if } e > 1 \end{cases} \tag{F.16}$$

In these equations, a and b are the semi-major and -minor axes, respectively, E the eccentric anomaly, F the hyperbolic eccentric anomaly, and $\boldsymbol{u}$ and $\boldsymbol{v}$ are unit vectors, $\boldsymbol{u}$ from C to pericenter, and $\boldsymbol{v}$ perpendicular to $\boldsymbol{u}$ towards the point b in Figure F.1 in the positive θ direction.

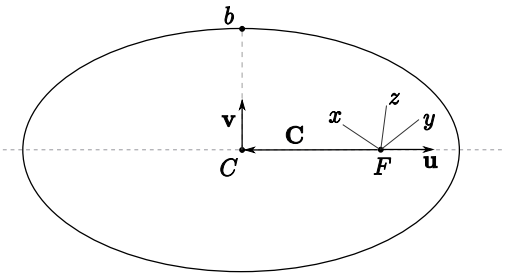


Figure F.1 Parametric representation of an ellipse.

For a representation in these coordinates, both the unit vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, the location of the center $\boldsymbol{C}$ and the eccentric anomaly E (or hyperbolic anomaly F) need to be determined. This can most easily be accomplished by using both the previous two coordinate transformations.

Referring to Figure F.1, it can be concluded that the unit vector $\boldsymbol{u}$ towards pericenter can be found by setting the true anomaly $\theta = 0$. The resulting coordinates must then be converted to Cartesian coordinates using the conversions outlined in section F.2, and the unit of the resulting vector is the unit vector $\boldsymbol{u}$.

Note that the magnitude of the vector that follows from the coordinate transformation is simply the pericenter distance $r_p = a(1 - e)$. Note also that it does not matter that this vector's origin lies at F rather than C – taking either origin would give the same vector of course. Also note that if the complete previous coordinate transformation from Keplerian elements to Cartesian coordinates would be used, the output would also include the velocity at pericenter $\boldsymbol{V}_{r_p}$. This result is now not used.

Since the sum of the lengths of the two lines drawn from the two foci in an ellipse to a point on the ellipse is always $2a$, it can be concluded that the line from F to b has length a. Because of this, it can be found that the true anomaly θ_b at the instant of passage through b would be

$$\theta_b = \pi - \arccos(e),$$

since the length of the line segment CF has length ae. The vector $\boldsymbol{v}$ can then be found from another coordinate conversion with this value for θ. However, this approach would only be valid in case the orbit is an ellipse.

A more general approach is to re-use the velocity at pericenter $\boldsymbol{V}_{r_p}$, which was not used in the previous step. This velocity gives the orbit's angular momentum vector:

$$\boldsymbol{r}_p \times \boldsymbol{V}_{r_p} = \boldsymbol{H}.$$

Taking the unit of the cross-product of $\boldsymbol{H}$ and $\boldsymbol{u}$ gives the vector $\boldsymbol{v}$:

$$\boldsymbol{v} = \frac{\boldsymbol{H} \times \boldsymbol{u}}{|\boldsymbol{H}|}, \tag{F.17}$$

which is both valid for hyperbolic trajectories and elliptic orbits. The location of the center C is easily found from the known vector $\boldsymbol{u}$:

$$\boldsymbol{C} = -ae\boldsymbol{u}, \tag{F.18}$$

and as always,

$$b = \begin{cases} a\sqrt{1-e^2} & \text{if } e < 1 \\ -a\sqrt{e^2-1} & \text{if } e > 1 \end{cases} \tag{F.19}$$

Finally, the (hyperbolic) eccentric anomaly can be found from the conventional conversions already given in Equation F.2.

Raw Data

G.1. High-thrust, First-order, all 146

```
======================
Sun
======================
GA      : 13.478 km/s
DE      : 13.375 km/s
PSO     : 13.526 km/s
ASA     : 12.699 km/s
GODLIKE: 12.716 km/s

======================
Jupiter
======================
GA      :  8.3888 km/s
DE      :  6.3925 km/s
PSO     :  6.3925 km/s
ASA     :  6.9892 km/s
GODLIKE:  6.3925 km/s

======================
Sun - Mercury
======================
GA      :  31.516 km/s
DE      :  30.609 km/s
PSO     :  28.774 km/s
ASA     :  26.479 km/s
GODLIKE:  25.875 km/s

======================
Sun - Venus
======================
GA      :  21.831 km/s
DE      :  23.417 km/s
PSO     :  21.966 km/s
ASA     :  21.268 km/s
GODLIKE:  20.656 km/s

======================
Sun - Earth
======================
GA      :  20.878 km/s
DE      :  21.391 km/s
PSO     :  20.716 km/s
ASA     :  20.800 km/s
GODLIKE:  28.678 km/s

======================
Sun - Mars
======================
GA      :  21.715 km/s
DE      :  21.728 km/s
PSO     :  23.280 km/s
ASA     :  21.715 km/s
GODLIKE:  21.728 km/s

======================
Sun - Jupiter
======================
GA      :  9.3112 km/s
DE      :  13.066 km/s
PSO     :  13.121 km/s
ASA     :  13.480 km/s
GODLIKE:  9.1247 km/s

======================
Sun - Saturn
======================
GA      :  18.466 km/s
DE      :  20.816 km/s
PSO     :  21.014 km/s
ASA     :  19.894 km/s
GODLIKE:  19.819 km/s

======================
Sun - Uranus
======================
GA      :  16.520 km/s
DE      :  16.424 km/s
PSO     :  21.431 km/s
ASA     :  22.835 km/s
GODLIKE:  21.045 km/s

======================
Mercury - Sun
======================
GA      :  21.188 km/s
DE      :  22.012 km/s
PSO     :  20.790 km/s
ASA     :  26.556 km/s
GODLIKE:  20.274 km/s

======================
Mercury - Jupiter
======================
GA      :  26.940 km/s
DE      :  23.414 km/s
PSO     :  34.903 km/s
ASA     :  29.620 km/s
GODLIKE:  19.544 km/s

======================
Mercury - Saturn
======================
GA      :  46.237 km/s
DE      :  43.935 km/s
PSO     :  44.274 km/s
ASA     :  42.249 km/s
GODLIKE:  42.248 km/s

======================
Venus - Sun
======================
GA      :  17.568 km/s
DE      :  18.978 km/s
PSO     :  16.590 km/s
ASA     :  17.949 km/s
GODLIKE:  16.583 km/s

======================
Venus - Jupiter
======================
GA      :  13.323 km/s
DE      :  13.008 km/s
PSO     :  13.451 km/s
ASA     :  15.733 km/s
GODLIKE:  12.905 km/s

======================
Venus - Saturn
======================
GA      :  23.230 km/s
DE      :  23.383 km/s
PSO     :  20.854 km/s
ASA     :  26.707 km/s
GODLIKE:  38.628 km/s

======================
Earth - Sun
======================
GA      :  20.686 km/s
DE      :  19.835 km/s
PSO     :  22.549 km/s
ASA     :  19.935 km/s
GODLIKE:  22.389 km/s

======================
Earth - Jupiter
======================
GA      :  12.463 km/s
DE      :  13.406 km/s
PSO     :  14.374 km/s
ASA     :  12.577 km/s
GODLIKE:  13.680 km/s

======================
Earth - Saturn
======================
GA      :  21.218 km/s
DE      :  21.982 km/s
PSO     :  36.074 km/s
ASA     :  28.973 km/s
GODLIKE:  22.762 km/s

======================
Mars - Jupiter
======================
GA      :  25.443 km/s
DE      :  26.090 km/s
PSO     :  25.538 km/s
ASA     :  25.443 km/s
GODLIKE:  25.538 km/s

======================
Mars - Sun
======================
GA      :  18.695 km/s
DE      :  19.329 km/s
PSO     :  19.326 km/s
ASA     :  19.577 km/s
GODLIKE:  18.796 km/s

======================
Mars - Saturn
======================
GA      :  30.750 km/s
DE      :  30.757 km/s
PSO     :  36.582 km/s
ASA     :  32.549 km/s
GODLIKE:  30.750 km/s

======================
Jupiter - Sun
======================
GA      :  14.575 km/s
DE      :  17.186 km/s
PSO     :  15.217 km/s
ASA     :  15.306 km/s
GODLIKE:  18.653 km/s

======================
Jupiter - Saturn
======================
GA      :  19.292 km/s
DE      :  21.340 km/s
PSO     :  26.962 km/s
ASA     :  19.835 km/s
GODLIKE:  22.527 km/s

======================
Jupiter - Uranus
======================
GA      :  10.162 km/s
DE      :  10.088 km/s
PSO     :  10.204 km/s
ASA     :  16.441 km/s
GODLIKE:  10.088 km/s

======================
Sun - Mercury - Venus
======================
GA      :  NaN    km/s
DE      :  46.956 km/s
PSO     :  59.339 km/s
ASA     :  48.316 km/s
GODLIKE:  49.009 km/s

======================
Sun - Mercury - Earth
======================
GA      :  197.40 km/s
DE      :  NaN    km/s
PSO     :  NaN    km/s

======================
Sun - Mercury - Mars
======================
GA      :  53.637 km/s
DE      :  56.538 km/s
PSO     :  80.350 km/s

======================
Sun - Mercury - Jupiter
======================
GA      :  NaN    km/s
DE      :  NaN    km/s
PSO     :  23.471 km/s

======================
Sun - Mercury - Saturn
======================
GA      :  NaN    km/s
DE      :  NaN    km/s
PSO     :  68.112 km/s

======================
Sun - Mercury - Uranus
======================
GA      :  101.14 km/s
DE      :  67.064 km/s
PSO     :  NaN    km/s
```

```
ASA    :  47.326 km/s
GODLIKE:  NaN    km/s

ASA    :  NaN    km/s
GODLIKE:  76.849 km/s

ASA    :  254.25 km/s
GODLIKE:

ASA    :  NaN    km/s
GODLIKE:  NaN    km/s

ASA    :  NaN    km/s
GODLIKE:  NaN    km/s


======================
Sun - Venus - Venus
======================
GA     :  NaN    km/s
DE     :  NaN    km/s
PSO    :
ASA    :
GODLIKE:

======================
Sun - Venus - Earth
======================
GA     :  NaN    km/s
DE     :  NaN    km/s
PSO    :  58.519 km/s
ASA    :  38.409 km/s
GODLIKE:  46.128 km/s

======================
Sun - Venus - Mars
======================
GA     :  47.922 km/s
DE     :  43.451 km/s
PSO    :  47.734 km/s
ASA    :  59.439 km/s
GODLIKE:  51.210 km/s

======================
Sun - Venus - Jupiter
======================
GA     :  NaN    km/s
DE     :  NaN    km/s
PSO    :  37.699 km/s
ASA    :  NaN    km/s
GODLIKE:  NaN    km/s

======================
Sun - Venus - Saturn
======================
GA     :  39.040 km/s
DE     :  NaN    km/s
PSO    :  52.601 km/s
ASA    :  47.137 km/s
GODLIKE:  33.066 km/s

======================
Sun - Venus - Uranus
======================
GA     :  NaN    km/s
DE     :  54.616 km/s
PSO    :  51.759 km/s
ASA    :  NaN    km/s
GODLIKE:  NaN    km/s

======================
Sun - Earth - Mars
======================
GA     :  41.929 km/s
DE     :  46.162 km/s
PSO    :  48.312 km/s
ASA    :  NaN    km/s
GODLIKE:  42.810 km/s

======================
Sun - Earth - Jupiter
======================
GA     :  13.247 km/s
DE     :  22.761 km/s
PSO    :  12.481 km/s
ASA    :  13.264 km/s
GODLIKE:  13.649 km/s

======================
Sun - Earth - Saturn
======================
GA     :  37.777 km/s
DE     :  NaN    km/s
PSO    :  42.924 km/s
ASA    :  48.479 km/s
GODLIKE:  NaN    km/s

======================
Sun - Earth - Uranus
======================
GA     :  31.256 km/s
DE     :  NaN    km/s
PSO    :  62.273 km/s
ASA    :  NaN    km/s
GODLIKE:  48.890 km/s

======================
Sun - Mars - Jupiter
======================
GA     :  29.132 km/s
DE     :  21.121 km/s
PSO    :  27.024 km/s
ASA    :  28.523 km/s
GODLIKE:  29.688 km/s

======================
Sun - Mars - Saturn
======================
GA     :  30.877 km/s
DE     :  35.482 km/s
PSO    :  35.482 km/s
ASA    :  35.731 km/s
GODLIKE:  35.483 km/s

======================
Sun - Mars - Uranus
======================
GA     :  NaN    km/s
DE     :  NaN    km/s
PSO    :  44.514 km/s
ASA    :  53.001 km/s
GODLIKE:  38.844 km/s

======================
Sun - Jupiter - Saturn
======================
GA     :  25.259 km/s
DE     :  32.913 km/s
PSO    :  28.816 km/s
ASA    :  20.424 km/s
GODLIKE:  19.756 km/s

======================
Sun - Jupiter - Uranus
======================
GA     :  NaN    km/s
DE     :  28.878 km/s
PSO    :  70.754 km/s
ASA    :  19.056 km/s
GODLIKE:  17.775 km/s

======================
Sun - Saturn - Uranus
======================
GA     :  43.912 km/s
DE     :  NaN    km/s
PSO    :  46.340 km/s
ASA    :  48.089 km/s
GODLIKE:  43.723 km/s

======================
Mercury - Sun - Venus
======================
GA     :  59.528 km/s
DE     :  NaN    km/s
PSO    :  50.858 km/s
ASA    :  46.589 km/s
GODLIKE:  41.669 km/s

======================
Mercury - Sun - Earth
======================
GA     :  50.970 km/s
DE     :  64.462 km/s
PSO    :  45.640 km/s
ASA    :  56.848 km/s
GODLIKE:  51.241 km/s

======================
Mercury - Sun - Mars
======================
GA     :  53.830 km/s
DE     :  72.857 km/s
PSO    :  54.248 km/s
ASA    :  43.992 km/s
GODLIKE:  39.356 km/s

======================
Mercury - Sun - Jupiter
======================
GA     :  27.757 km/s
DE     :  52.652 km/s
PSO    :  27.257 km/s
ASA    :  48.876 km/s
GODLIKE:  36.889 km/s

======================
Mercury - Sun - Saturn
======================
GA     :  43.417 km/s
DE     :  89.931 km/s
PSO    :  93.071 km/s
ASA    :  93.255 km/s
GODLIKE:  64.224 km/s

======================
Mercury - Sun - Uranus
======================
GA     :  32.353 km/s
DE     :  26.284 km/s
PSO    :  27.400 km/s
ASA    :  53.659 km/s
GODLIKE:  39.314 km/s

======================
Mercury - Venus - Sun
======================
GA     :  26.598 km/s
DE     :  26.684 km/s
PSO    :  19.606 km/s
ASA    :  26.040 km/s
GODLIKE:  25.924 km/s

======================
Mercury - Venus - Jupiter
======================
GA     :  33.339 km/s
DE     :  30.746 km/s
PSO    :  32.605 km/s
ASA    :  39.148 km/s
GODLIKE:  32.358 km/s

======================
Mercury - Venus - Saturn
======================
GA     :  33.716 km/s
DE     :  43.522 km/s
PSO    :  39.317 km/s
ASA    :  31.166 km/s
GODLIKE:  43.589 km/s

======================
Mercury - Earth - Sun
======================
GA     :  21.858 km/s
DE     :  23.987 km/s
PSO    :  25.678 km/s
ASA    :  25.516 km/s
GODLIKE:  22.364 km/s

======================
Mercury - Earth - Jupiter
======================
GA     :  26.513 km/s
DE     :  28.804 km/s
PSO    :  102.94 km/s
ASA    :  194.61 km/s
GODLIKE:  25.684 km/s

======================
Mercury - Earth - Saturn
======================
GA     :  35.000 km/s
DE     :  43.915 km/s
PSO    :  55.351 km/s
ASA    :  29.264 km/s
GODLIKE:  292.18 km/s

======================
Mercury - Mars - Sun
======================
GA     :  24.144 km/s
DE     :  25.218 km/s
PSO    :  26.748 km/s
ASA    :  25.236 km/s
GODLIKE:  25.247 km/s

======================
Mercury - Mars - Jupiter
======================
GA     :  37.729 km/s
DE     :  36.152 km/s
PSO    :  30.354 km/s
ASA    :  107.79 km/s
GODLIKE:  29.632 km/s

======================
Mercury - Mars - Saturn
======================
GA     :  44.590 km/s
DE     :  39.505 km/s
PSO    :  69.456 km/s
ASA    :  38.268 km/s
GODLIKE:  53.483 km/s

======================
Mercury - Jupiter - Sun
======================
GA     :  32.839 km/s
DE     :  38.120 km/s
PSO    :  48.452 km/s
ASA    :  41.728 km/s
GODLIKE:  33.495 km/s

======================
Mercury - Jupiter - Saturn
======================
GA     :  36.877 km/s
DE     :  37.404 km/s
PSO    :  32.296 km/s
ASA    :  40.882 km/s
GODLIKE:  42.288 km/s

======================
Mercury - Jupiter - Uranus
======================
GA     :  37.020 km/s
DE     :  38.180 km/s
PSO    :  44.251 km/s
ASA    :  40.002 km/s
GODLIKE:  143.35 km/s

======================
Mercury - Saturn - Uranus
======================
GA     :  53.780 km/s
DE     :  53.758 km/s
PSO    :  79.593 km/s
ASA    :  48.114 km/s
GODLIKE:  60.002 km/s

======================
Venus - Sun - Mercury
======================
GA     :  31.549 km/s
DE     :  30.585 km/s
PSO    :  31.610 km/s
ASA    :  35.547 km/s
GODLIKE:  31.224 km/s

======================
Venus - Sun - Venus
======================
GA     :  23.202 km/s
DE     :  24.796 km/s
PSO    :  61.962 km/s
ASA    :  24.242 km/s
GODLIKE:  22.701 km/s

======================
Venus - Sun - Earth
======================
GA     :  30.748 km/s
DE     :  44.209 km/s
PSO    :  40.452 km/s
ASA    :  29.328 km/s
GODLIKE:  25.071 km/s

======================
Venus - Sun - Mars
======================
GA     :  43.917 km/s
DE     :  NaN    km/s
PSO    :  54.764 km/s
ASA    :  61.371 km/s
GODLIKE:  35.749 km/s

======================
Venus - Sun - Jupiter
======================
GA     :  24.911 km/s
DE     :  27.294 km/s
PSO    :  36.139 km/s
ASA    :  24.819 km/s
GODLIKE:  27.099 km/s

======================
Venus - Sun - Saturn
======================
GA     :  24.044 km/s
DE     :  32.264 km/s
PSO    :  41.973 km/s
ASA    :  43.408 km/s
GODLIKE:  19.846 km/s

======================
Venus - Sun - Uranus
======================
GA     :  20.885 km/s
DE     :  26.166 km/s
PSO    :  31.373 km/s
ASA    :  27.481 km/s
GODLIKE:  28.063 km/s

======================
Venus - Mercury - Sun
======================
GA     :  26.143 km/s
DE     :  26.764 km/s
PSO    :  26.896 km/s
ASA    :  26.130 km/s
GODLIKE:  27.031 km/s

======================
Venus - Mercury - Jupiter
======================
GA     :  41.200 km/s
DE     :  25.983 km/s
PSO    :  80.851 km/s
ASA    :  23.475 km/s
GODLIKE:  276.86 km/s

======================
Venus - Mercury - Saturn
======================
GA     :  32.999 km/s
DE     :  43.302 km/s
PSO    :  182.28 km/s
ASA    :  45.702 km/s
GODLIKE:  61.772 km/s

======================
Venus - Venus - Sun
======================
GA     :  28.981 km/s
DE     :  32.985 km/s
PSO    :  34.221 km/s
ASA    :  34.444 km/s

======================
Venus - Venus- Jupiter
======================
GA     :  20.750 km/s
DE     :  17.238 km/s
PSO    :  42.430 km/s
ASA    :  15.458 km/s

======================
Venus - Venus - Saturn
======================
GA     :  29.351 km/s
DE     :  20.722 km/s
PSO    :  35.104 km/s
ASA    :  21.586 km/s

======================
Venus - Earth - Sun
======================
GA     :  25.335 km/s
DE     :  27.962 km/s
PSO    :  27.372 km/s
ASA    :  34.315 km/s

======================
Venus - Earth - Jupiter
======================
GA     :  20.028 km/s
DE     :  22.170 km/s
PSO    :  38.553 km/s
ASA    :  23.485 km/s
```

GODLIKE: 26.729 km/s

Venus - Earth - Saturn
GA : 28.062 km/s
DE : 25.842 km/s
PSO : 30.244 km/s
ASA : 26.036 km/s
GODLIKE: 34.519 km/s

Venus - Jupiter - Mercury
GA : 31.693 km/s
DE : 31.726 km/s
PSO : 28.567 km/s
ASA : 29.319 km/s
GODLIKE: 28.539 km/s

Earth - Sun - Venus
GA : 25.928 km/s
DE : 40.406 km/s
PSO : 29.721 km/s
ASA : 36.032 km/s
GODLIKE: 22.122 km/s

Earth - Sun - Uranus
GA : 29.023 km/s
DE : 28.414 km/s
PSO : 29.358 km/s
ASA : 28.530 km/s
GODLIKE: 41.692 km/s

Earth - Venus - Jupiter
GA : 24.739 km/s
DE : 21.582 km/s
PSO : 86.981 km/s
ASA : 20.719 km/s
GODLIKE: 32.434 km/s

Earth - Mars - Sun
GA : 27.901 km/s
DE : 33.072 km/s
PSO : 29.760 km/s
ASA : 33.993 km/s
GODLIKE: 26.389 km/s

Earth - Jupiter - Saturn
GA : 25.636 km/s
DE : 24.225 km/s
PSO : 29.819 km/s
ASA : 24.536 km/s
GODLIKE: 25.299 km/s

Mars - Sun - Earth
GA : 115.02 km/s
DE : 37.078 km/s
PSO : 33.041 km/s
ASA : 34.742 km/s
GODLIKE: 30.148 km/s

Mars - Mercury - Jupiter
GA : 36.637 km/s
DE : 30.286 km/s
PSO : 55.233 km/s
ASA : 30.959 km/s
GODLIKE: 37.199 km/s

Mars - Earth - Sun
GA : 25.314 km/s
DE : 25.526 km/s
PSO : 26.146 km/s
ASA : 25.900 km/s
GODLIKE: 24.966 km/s

Mars - Jupiter - Saturn

GODLIKE: 26.022 km/s

Venus - Mars - Sun
GA : 25.264 km/s
DE : 25.739 km/s
PSO : 26.660 km/s
ASA : 24.844 km/s
GODLIKE: 25.312 km/s

Venus - Jupiter - Saturn
GA : 31.535 km/s
DE : 26.778 km/s
PSO : 26.353 km/s
ASA : 26.723 km/s
GODLIKE: 34.964 km/s

Earth - Sun - Earth
GA : 21.288 km/s
DE : 25.606 km/s
PSO : 22.387 km/s
ASA : 36.553 km/s
GODLIKE: 20.207 km/s

Earth - Mercury - Sun
GA : 26.335 km/s
DE : 26.786 km/s
PSO : 26.824 km/s
ASA : 26.588 km/s
GODLIKE: 24.663 km/s

Earth - Venus - Saturn
GA : 30.009 km/s
DE : 28.858 km/s
PSO : 34.508 km/s
ASA : 24.667 km/s
GODLIKE: 33.283 km/s

Earth - Mars - Jupiter
GA : 30.838 km/s
DE : 27.914 km/s
PSO : 40.366 km/s
ASA : 27.568 km/s
GODLIKE: 129.79 km/s

Earth - Jupiter - Uranus
GA : 25.266 km/s
DE : 20.852 km/s
PSO : 77.999 km/s
ASA : 20.764 km/s
GODLIKE: 26.527 km/s

Mars - Sun - Jupiter
GA : 17.480 km/s
DE : 22.955 km/s
PSO : 26.298 km/s
ASA : 18.501 km/s
GODLIKE: 19.858 km/s

Mars - Mercury - Saturn
GA : 172.52 km/s
DE : 37.355 km/s
PSO : 144.64 km/s
ASA : 45.512 km/s
GODLIKE: 58.102 km/s

Mars - Earth - Jupiter
GA : 25.599 km/s
DE : 23.191 km/s
PSO : 37.197 km/s
ASA : 12.008 km/s
GODLIKE: 24.941 km/s

Mars - Jupiter - Uranus

GODLIKE: 151.52 km/s

Venus - Mars - Jupiter
GA : 24.632 km/s
DE : 17.047 km/s
PSO : 31.219 km/s
ASA : 22.173 km/s
GODLIKE: 84.904 km/s

Venus - Jupiter - Uranus
GA : 31.122 km/s
DE : 27.768 km/s
PSO : 54.110 km/s
ASA : 27.769 km/s
GODLIKE: 35.839 km/s

Earth - Sun - Mars
GA : 32.674 km/s
DE : 42.323 km/s
PSO : 42.558 km/s
ASA : 37.381 km/s
GODLIKE: 31.676 km/s

Earth - Mercury - Jupiter
GA : 33.126 km/s
DE : 36.984 km/s
PSO : 38.164 km/s
ASA : 34.533 km/s
GODLIKE: 35.771 km/s

Earth - Earth - Sun
GA : 23.835 km/s
DE : 24.548 km/s
PSO : 24.706 km/s
ASA : 24.093 km/s
GODLIKE: 23.522 km/s

Earth - Mars - Saturn
GA : 34.356 km/s
DE : 32.726 km/s
PSO : 43.518 km/s
ASA : 33.008 km/s
GODLIKE: 38.500 km/s

Earth - Saturn - Uranus
GA : 45.312 km/s
DE : 48.921 km/s
PSO : 58.713 km/s
ASA : 45.849 km/s
GODLIKE: 96.639 km/s

Mars - Sun - Saturn
GA : 26.059 km/s
DE : 29.291 km/s
PSO : 33.457 km/s
ASA : 37.882 km/s
GODLIKE: 26.548 km/s

Mars - Venus - Sun
GA : 25.960 km/s
DE : 27.632 km/s
PSO : 37.720 km/s
ASA : 30.561 km/s
GODLIKE: 26.069 km/s

Mars - Earth - Saturn
GA : 28.090 km/s
DE : 24.926 km/s
PSO : 858.60 km/s
ASA : 25.000 km/s
GODLIKE: 34.543 km/s

Mars - Saturn - Uranus

GODLIKE: 27.875 km/s

Venus - Mars - Saturn
GA : 33.589 km/s
DE : 34.392 km/s
PSO : 51.747 km/s
ASA : 34.841 km/s
GODLIKE: 47.419 km/s

Venus - Saturn - Uranus
GA : 50.441 km/s
DE : 47.599 km/s
PSO : 43.751 km/s
ASA : 44.145 km/s
GODLIKE: 91.613 km/s

Earth - Sun - Jupiter
GA : 25.350 km/s
DE : 30.188 km/s
PSO : 30.687 km/s
ASA : 27.049 km/s
GODLIKE: 16.843 km/s

Earth - Mercury - Saturn
GA : 40.749 km/s
DE : 40.396 km/s
PSO : 44.764 km/s
ASA : 48.582 km/s
GODLIKE: 82.705 km/s

Earth - Earth - Jupiter
GA : 13.459 km/s
DE : 12.730 km/s
PSO : 16.109 km/s
ASA : 12.704 km/s
GODLIKE: 24.091 km/s

Earth - Jupiter - Sun
GA : 23.445 km/s
DE : 25.080 km/s
PSO : 25.888 km/s
ASA : 24.116 km/s
GODLIKE: 24.585 km/s

Mars - Sun - Mercury
GA : 35.243 km/s
DE : 34.483 km/s
PSO : 31.659 km/s
ASA : 48.728 km/s
GODLIKE: 37.429 km/s

Mars - Sun - Uranus
GA : 29.518 km/s
DE : 34.787 km/s
PSO : 35.486 km/s
ASA : 29.397 km/s
GODLIKE: 30.951 km/s

Mars - Venus - Jupiter
GA : 28.534 km/s
DE : 24.876 km/s
PSO : 36.496 km/s
ASA : 15.434 km/s
GODLIKE: 27.202 km/s

Mars - Jupiter - Sun
GA : 24.743 km/s
DE : 31.034 km/s
PSO : 27.927 km/s
ASA : 32.194 km/s
GODLIKE: 22.759 km/s

Jupiter - Sun - Mercury

GODLIKE: 32.527 km/s

Venus - Jupiter - Sun
GA : 22.857 km/s
DE : 24.861 km/s
PSO : 24.785 km/s
ASA : 25.997 km/s
GODLIKE: 23.267 km/s

Earth - Sun - Mercury
GA : 46.336 km/s
DE : 52.133 km/s
PSO : 47.360 km/s
ASA : 37.452 km/s
GODLIKE: 35.673 km/s

Earth - Sun - Saturn
GA : 30.980 km/s
DE : 31.076 km/s
PSO : 30.594 km/s
ASA : 87.320 km/s
GODLIKE: 47.734 km/s

Earth - Venus - Sun
GA : 22.748 km/s
DE : NaN km/s
PSO : 30.686 km/s
ASA : 33.074 km/s
GODLIKE: 27.844 km/s

Earth - Earth - Saturn
GA : 27.875 km/s
DE : 21.078 km/s
PSO : 28.836 km/s
ASA : 21.580 km/s
GODLIKE: 20.184 km/s

Earth - Jupiter - Mercury
GA : 28.168 km/s
DE : 30.549 km/s
PSO : 42.782 km/s
ASA : 27.715 km/s
GODLIKE: 33.325 km/s

Mars - Sun - Venus
GA : 23.956 km/s
DE : NaN km/s
PSO : 33.383 km/s
ASA : 32.339 km/s
GODLIKE: 27.981 km/s

Mars - Mercury - Sun
GA : 25.209 km/s
DE : 32.959 km/s
PSO : 44.859 km/s
ASA : 45.991 km/s
GODLIKE: 35.447 km/s

Mars - Venus - Saturn
GA : 28.405 km/s
DE : 29.424 km/s
PSO : 36.834 km/s
ASA : 64.296 km/s
GODLIKE: 192.81 km/s

Mars - Jupiter - Mercury
GA : 90.781 km/s
DE : 37.191 km/s
PSO : 94.287 km/s
ASA : 32.494 km/s
GODLIKE: 33.811 km/s

Jupiter - Sun - Venus

```
======================
GA     :  25.203 km/s
DE     :  25.046 km/s
PSO    :  32.817 km/s
ASA    :  33.514 km/s
GODLIKE:  25.022 km/s

======================
GA     :  31.771 km/s
DE     :  31.968 km/s
PSO    :  36.839 km/s
ASA    :  31.778 km/s
GODLIKE:  31.776 km/s

======================
GA     :  62.180 km/s
DE     :  54.546 km/s
PSO    :  69.897 km/s
ASA    :  56.512 km/s
GODLIKE:  72.027 km/s

======================
GA     :  15.579 km/s
DE     :  25.360 km/s
PSO    :  32.216 km/s
ASA    :  36.325 km/s
GODLIKE:  73.283 km/s

======================
GA     :  21.692 km/s
DE     :  34.619 km/s
PSO    :  50.701 km/s
ASA    :  48.603 km/s
GODLIKE:  28.445 km/s

======================
Jupiter - Sun - Earth
======================
GA     :  22.082 km/s
DE     :  28.863 km/s
PSO    :  39.538 km/s
ASA    :  35.844 km/s
GODLIKE:  31.058 km/s

======================
Jupiter - Sun - Mars
======================
GA     :  26.745 km/s
DE     :  32.666 km/s
PSO    :  40.011 km/s
ASA    :  27.525 km/s
GODLIKE:  27.205 km/s

======================
Jupiter - Sun - Jupiter
======================
GA     :  15.138 km/s
DE     :  36.672 km/s
PSO    :  33.728 km/s
ASA    :  33.709 km/s
GODLIKE:  18.769 km/s

======================
Jupiter - Sun - Saturn
======================
GA     :  35.717 km/s
DE     :  48.601 km/s
PSO    :  40.750 km/s
ASA    :  41.384 km/s
GODLIKE:  36.298 km/s

======================
Jupiter - Sun - Uranus
======================
GA     :  28.026 km/s
DE     :  NaN    km/s
PSO    :  34.849 km/s
ASA    :  47.680 km/s
GODLIKE:  33.150 km/s

======================
Jupiter - Mercury - Venus
======================
GA     :  52.364 km/s
DE     :  46.336 km/s
PSO    :  54.319 km/s
ASA    :  43.805 km/s
GODLIKE:  49.700 km/s

======================
Jupiter - Mercury - Earth
======================
GA     :  36.966 km/s
DE     :  58.164 km/s
PSO    :  69.942 km/s
ASA    :  54.422 km/s
GODLIKE:  37.994 km/s

======================
Jupiter - Mercury - Mars
======================
GA     :  32.474 km/s
DE     :  45.937 km/s
PSO    :  65.829 km/s
ASA    :  37.471 km/s
GODLIKE:  36.579 km/s

======================
Jupiter - Mercury - Saturn
======================
GA     :  23.635 km/s
DE     :  31.974 km/s
PSO    :  28.164 km/s
ASA    :  25.198 km/s
GODLIKE:  27.227 km/s

======================
Jupiter - Mercury - Uranus
======================
GA     :  28.249 km/s
DE     :  27.265 km/s
PSO    :  29.430 km/s
ASA    :  38.789 km/s
GODLIKE:  25.304 km/s

======================
Jupiter - Saturn - Uranus
======================
GA     :  45.559 km/s
DE     :  47.697 km/s
PSO    :  45.119 km/s
ASA    :  46.608 km/s
GODLIKE:  38.819 km/s
```

G.2. High-thrust, First-order, Best 10, Stagnation Point

```
============================================================
Jupiter
============================================================
GA     :  6.3925   6.4081   6.9914   6.9922   8.0195 (km/s)
DE     :  6.9117   6.9117   6.9891   6.9117   6.9117 (km/s)
PSO    :  7.1735   7.2209   7.2246   7.2159   6.9970 (km/s)
ASA    :  7.7537   6.9117   8.2891   7.2280   8.6814 (km/s)
GODLIKE:  6.9117   6.3925   6.9902   6.3927   6.3934 (km/s)

============================================================
Jupiter Uranus
============================================================
GA     : 10.4411  10.0976  10.3102  10.3385  10.4194 (km/s)
DE     : 19.5189  19.5189  10.0877  19.5189  19.5189 (km/s)
PSO    : 14.6508  15.5783  17.1697  11.5133  11.9089 (km/s)
ASA    : 22.4589  20.2807  30.1946  15.1633  15.3582 (km/s)
GODLIKE: 10.2149  13.2533  10.1187  11.6005  10.0886 (km/s)

============================================================
Earth Jupiter
============================================================
GA     : 14.0216  30.0315  13.2407  12.9225  12.8917 (km/s)
DE     : 11.6188  15.3145  12.1304  11.6006  11.7034 (km/s)
PSO    : 13.2577  14.0646  12.8426  14.6003  15.1619 (km/s)
ASA    : 14.5057  16.7278  14.2039  13.6920  13.8627 (km/s)
GODLIKE: 12.7994  11.8019  12.8001  13.0289  12.1811 (km/s)

============================================================
Sun
============================================================
GA     : 12.9845  12.9859  12.9937  12.9971  12.9882 (km/s)
DE     : 12.9859  12.9859  12.9882  12.9845  12.9859 (km/s)
PSO    : 13.0000  12.9883  13.0027  12.9859  12.9910 (km/s)
ASA    : 12.9937  12.9859  13.0030  13.0044  12.9859 (km/s)
GODLIKE: 13.0027  13.0000  12.9859  12.9859  12.9859 (km/s)

============================================================
Venus Jupiter
============================================================
GA     : 12.6449  13.3574  14.1132  13.6945  13.0136 (km/s)
DE     : 13.8944  12.5432  12.4707  11.4488  11.5070 (km/s)
PSO    : 15.3188  16.6187  15.1745  27.5594  17.3869 (km/s)
ASA    : 20.7028  20.9439  14.2075  23.5926  17.2485 (km/s)
GODLIKE: 12.6355  12.5713  12.4704  14.4301  13.0434 (km/s)

============================================================
Sun Jupiter
============================================================
GA     :  6.4330   6.3171   6.3774   6.3132  13.0256 (km/s)
DE     :  6.2842   6.2842   6.2842   6.2842   6.2843 (km/s)
PSO    :  6.3439  15.5436   8.6731  10.6069  11.1893 (km/s)
ASA    : 14.2310  16.0846  15.6705  14.8357  14.2618 (km/s)
GODLIKE: 20.3534   8.1394   7.7662   7.6357   6.4366 (km/s)

============================================================
Mars Earth Jupiter
============================================================
GA     : 12.0672  18.7218  23.2527  12.6124  13.1649 (km/s)
DE     : 17.5163  17.7038  17.5496  11.8773  20.3577 (km/s)
PSO    : 36.6833  29.4206  31.0052  29.9862  29.8163 (km/s)
ASA    : 49.6597  29.8815  30.2718  32.1885  30.3028 (km/s)
GODLIKE: 34.3570  22.0991  21.8474  25.7380  18.1183 (km/s)

============================================================
Sun Earth Jupiter
============================================================
GA     : 12.5681  17.0518  12.5558  13.6298  13.7374 (km/s)
DE     : 20.4124  19.0302  17.4246  20.6846  24.8004 (km/s)
PSO    : 27.6053  29.3465  16.1713  19.1708  16.5275 (km/s)
ASA    : 28.4539  17.0978  18.5061  16.8321  18.4143 (km/s)
GODLIKE: 18.7779  13.6515  23.2708  13.6830  30.3341 (km/s)

============================================================
Earth Earth Jupiter
============================================================
GA     : 13.9098  12.9193  22.4066  13.3197  14.0279 (km/s)
DE     : 13.2294  13.1323  12.9840  14.0030  13.1346 (km/s)
PSO    : 13.3082  13.3475  13.2941  13.6179  12.8520 (km/s)
ASA    : 20.6586  17.4108  18.6284  13.4247  13.7199 (km/s)
GODLIKE: 13.3452  13.7592  13.1875  13.8012  13.0220 (km/s)

============================================================
Jupiter Sun
============================================================
GA     : 13.5740  13.5176   9.4601  13.6980   9.4223 (km/s)
DE     :  8.9513   8.9513   8.9547   8.9513   8.9512 (km/s)
PSO    : 13.4505   8.9510   8.9512   8.9513   8.9514 (km/s)
ASA    :  8.9513  13.5143   9.4222   8.9509   9.4222 (km/s)
GODLIKE: 13.4461  13.5519  15.1240  13.4461   9.4629 (km/s)
```

G.3. High-thrust, First-order, Best 3, Surrounding Points

Jupiter (point 2)

GA	7.0068	7.0068	7.0068	7.0068	6.9792 (km/s)
DE	6.8694	7.3319	7.0068	7.0068	6.8747 (km/s)
PSO	16.2728	10.6435	10.3099	8.3129	10.1845 (km/s)
ASA	11.7541	8.5164	9.7106	7.0068	9.2520 (km/s)
GODLIKE:	7.0068	7.6973	6.8910	6.8756	7.0068 (km/s)

Sun Jupiter (point 2)

GA	10.5869	7.5725	8.6273	6.2609	7.8487 (km/s)
DE	8.8472	9.5637	10.2985	6.4718	7.0210 (km/s)
PSO	6.4863	20.2676	17.4803	8.9196	18.0216 (km/s)
ASA	12.9513	17.7972	15.6906	8.3208	9.6541 (km/s)
GODLIKE:	7.5141	8.9092	6.2205	6.6999	7.6265 (km/s)

Jupiter (point 3)

GA	5.9822	6.0753	6.0378	5.9882	5.9850 (km/s)
DE	5.9992	6.0103	5.9752	5.9910	5.9753 (km/s)
PSO	9.5781	7.1641	8.4479	11.0504	9.2557 (km/s)
ASA	8.0592	6.7919	12.1494	8.2878	11.9228 (km/s)
GODLIKE:	6.8249	6.2110	6.0087	5.9752	5.9752 (km/s)

Sun Jupiter (point 3)

GA	9.5581	6.7306	7.7038	13.3454	9.5657 (km/s)
DE	8.0036	7.8732	6.7352	9.8419	7.6634 (km/s)
PSO	12.2864	10.3717	16.7137	7.7999	13.4745 (km/s)
ASA	7.4520	12.0133	9.3197	15.2677	9.9089 (km/s)
GODLIKE:	7.0335	9.6153	7.8732	6.6191	7.8733 (km/s)

Jupiter (point 4)

GA	6.0100	5.9898	6.1494	14.3081	7.9799 (km/s)
DE	5.9762	5.9752	5.9754	7.7837	5.9783 (km/s)
PSO	10.1264	11.7996	9.2529	9.8145	9.3543 (km/s)
ASA	7.8935	8.1055	6.7393	11.6695	10.1415 (km/s)
GODLIKE:	5.9752	5.9767	5.9752	5.9752	5.9762 (km/s)

Sun Jupiter (point 4)

GA	6.9198	7.3443	8.1492	6.6078	7.2374 (km/s)
DE	12.9723	12.2610	7.7590	8.1425	8.2353 (km/s)
PSO	11.8419	14.7262	12.4196	8.2225	7.1530 (km/s)
ASA	9.8041	12.1215	15.4156	16.7994	14.0829 (km/s)
GODLIKE:	6.8452	6.8802	6.8539	10.4278	7.5112 (km/s)

Jupiter (point 5)

GA	6.1967	6.9685	6.1891	6.1898	6.1882 (km/s)
DE	7.1083	7.4195	8.8860	6.2036	6.1920 (km/s)
PSO	9.9265	10.5046	7.5816	7.9884	10.6831 (km/s)
ASA	13.3705	8.0101	12.0917	7.6809	11.1989 (km/s)
GODLIKE:	6.3050	6.1911	8.0128	13.9969	6.1889 (km/s)

Sun Jupiter (point 5)

GA	14.8669	9.9735	8.0192	9.1198	10.2085 (km/s)
DE	10.0242	6.8772	12.0558	6.9092	22.3410 (km/s)
PSO	8.0755	10.4483	13.8153	9.0203	10.3294 (km/s)
ASA	11.4880	10.9073	8.5729	15.5759	12.5433 (km/s)
GODLIKE:	9.9866	9.9715	8.0954	6.7829	10.3021 (km/s)

Jupiter (point 6)

GA	7.1852	7.1852	7.1852	7.3559	7.1852 (km/s)
DE	10.9979	7.1345	7.1340	7.1592	10.6547 (km/s)
PSO	9.4172	10.3903	8.4761	11.0509	8.7239 (km/s)
ASA	10.7197	14.9482	9.9307	12.7947	12.4850 (km/s)
GODLIKE:	7.1852	7.2637	7.2688	7.1340	7.1344 (km/s)

Sun Jupiter (point 6)

GA	6.4678	7.6359	8.2284	9.2763	9.2057 (km/s)
DE	7.1042	7.7759	12.8910	9.4249	11.7492 (km/s)
PSO	13.1809	9.1899	14.1824	16.6891	7.9851 (km/s)
ASA	11.8348	12.1109	9.2792	16.3855	12.7700 (km/s)
GODLIKE:	6.5614	7.6864	6.6141	7.4605	7.6352 (km/s)

Jupiter (point 7)

GA	6.3770	6.3770	13.6557	13.5672	6.3770 (km/s)
DE	6.3770	6.3772	6.3770	6.3770	6.3770 (km/s)
PSO	9.4737	7.6734	9.2218	11.7891	10.0170 (km/s)
ASA	17.0011	10.4383	9.1443	15.7444	7.3711 (km/s)
GODLIKE:	6.3778	6.3770	7.5915	6.3770	6.3770 (km/s)

Sun Jupiter (point 7)

GA	6.0702	7.0803	20.7179	10.1798	17.2066 (km/s)
DE	20.9738	9.0804	9.5816	7.5136	10.4387 (km/s)
PSO	14.8623	11.6239	12.1225	9.2171	14.5730 (km/s)
ASA	11.6053	7.4950	8.2242	7.0803	14.4757 (km/s)
GODLIKE:	7.0803	7.0803	7.0803	9.2292	7.0803 (km/s)

Jupiter (point 8)

GA	5.9880	5.9880	5.9880	5.9880	5.9880 (km/s)
DE	5.9880	13.3363	9.6969	5.9880	5.9880 (km/s)
PSO	6.8959	8.2348	11.6095	9.7039	14.8316 (km/s)
ASA	10.7101	11.4085	15.2212	13.9467	11.6217 (km/s)
GODLIKE:	5.9880	5.9880	5.9880	5.9880	5.9880 (km/s)

Sun Jupiter (point 8)

GA	7.2421	9.7637	8.1033	18.7298	10.0107 (km/s)
DE	5.8310	5.8233	8.0711	16.4471	7.9829 (km/s)
PSO	16.0248	8.6671	12.2570	12.2513	10.0354 (km/s)
ASA	6.7665	6.7746	8.6808	6.9258	5.9547 (km/s)
GODLIKE:	5.8233	8.4325	5.8233	5.8241	5.8237 (km/s)

Jupiter (point 9)

GA	6.2311	6.4158	6.2311	13.9989	8.0375 (km/s)
DE	9.0968	6.2311	6.2311	6.2311	13.3769 (km/s)
PSO	8.0333	13.8825	14.9754	9.9501	16.1315 (km/s)
ASA	16.0097	7.6868	13.9803	10.6646	13.8807 (km/s)
GODLIKE:	6.2311	8.8220	6.2311	6.2311	6.2311 (km/s)

Sun Jupiter (point 9)

GA	13.4324	8.6368	6.2689	6.2524	6.3252 (km/s)
DE	6.2432	6.2377	6.2362	6.2323	6.2313 (km/s)
PSO	7.0976	8.8836	9.8281	8.5541	9.0506 (km/s)
ASA	9.7062	10.9026	7.9196	10.3098	9.0174 (km/s)
GODLIKE:	6.2311	6.2513	6.2311	6.2315	6.3060 (km/s)

Jupiter (point 10)

GA	7.0042	7.1157	14.0673	7.0042	7.0042 (km/s)
DE	6.9190	6.9191	7.1857	6.9194	7.0017 (km/s)
PSO	13.9416	8.5132	7.5606	10.5195	7.8100 (km/s)
ASA	7.9973	10.8685	9.2621	8.1821	18.1449 (km/s)
GODLIKE:	7.0042	6.9190	17.1710	13.7649	7.0952 (km/s)

Sun Jupiter (point 10)

GA	7.0043	7.0260	7.0057	7.5991	6.9342 (km/s)
DE	6.9388	7.0051	7.0086	7.0049	6.9194 (km/s)
PSO	7.2065	7.3250	7.3414	11.8395	8.0800 (km/s)
ASA	11.0994	15.1339	8.6019	10.2601	7.1148 (km/s)
GODLIKE:	6.9190	7.0185	7.0042	7.0056	7.0043 (km/s)

Jupiter (point 11)

GA	5.4372	4.9831	11.1335	5.325498	4.9878 (km/s)
DE	6.3518	4.9657	5.7019	4.977568	4.9732 (km/s)
PSO	7.8822	12.6419	14.5865	8.720934	8.2797 (km/s)
ASA	6.2000	7.2670	7.5160	7.992071	9.3956 (km/s)
GODLIKE:	4.9657	5.3685	5.1510	5.849917	4.9657 (km/s)

Sun Jupiter (point 11)

GA	5.0359	4.9795	5.1467	4.9765	10.4464 (km/s)
DE	5.9356	4.9657	4.9930	4.9657	4.9658 (km/s)
PSO	10.3893	7.2648	12.4093	6.8617	7.8200 (km/s)
ASA	10.1495	11.3375	8.4172	9.3743	6.1668 (km/s)
GODLIKE:	4.9657	4.9657	4.9657	4.9657	4.9670 (km/s)

Jupiter (point 12)

GA	:	4.1641	4.3211	4.1101	4.1375	4.1184 (km/s)
DE	:	4.1277	4.1636	4.0965	4.0965	4.1130 (km/s)
PSO	:	9.1465	7.0756	8.0672	17.0268	9.5183 (km/s)
ASA	:	8.9793	8.2366	9.0998	4.5218	8.2211 (km/s)
GODLIKE:		4.0965	7.9231	4.0965	4.0965	4.0965 (km/s)

Jupiter (point 13)

GA	:	5.4846	5.4675	5.4599	13.9210	8.9752 (km/s)
DE	:	5.4608	5.4580	16.2592	6.0003	5.4566 (km/s)
PSO	:	7.6806	12.3536	15.0205	12.2398	7.6754 (km/s)
ASA	:	14.4648	7.4515	12.1737	8.4485	10.1297 (km/s)
GODLIKE:		5.4530	5.5091	5.8509	5.7860	5.5165 (km/s)

Jupiter (point 14)

GA	:	7.3546	7.6132	7.3546	7.5171	7.4645 (km/s)
DE	:	7.4658	7.4431	7.5433	7.4429	7.3546 (km/s)
PSO	:	10.1825	11.1681	8.7231	8.4135	11.4227 (km/s)
ASA	:	7.9251	9.2153	10.4849	8.0531	8.9908 (km/s)
GODLIKE:		8.6197	11.1258	7.3546	7.4490	7.3546 (km/s)

Jupiter (point 15)

GA	:	5.8260	6.1393	5.8260	5.8260	5.8260 (km/s)
DE	:	5.8260	5.8260	13.4013	5.8260	5.8260 (km/s)
PSO	:	8.2514	7.4274	9.8332	15.2609	8.9192 (km/s)
ASA	:	17.6346	9.3022	12.7555	13.6332	16.2043 (km/s)
GODLIKE:		5.8260	5.8260	5.8260	14.4197	5.8260 (km/s)

Jupiter (point 16)

GA	:	5.0738	5.0739	5.0738	5.0738	13.2401 (km/s)
DE	:	5.0738	5.0738	5.0738	5.0738	5.0738 (km/s)
PSO	:	16.2661	8.0480	15.1140	5.8439	14.6521 (km/s)
ASA	:	6.5038	9.8194	13.3028	12.9375	12.5684 (km/s)
GODLIKE:		8.8626	5.0738	5.0738	5.0738	5.0738 (km/s)

Jupiter (point 17)

GA	:	14.1282	5.5079	5.5815	5.5079	5.5079 (km/s)
DE	:	5.5079	5.5079	5.5079	5.5079	5.5079 (km/s)
PSO	:	15.5674	15.7616	9.5474	5.9605	14.3995 (km/s)
ASA	:	13.6291	6.4439	14.1982	9.0694	8.2049 (km/s)
GODLIKE:		5.5079	5.5079	6.5438	5.5079	5.5079 (km/s)

Sun Jupiter (point 12)

GA	:	4.1546	5.2132	4.2885	4.1331	4.1104 (km/s)
DE	:	6.2609	4.1378	4.0982	4.0969	4.0976 (km/s)
PSO	:	5.3547	4.7310	6.8104	14.2287	6.5234 (km/s)
ASA	:	7.4869	15.3484	8.9994	10.3911	9.0264 (km/s)
GODLIKE:		4.0965	5.3884	4.0965	4.1016	4.0965 (km/s)

Sun Jupiter (point 13)

GA	:	5.7695	5.5095	5.4745	5.5400	5.4579 (km/s)
DE	:	5.4544	5.4537	5.4530	5.4532	5.4601 (km/s)
PSO	:	9.1651	10.3854	6.3096	10.3278	11.6580 (km/s)
ASA	:	7.1745	9.2978	6.3077	7.0084	9.9263 (km/s)
GODLIKE:		5.4530	5.4530	5.5768	5.4530	5.4530 (km/s)

Sun Jupiter (point 14)

GA	:	7.4596	7.4317	7.5271	7.3670	7.3785 (km/s)
DE	:	7.4428	7.3583	7.3547	7.3630	7.4429 (km/s)
PSO	:	9.2373	9.7596	9.8568	10.7868	9.7895 (km/s)
ASA	:	11.8030	8.6481	10.4923	9.0778	10.8899 (km/s)
GODLIKE:		7.3571	7.3546	7.3591	7.3608	7.3553 (km/s)

Sun Jupiter (point 15)

GA	:	5.8995	5.9181	5.8388	5.8716	8.0454 (km/s)
DE	:	5.8637	5.8280	5.8306	7.1782	5.8352 (km/s)
PSO	:	11.6703	11.1572	8.0391	6.5045	10.7687 (km/s)
ASA	:	8.0387	11.6048	5.8597	7.3901	13.8721 (km/s)
GODLIKE:		5.8262	5.8260	5.8264	5.8387	5.8264 (km/s)

Sun Jupiter (point 16)

GA	:	5.0739	5.0782	5.0738	5.0909	5.1437 (km/s)
DE	:	5.0793	5.0753	5.0738	5.0850	5.1166 (km/s)
PSO	:	7.2335	12.9497	6.6321	8.7604	5.2284 (km/s)
ASA	:	7.5249	11.0863	14.3428	8.6595	5.4089 (km/s)
GODLIKE:		5.0738	5.0738	5.0738	5.1360	5.1182 (km/s)

Sun Jupiter (point 17)

GA	:	6.1440	5.6233	5.5165	5.6511	5.5824 (km/s)
DE	:	5.5169	5.5090	5.5100	5.5524	5.5278 (km/s)
PSO	:	8.6466	7.4224	6.3202	6.8714	15.7644 (km/s)
ASA	:	14.2856	11.4712	14.0429	9.2776	11.2650 (km/s)
GODLIKE:		5.5083	5.9107	5.5079	6.1439	5.5192 (km/s)

Jupiter Uranus (point 2)

GA	:	9.8166	24.3736	8.1372	11.1251	8.7757 (km/s)
DE	:	8.3195	9.4250	8.9409	8.8348	28.7417 (km/s)
PSO	:	36.4019	29.5090	26.4037	33.9848	22.1909 (km/s)
ASA	:	37.0971	35.9450	36.9354	35.3230	36.5242 (km/s)
GODLIKE:		8.0243	8.0275	8.3623	8.0891	8.0243 (km/s)

Jupiter Uranus (point 4)

GA	:	8.7011	8.8744	8.4628	8.6779	10.2979 (km/s)
DE	:	8.2997	9.3460	8.6226	18.8333	8.2365 (km/s)
PSO	:	17.3688	28.1197	25.6553	34.2159	19.2867 (km/s)
ASA	:	33.1582	36.6359	22.8781	15.6640	36.8090 (km/s)
GODLIKE:		8.2916	8.1153	13.9795	8.1121	8.2937 (km/s)

Jupiter Uranus (point 6)

GA	:	13.9383	13.9007	14.5278	16.5073	16.2403 (km/s)
DE	:	27.4238	14.1877	19.3588	13.8099	13.9613 (km/s)
PSO	:	35.9955	36.5197	37.0112	37.1005	19.0223 (km/s)
ASA	:	36.9833	31.2356	37.1019	27.9022	20.7384 (km/s)
GODLIKE:		18.2262	13.7355	29.8760	13.9705	18.4893 (km/s)

Jupiter Uranus (point 8)

GA	:	14.3832	19.1113	14.4199	23.0956	20.0451 (km/s)
DE	:	13.5432	37.0806	13.7052	13.6101	75.0671 (km/s)
PSO	:	34.2624	33.6875	33.3215	19.6833	37.1011 (km/s)

Jupiter Uranus (point 3)

GA	:	6.7976	8.3488	6.2697	6.1595	6.7542 (km/s)
DE	:	10.2610	24.1525	13.0798	29.6799	6.1303 (km/s)
PSO	:	35.7992	12.1413	12.9682	19.6795	26.4116 (km/s)
ASA	:	37.0447	31.0030	19.5979	74.2656	35.9110 (km/s)
GODLIKE:		5.9896	5.9652	20.1612	6.3043	6.3496 (km/s)

Jupiter Uranus (point 5)

GA	:	27.1714	14.1114	11.5751	27.0515	14.6738 (km/s)
DE	:	11.9095	11.6192	12.4582	12.1395	11.5899 (km/s)
PSO	:	34.4769	34.3808	32.0464	34.8019	24.5237 (km/s)
ASA	:	34.9852	27.7134	20.1603	37.1030	37.0990 (km/s)
GODLIKE:		12.3510	11.4907	12.1768	19.9226	11.4915 (km/s)

Jupiter Uranus (point 7)

GA	:	17.9396	17.4238	14.3493	27.8139	14.4462 (km/s)
DE	:	14.8425	14.5341	26.4409	34.9205	15.9905 (km/s)
PSO	:	91.6886	18.1211	36.3646	37.0455	29.2676 (km/s)
ASA	:	36.0077	36.4350	36.5896	31.0736	35.7007 (km/s)
GODLIKE:		14.3193	14.3207	15.4410	14.3201	14.4783 (km/s)

Jupiter Uranus (point 9)

GA	:	11.1572	12.0051	11.8005	11.8927	11.9693 (km/s)
DE	:	20.0814	12.3912	11.1192	14.4276	22.1150 (km/s)
PSO	:	29.6280	35.0565	27.1181	37.1012	34.9724 (km/s)

```
ASA    : 33.1391   37.1030   36.4105   28.9335   36.9439 (km/s)
GODLIKE: 14.0958   13.3334   13.3334   14.7041   13.4785 (km/s)

================================================================
Jupiter Uranus (point 10)
================================================================
GA     :  8.8657    9.0867    9.3935    9.6443   11.7226 (km/s)
DE     : 37.0475   14.6703   28.1409    8.9619   14.3296 (km/s)
PSO    : 33.4938  109.2080   21.6031   24.9696   30.7888 (km/s)
ASA    : 37.0761   36.5588   37.1003   24.4884   37.1008 (km/s)
GODLIKE: 13.6326    8.7939    8.7667    8.7691    9.3924 (km/s)

================================================================
Jupiter Uranus (point 12)
================================================================
GA     :  9.6546   11.3914   20.5387    9.3631    9.0326 (km/s)
DE     : 20.6609   20.2568    8.9825   20.8331   17.6541 (km/s)
PSO    : 37.1030   18.5989   32.2194   20.3298   30.1490 (km/s)
ASA    : 36.9355   23.2252   16.1820   14.1093   33.2723 (km/s)
GODLIKE:  8.9237    8.9237    8.9509   10.9152    9.1035 (km/s)

================================================================
Jupiter Uranus (point 14)
================================================================
GA     : 18.9453   17.9296   18.2153   17.9637   17.9859 (km/s)
DE     : 17.9726   24.3293   75.6369   18.3085   17.9447 (km/s)
PSO    : 37.0516   21.0206   31.5244   37.0934   31.3896 (km/s)
ASA    : 37.0987   37.1022  573.9857   36.5112   39.3312 (km/s)
GODLIKE: 17.9179   18.5172   17.9128   18.6780   24.2675 (km/s)

================================================================
Jupiter Uranus (point 16)
================================================================
GA     : 26.2052   22.6268   17.0045   16.9673   21.7535 (km/s)
DE     : 20.6776   28.8535   17.1900   33.6390   29.2751 (km/s)
PSO    : 35.5279   36.8696   18.3985   69.9415   35.1496 (km/s)
ASA    : 37.1027   37.0102   36.9671   37.0973   37.0860 (km/s)
GODLIKE: 25.5261   24.7662   25.2846   17.3956   17.1885 (km/s)
```

```
ASA    : 27.4714   37.1030   37.1030   36.3472   31.7732 (km/s)
GODLIKE: 11.8107   19.9991   12.2054   11.1495   17.5807 (km/s)

================================================================
Jupiter Uranus (point 11)
================================================================
GA     :  4.9802   20.6169    5.4227    4.9887    5.3226 (km/s)
DE     :  6.7304    5.4582   20.1981    4.9982    5.9442 (km/s)
PSO    :  8.5090   25.4931   34.6146   35.5761   23.1853 (km/s)
ASA    : 36.2522   37.0492   34.9930   35.2263   10.2772 (km/s)
GODLIKE:  5.0460    5.0289    4.9582   19.0059    4.9538 (km/s)

================================================================
Jupiter Uranus (point 13)
================================================================
GA     : 17.1736   15.6500   14.6739   20.4132   25.6911 (km/s)
DE     : 36.9973   17.4979   14.6407   28.0755   29.1601 (km/s)
PSO    : 36.6095   21.7523   36.7581   50.6349   36.1989 (km/s)
ASA    : 20.0734   37.1030   37.0624   37.0613   34.9757 (km/s)
GODLIKE: 14.6346   14.6346   14.8962   14.6347   16.4731 (km/s)

================================================================
Jupiter Uranus (point 15)
================================================================
GA     : 21.7316   21.7937   18.6387   18.5554   18.6130 (km/s)
DE     : 18.7410   18.8820   20.4548   23.8781   19.0213 (km/s)
PSO    : 32.6135   37.0582   35.5860   35.8446   37.0402 (km/s)
ASA    : 34.3274   69.2175   33.4025   36.4716   69.1406 (km/s)
GODLIKE: 18.5098   18.5303   18.5474   26.1777   36.6624 (km/s)

================================================================
Jupiter Uranus (point 17)
================================================================
GA     : 13.6095   24.6647   14.2877   13.6809   17.9040 (km/s)
DE     : 20.1979   21.4679   16.3011   15.1773   13.6000 (km/s)
PSO    : 36.3406   26.2804   37.1030   37.1030   37.1030 (km/s)
ASA    : 76.0851   37.0034   36.5195   37.1030   34.8875 (km/s)
GODLIKE: 13.5780   19.2960   19.2917   21.7773   13.6005 (km/s)
```

G.4. Low-thrust, First-order, Stagnation Point

```
======================
Jupiter
======================
GA     : 2226.938415 kg
DE     : 2219.173432 kg
PSO    : 2344.041573 kg
ASA    : 2262.493358 kg
GODLIKE: 2216.829420 kg

======================
(Sun) Jupiter
======================
GA     : 2251.832254 kg
DE     : 2036.587551 kg
PSO    : 2211.211021 kg
ASA    : 1979.313064 kg
GODLIKE: 1667.049651 kg

======================
Jupiter Uranus
======================
GA     : 2002.855335 kg
DE     : 1490.341810 kg
PSO    : 1385.452486 kg
ASA    : 1542.594086 kg
GODLIKE: 1904.892668 kg

======================
Mars Earth Jupiter
======================
GA     : 1255.512712 kg
DE     : 1082.151463 kg
PSO    :  642.628800 kg
ASA    : 1443.305911 kg
GODLIKE: 1290.613065 kg

======================
Earth Jupiter
======================
GA     : 1939.856981 kg
DE     : 1920.911818 kg
PSO    : 2258.331933 kg
ASA    : 2051.919334 kg
GODLIKE: 1981.959311 kg

======================
(Sun) Earth Jupiter
======================
GA     : 1940.149780 kg
DE     : 2080.136526 kg
PSO    :  699.039273 kg
ASA    : 1149.523990 kg
GODLIKE:  665.392429 kg

======================
(Sun)
======================
GA     : NaN kg
DE     : NaN kg
PSO    : NaN kg
ASA    : NaN kg
GODLIKE: NaN kg

======================
Earth Earth Jupiter
======================
GA     : 1931.170424 kg
DE     : 2129.231021 kg
PSO    : 1777.076751 kg
ASA    : 1749.996319 kg
GODLIKE: 1918.539523 kg

======================
Venus Jupiter
======================
GA     : 1734.006906 kg
DE     : 1475.504773 kg
PSO    : 1267.593743 kg
ASA    : 1525.475094 kg
GODLIKE: 1335.507671 kg

======================
Jupiter (Sun)
======================
GA     : NaN kg
DE     : NaN kg
PSO    : NaN kg
ASA    : NaN kg
GODLIKE: NaN kg
```

G.5. High-thrust, First-order, Nearby Minor Planets

G.5.1 Earth/Jupiter/Uranus/Bow Shock, $t_f^{\max} = 25.0$ years

```
========================================================================
Number/ Name/desig.   Vrel [km/s]   Dmin [km]     Dmin [AU]   Encounter Epoch
========================================================================
(5052) Nancyruth        24.1423      1.4385e+06      0.0096      Aug 20th, 2021
(5980) 1993 FP2         23.5888      3.7038e+06      0.0248      Aug 29th, 2021
(6745) Nishiyama        18.1976      9.4807e+05      0.0063      Sep 18th, 2021
(8301) Haseyuji         20.8310      1.4451e+06      0.0097      Oct 11th, 2021
(9452) Rogerpeeters     25.9261      2.3889e+06      0.0160      Oct 12th, 2021
(9833) Rilke            22.4675      3.1030e+06      0.0207      Sep 28th, 2021
(14414) 1991 RF6        24.2772      2.4386e+06      0.0163      Aug 17th, 2021
(14607) 1998 SG132      18.2191      2.6745e+06      0.0179      Sep 10th, 2021
(21658) 1999 PA2        18.6801      3.4725e+06      0.0232      Aug 18th, 2021
(36810) 2000 SN69       22.2295      3.3099e+06      0.0221      Oct  7th, 2021
(77949) 2002 GM132      23.2923      2.0292e+06      0.0136      Sep  5th, 2021
```

Number/ Name/desig.	Vrel [km/s]	Dmin [km]	Dmin [AU]	Encounter Epoch
(79527) 1998 OL14	18.2506	2.7326e+06	0.0183	Aug 30th, 2021
(83354) 2001 RB151	20.8898	3.1891e+06	0.0213	Dec 5th, 2021
(109130) 2001 QK51	23.7070	1.8396e+06	0.0123	Oct 1st, 2021
(173776) 2001 SE19	23.4409	2.0371e+06	0.0136	Oct 21st, 2021
(191708) 2004 RQ194	21.6274	2.6008e+06	0.0174	Sep 20th, 2021

G.5.2 Earth/Jupiter/Uranus/Bow Shock, $t_f^{\max} = 22.5$ years

Number/ Name/desig.	Vrel [km/s]	Dmin [km]	Dmin [AU]	Encounter Epoch
(5052) Nancyruth	23.8455	2.7951e+06	0.0187	Aug 20th, 2021
(6745) Nishiyama	17.8406	2.0143e+06	0.0135	Sep 20th, 2021
(8301) Haseyuji	20.4360	1.0268e+06	0.0069	Oct 12th, 2021
(9452) Rogerpeeters	25.5298	1.5162e+06	0.0101	Oct 13th, 2021
(9833) Rilke	22.0931	3.0140e+06	0.0201	Sep 29th, 2021
(14414) 1991 RF6	23.9846	3.4835e+06	0.0233	Aug 17th, 2021
(14607) 1998 SG132	17.8785	1.6582e+06	0.0111	Sep 12th, 2021
(15855) 1996 CP7	18.4319	3.3095e+06	0.0221	Sep 18th, 2021
(36810) 2000 SN69	21.8438	2.7423e+06	0.0183	Oct 8th, 2021
(64265) 2001 TR192	17.7422	3.1963e+06	0.0214	Sep 11th, 2021
(77949) 2002 GM132	22.9620	1.5536e+06	0.0104	Sep 6th, 2021
(79527) 1998 OL14	17.9318	3.5653e+06	0.0238	Aug 31st, 2021

G.5.3 Earth/Jupiter/Uranus/Bow Shock, $t_f^{\max} = 20.0$ years

Number/ Name/desig.	Vrel [km/s]	Dmin [km]	Dmin [AU]	Encounter Epoch
(20415) Amandalu	16.9205	1.7862e+06	0.0119	Sep 5th, 2021
(37598) 1992 EL17	20.9999	3.7208e+06	0.0249	Sep 3rd, 2021
(56485) 2000 GL125	19.1901	3.6854e+06	0.0246	Oct 21st, 2021
(156441) 2002 AU154	22.3218	1.7465e+06	0.0117	Oct 21st, 2021

List of Figures

List of Tables