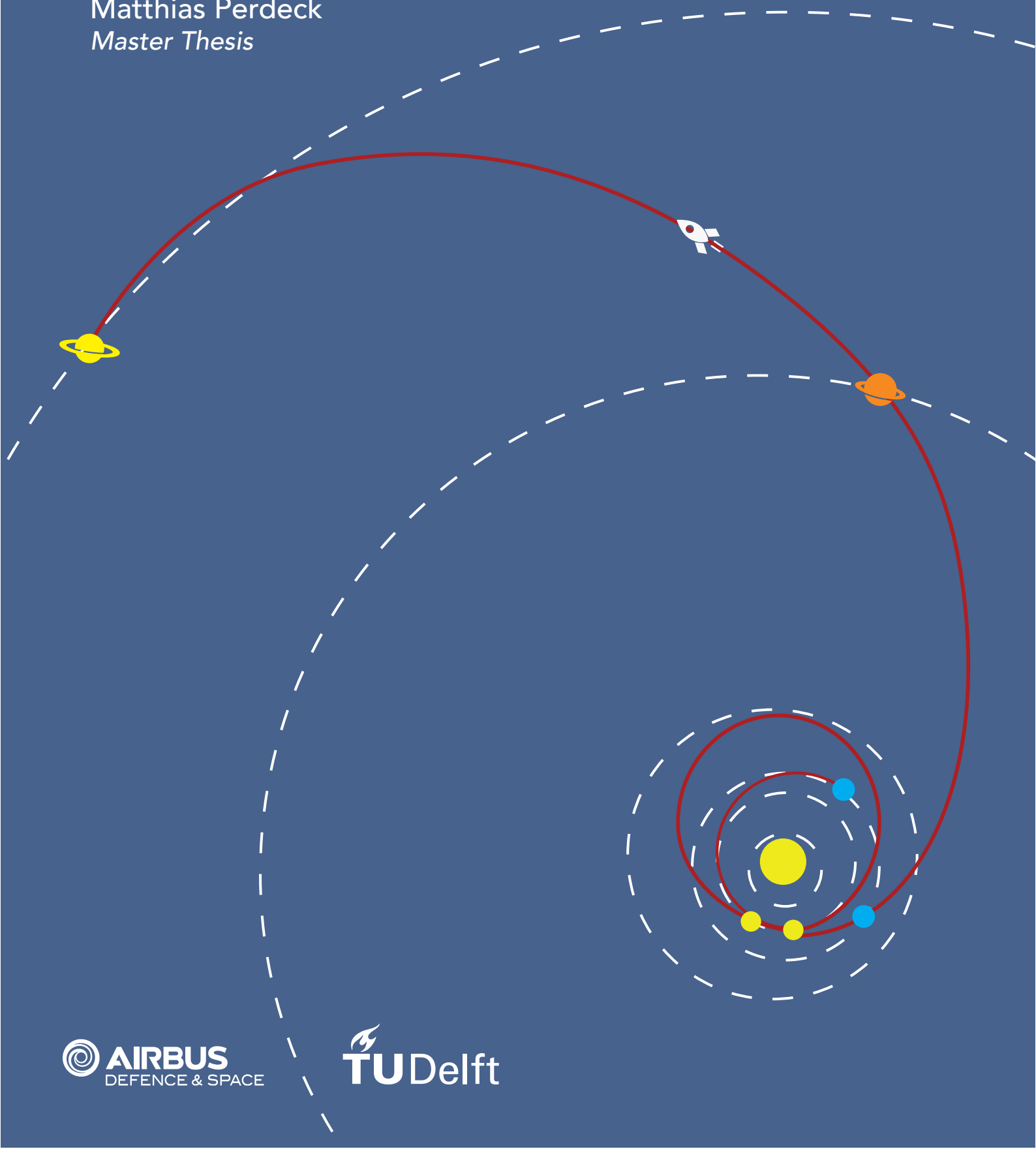
Searching for two optimal trajectories

A study on different approaches to global optimization of gravity-assist trajectories that have a backup departure opportunity

Matthias Perdeck

Master Thesis



AIRBUS DEFENCE & SPACE

TU Delft

Figure on the cover
The figure on the cover of this report displays a trajectory to Saturn, inspired by the Cassini mission. It is the solution to one of the problems that are studied in this research. The planetary orbits and the spacecraft trajectory are to scale (celestial bodies and the spacecraft are not).

Searching for two optimal trajectories

A study on different approaches to global optimization of gravity-assist trajectories that have a backup departure opportunity

Master of Science Thesis

In partial fulfillment of the degree of Master of Science in Aerospace Engineering at Delft University of Technology

M.J. (Matthias) Perdeck

To be defended on 07-06-2018 at 14:00

Assessment committee

Prof. dr. ir. P.N.A.M. Visser	Chair
K.J. Cowan, MSc MBA	Responsible thesis supervisor
Dr. A. Menicucci	Examinar

This thesis is confidential and under embargo until 07-06-2023,
after which an electronic version will be available at http://repository.tudelft.nl

This page is intentionally left blank.

Preface

The initiative for this study originates from the attitude and orbital control systems/guidance, navigation and control department of Airbus Defence and Space in Friedrichshafen (Germany). The consequent thesis research for the degree of Master of Science in Aerospace Engineering at Delft University of Technology has been executed on site in Friedrichshafen between September 2017 and May 2018.

This report is the symbiosis of eight months of research on the fascinating topic of interplanetary spacecraft trajectories. It connects physics, mathematics and computer science. While the former two have formed the core of my curriculum for several years, this thesis has kindled a new found enthusiasm for the rigorous logic of the latter.

I would like to thank Razvan Luzi and Yannick Enginger for their guidance during my stay at Airbus. Their continued interest in the research and relaxed approach have created a productive and pleasant working atmosphere. The periodic teleconferences with Kevin Cowan from Delft University of Technology have helped determining the coherence of the various fields of science that meet in this research. A comprehensive list of scientists and engineers that have been contacted over the course of this study is provided in Appendix E.

Matthias Perdeck
May 2018

Summary

Departure epoch $T0$ of an interplanetary space mission can slip due to various unforeseen reasons. The consequences can be mitigated by having a backup departure opportunity at a time $\Delta T0$ after the first possibility (21 days in this research). The design of a trajectory is usually approached as a minimization of the velocity (ΔV) budget; designing a trajectory with a backup (a trajectory pair) is then a minimization of the maximum ΔV of the pair (ΔV_{rb}). In this study, two different approaches to this problem (the a priori and the a posteriori approaches, explained below) are compared on their performance (computational efficiency[1], accuracy) and algorithm characteristics (complexity, influence of design parameters, versatility[2]). High-thrust trajectories are considered, they are modeled using the linked-conics approximation and include gravity-assists and (optionally) deep space maneuvers. Differential evolution is used for optimization.

The a priori approach minimizes ΔV_{rb} as a function of the control variables of both trajectories, given a $\Delta T0$. Since two trajectories with close departure epochs are likely to be similar, three (mutually exclusive) methods that exploit this similarity have been devised: 1) Symmetric initialization biases optimization by initializing paired trajectories equally. 2) The variable mirror operator biases optimization (in the optimization loop) by overwriting the control variables of an unfit trajectory with those of a fit paired trajectory. 3) The narrow relative constraints algorithm prunes the solution space by constraining the difference between the control variables of both trajectories.

The a posteriori approach first computes the minimum ΔV for a range of departure epochs by solving minimization problems for discrete departure epoch values. To increase computational efficiency, the solution of a previous departure epoch is used as initial guess for a following departure epoch. The pair with the lowest ΔV_{rb} is then selected from the minimized ΔV values. Two variants are proposed, one on the full $T0$ range (1000 days) and one on a limited range of $\pm\Delta T0$ around the global minimum (it prunes the other 958 days).

By applying the methods to three by Cassini inspired trajectories to Saturn, the following conclusions are drawn. The performance of the a posteriori approach scales better with an increasing dimensionality D, than the a priori approach. On the simplest trajectory ($D = 3$), the former requires ten times more function evaluations than the latter, but on the most difficult trajectory ($D = 22$), it requires five times fewer function evaluations than the latter. Between the variants of the a priori method, the variable mirror and narrow relative constraints algorithms perform best. The performance of the first is determined by a design parameter, while the second requires the user to formulate the boundaries. Regarding the a posteriori approach, the limited range method can be highly efficient. Compared to optimizing a single trajectory, it requires just 3% more function evaluations to optimize a pair, on the difficult trajectory. However, it did not find the optimal solution of the simple trajectory because it was outside its range. Furthermore, the accuracy of the a posteriori approach is limited at 0.5 m/s by the chosen resolution of $T0$ (0.1 day), the a priori approach does not have such a limit.

The algorithm review leads to the conclusion that the a posteriori approach is more complex than the a priori approach due to its initial guess algorithm. Furthermore, symmetric initialization is the only algorithm that does not rely on design parameters. Regarding versatility, the array of ΔV values computed by the a posteriori approach has the advantage that it can be used for a general launch window analysis. Lastly, the a posteriori approach has been adapted most to the considered problems, therefore it is likely to be less generally applicable.

[1] Number of objective function evaluations required to reach a specified objective function value.
[2] Applicability to other purposes than minimizing a trajectory pair.

Table of contents

Preface v

Summary vi

List of symbols xii

List of definitions xiv

1 Introduction 1
- 1.1 Purpose . . . 2
  - 1.1.1 Scope . . . 2
  - 1.1.2 Research questions . . . 3
- 1.2 Structure . . . 3

2 Theoretical context 5
- 2.1 Historical background . . . 5
- 2.2 Physical models . . . 6
  - 2.2.1 Orbit propagation . . . 6
  - 2.2.2 Conic approximations . . . 6
  - 2.2.3 Comprehensive trajectory models . . . 8
- 2.3 Optimization techniques . . . 9
  - 2.3.1 Evolutionary algorithms . . . 10
  - 2.3.2 Social behavior algorithms . . . 11
  - 2.3.3 Repeated local searches . . . 12
  - 2.3.4 Other optimizers . . . 12
- 2.4 Sensitivity analyses . . . 12
  - 2.4.1 Pork chop plots . . . 12
  - 2.4.2 Sensitivity analyses . . . 13
  - 2.4.3 Python EMTG automated trade study application . . . 14

3 Methodology 15
- 3.1 Definitions . . . 15
  - 3.1.1 Definition of robustness with respect to departure epoch . . . 15
  - 3.1.2 Definitions of the a posteriori and a priori approaches . . . 16
- 3.2 Use of existing methods . . . 17
  - 3.2.1 Use of MGA and MGADSM trajectory models . . . 17
  - 3.2.2 Use of differential evolution . . . 17
  - 3.2.3 Use of MATLAB and C++ programming languages . . . 18
- 3.3 Development of new algorithms . . . 18
- 3.4 Algorithm verification . . . 18

3.5 Assessment of results . . . 18
3.5.1 Performance . . . 18
3.5.2 Algorithm review . . . 20
3.6 Application to the Cassini mission . . . 20

4 Trajectory models **22**
4.1 Two-body problem propagators . . . 22
4.1.1 Kepler propagation . . . 22
4.1.2 Lambert propagation . . . 23
4.2 Trajectory events . . . 23
4.2.1 Departure . . . 24
4.2.2 Deep space maneuvers (MGADSM only) . . . 24
4.2.3 Gravity-assist maneuvers . . . 26
4.2.4 Arrival . . . 27
4.3 Application to the trajectory of Cassini . . . 28
4.3.1 Cassini0 MGA trajectory . . . 28
4.3.2 Cassini1 MGA trajectory . . . 30
4.3.3 Cassini2 MGADSM trajectory . . . 30
4.3.4 Comparison with the actual trajectory . . . 30

5 Objective functions **35**
5.1 A posteriori problem analysis . . . 35
5.1.1 Computing the optimal ΔV budget curve . . . 36
5.1.2 Determining the minimum robust trajectory pair . . . 36
5.2 A priori problem analysis . . . 38
5.2.1 Absolute decision vector formulation . . . 38
5.2.2 Relative decision vector formulation . . . 40
5.3 Conclusion . . . 40

6 Optimization **43**
6.1 Differential evolution . . . 43
6.1.1 Algorithm description . . . 43
6.1.2 Constraint handling . . . 45
6.1.3 Stopping criteria . . . 46
6.1.4 Verification of the differential evolution algorithm . . . 48
6.2 Performance on the objective functions . . . 49
6.2.1 Experimental set-up . . . 49
6.2.2 Optimization results and discussion . . . 50
6.3 Conclusion . . . 51

7 Pruning and biasing **52**
7.1 A posteriori approach . . . 52
7.1.1 Generating an initial guess (biasing) . . . 52
7.1.2 Limited departure epoch range (pruning) . . . 59
7.2 A priori approach . . . 60
7.2.1 Symmetric initialization (biasing) . . . 60
7.2.2 Variable mirror algorithm (biasing) . . . 61
7.2.3 Narrow relative boundary constraints (pruning) . . . 66

8 Results **70**
8.1 Results of the a posteriori approach . . . 70
8.1.1 Visualization of the optimal ΔV budget curve . . . 70
8.1.2 Accuracy . . . 73
8.1.3 Computational efficiency . . . 73
8.1.4 Algorithm review . . . 77
8.1.5 Conclusion a posteriori results . . . 78
8.2 Results of the a priori method . . . 79
8.2.1 Accuracy . . . 79
8.2.2 Computational efficiency . . . 79
8.2.3 Algorithm review . . . 82
8.2.4 Conclusion a priori results . . . 83

9 Conclusion **85**
9.1 Comparison between both methods . . . 85
9.1.1 Performance . . . 85
9.1.2 Algorithm review . . . 88
9.1.3 Final word on the comparison . . . 88
9.2 Recommendations for further research . . . 89

Bibliography **89**

Appendices **95**

A Solution space analysis **96**

B Monotonic basin hopping **102**
B.1 Algorithm description . . . 102
B.1.1 Local optimization approach . . . 103
B.1.2 Perturbing operator . . . 103
B.2 Constraints . . . 105

C Verification functions for differential evolution **106**
C.1 Rosenbrock saddle . . . 106
C.2 Rastrigin function . . . 106

D Mirror threshold research Cassini0 **107**

E Contacted experts **109**

List of symbols

Greek symbols

α_{LA}	Right ascension of the launch asymptote	radians
β	Angle in the Lambert solver	radians
δ	Hyperbolic transfer angle	radians
δ_{LA}	Declination of the launch asymptote	radians
ΔT	Relative time of flight	days
$\Delta T0$	Departure epoch interval	days
$\delta T0$	Departure epoch step size	days
ΔV	ΔV budget	km/s
$\Delta V0$	Departure impulse	km/s
ΔV_1	ΔV budget of the first trajectory	km/s
ΔV_2	ΔV budget of the second trajectory	km/s
ΔV_{best}	Best attained ΔV budget	km/s
ΔV_{GAM}	Magnitude of the perigee impulse in a GAM	km/s
ΔV_{min}	ΔV budget of the global minimum of a trajectory function	km/s
ΔV_{opt}	Optimal ΔV budget	km/s
ΔV_{rb}	ΔV budget of the robust trajectory pair	km/s
η	DSM ratio	-
γ	b-plane insertion angle	radians
μ	Gravitational parameter	km^3/s^2
Ω	Right ascension of the ascending node	radians
ω	Argument of the perigee	radians
ϕ	Angle in the Lambert solver	radians
ψ	Angle in the Lambert solver	radians
σ_p	Standard deviation of the success rate	-

Roman symbols

a	Semi-major axis	km
$\mathbf{B}$	Vector through the b-plane to the incoming asymptote	km
$\mathbf{b}$	Best member of the population	km
$\mathbf{c}$	Competing member vector	-
CR	Crossover ratio	-
D	Number of trajectory control variables	-
d	Dimensionless distance in the solution space	-
D_{re}	Number of relative optimization variables	-
E	Eccentric anomaly	radians
e	Eccentricity	-
$\mathbf{e}$	Random member	-
F	Scaling factor	-

FT	Fitness threshold	km/s
$\mathbf{f}$	Random member	-
$\mathbf{g}$	Random member	-
$\mathbf{h}$	Random member	-
i	Inclination	radians
$\hat{i}$	Unit vector of the b-plane	-
$\hat{j}$	Unit vector of the b-plane	-
$\hat{k}$	Unit vector of the b-plane	-
$\mathbf{m}$	Mutant member	-
M	Mean anomaly	radians
$\mathbf{M}$	Solution matrix of the initial guess algorithm	-
MR	Mirror ratio	-
FT	Maximum number of stalled generations	-
MT	Mirror threshold	-
N	Number of optimization variables	-
$N95$	Performance indicator	-
n_{95}	Performance measure used by Musegaas (2012)	-
N_c	Number of combinations of members	-
N_{eps}	Number of entry points	-
N_{fe}	Number of function evaluations in a run	-
N_{GAM}	Number of gravity-assist maneuvers	-
N_{mirror}	Number of mirror operations	-
N_P	Population size	-
N_s	Number of entry point sections	-
N_{dps}	Number of departure epochs	-
N_T	Total number of functions evaluated by the a posteriori approach	-
$\mathbf{P}$	Population	-
$\mathbf{Q}$	Average distance calculation matrix	-
$\mathbf{q}$	Boundary constraint	-
R	Perigee to planetary radius ratio	-
r	Dimensionless radial distance	-
r_p	Perigee radius	km
$\mathbf{R}$	Average distance calculation matrix	-
$\mathbf{S}$	Average distance calculation matrix	-
T	Time of flight	days
$\mathbf{T0}$	Departure epoch vector	MJD2000
$T0$	Departure epoch	MJD2000
$T0_1$	Departure epoch of the first trajectory	MJD2000
$\mathbf{T0_1}$	Vector with departure epochs of the first trajectory	MJD2000
$T0_2$	Departure epoch of the second trajectory	MJD2000
V	Dimensionless volume	-
V_∞	Generic hyperbolic excess velocity	km/s
$\mathbf{V}_{+\infty,\mathbf{P}}$	Vector of the outgoing hyperbolic excess velocity	km/s
$\mathbf{V}_{-\infty,\mathbf{P}}$	Vector of the incoming hyperbolic excess velocity	km/s
$V_{+\infty,P}$	Magnitude of the outgoing hyperbolic excess velocity	km/s
$V_{-\infty,P}$	Magnitude of the incoming hyperbolic excess velocity	km/s
$\mathbf{V}_{\mathbf{P},\mathbf{H}}$	Planetary velocity vector in the heliocentric frame	km/s
VTR	Objective function value-to-reach	km/s

k	Optimization parameter in the Lambert solver	-
x	Cartesian x-coordinate	km
$\mathbf{x_1}$	Decision vector of the first trajectory	various
$\mathbf{x_2}$	Decision vector of the second trajectory	various
$\mathbf{x_{ab}}$	Decision vector of the a priori absolute objective function	various
$\mathbf{x_{pt}}$	Decision vector of the a posteriori approach	various
$\mathbf{x_{re}}$	Decision vector of the a priori relative objective function	various
y	Cartesian y-coordinate	km
z	Cartesian z-coordinate	km

List of definitions

The **absolute objective function** is an objective function used in the a priori approach and takes the absolute values of the control variables of both trajectories of the robust trajectory pair as input, except for $T0_2$. Section 5.2.1.

The **absolute solution space** is the multidimensional space defined by the absolute control variables. Section 5.2.2.

The **a posteriori approach** first computes minimum ΔV budget ΔV_{opt} for a range of departure epochs $T0$. Thereafter, robust ΔV budget ΔV_{rb} is obtained from pairs of ΔV_{opt} values that are departure epoch interval $\Delta T0$ apart. Section 5.1.

An **a posteriori objective function** is a trajectory function with the departure epoch fixed. Section 5.1.

The **a priori approach** minimizes robust ΔV budget ΔV_{rb} directly as a function of the control variables of both trajectories (except second departure epoch $T0_2$), given a departure epoch interval $\Delta T0$. Section 5.2.

An **a priori objective function** calculates the ΔV budget of the robust trajectory pair as a function of the control variables of the two trajectories (except second departure epoch $T0_2$). Section 5.2.

A **basin** contains all the points in a multidimensional space, from which a (local) minimum can be reached via a continuously (monotonically) descending path. Chapter 2.3.3.

Biasing mechanisms exploit similarity between the paired trajectories, by steering the optimizer towards solutions with similar values for the control variables of both trajectories. Chapter 7.

Computational efficiency is the required computational effort, measured in the number of trajectory function evaluations, for obtaining a specified objective function value. Section 3.5.

Control variables are the input variables of the trajectory function, they fully determine the trajectory. Together, they span the solution space of the trajectory function. Section 4.2.

A **decision vector** contains the optimization variables of a problem. Section 4.2.

The **departure epoch interval** is the interval between the departure epochs of two trajectories of a robust trajectory pair ($\Delta T0$). Section 3.1.

Design parameters influence the performance of an optimization algorithm. They need to be set (or tuned) by the user. Section 2.3.

Dimensionality is the number of optimization variables. It is represented by N in the general case and by D for a trajectory function. Table 5.2.

A **deep space maneuver (DSM)** is a short period of thrust during interplanetary coasting. Section 4.2.2.

Entry points are the departure epochs from which ΔV_{opt} is propagated in the a posteriori approach including the initial guess algorithm. Section 7.1.1.

A **gravity assist maneuver (GAM)** is a planetary flyby that is used to reduce he ΔV budget of an interplanetary trajectory. Section 4.2.3.

The **global minimum basin** is the basin that contains the global minimum of the trajectory function. Figure 3.1.

The **linear extrapolation strategy** is part of the initial guess algorithm of the a posteriori approach. It extrapolates the solution of two neighboring departure epochs linearly, to generate an initial guess for the current epoch. Section 7.1.1.

MJD2000 is short for Modified Julian Dates 2000. It is a time model with Julian days as units and is zero on January first 2000 at 12:00 midday (McCarthy and Hoskin 1998). Section 4.2.

The **narrow (relative boundary) constraints algorithm** is a pruning variant of the a priori approach. It imposes narrow boundary constraints on the relative optimization variables. Section 7.2.3.

The **optimal ΔV budget curve** shows the optimal (minimal) ΔV budget for a range of departure epochs ($\Delta V_{opt}(T0)$). Section 3.1.

Optimization variables are the variables that are to be optimized. When optimizing the trajectory function, the optimization variables are the control variables. Chapter 5.

Performance indicator $N95$ is used to quantify the computational efficiency of the a priori approach (see Section 3.5).

The **previous solution strategy** is part of the initial guess generator of the a posteriori approach. It uses the solution of a neighboring departure epoch as initial guess for the current departure epoch. Section 7.1.1.

Pruning is defined as reducing the size of the solution space by excluding regions from the search, that are not likely to contain the solution. Chapter 7.

The **relative objective function** is an objective function used in the a priori approach. It defines the control variables of the second trajectory of the robust trajectory pair, relative to those of the first, using the relative optimization variables. 5.2.2.

The **relative optimization variables** define the control variables of the second trajectory relative to the control variables of the first trajectory. Section 5.2.2.

The **relative solution space** is the multidimensional space defined by the relative optimization variables. Section 5.2.2.

A **(robust) trajectory pair** consists two trajectories with a user-specified interval $\Delta T0$ between their departure epochs. ΔV budget ΔV_{rb} of the robust trajectory pair is the highest of the ΔV budgets of the two trajectories. Furthermore, the minimum robust (trajectory) pair has the global minimum ΔV_{rb} value. Section 3.1.

Symmetric initialization is a biasing variant of the a a priori approach. Corresponding control variables of the two trajectories are initialized with the same (random) values. Section 7.2.1.

A **trajectory function** takes control variables as input and produces the ΔV budget of the corresponding trajectory. Chapter 4.

The **variable mirror algorithm** is a biasing variant of the a priori approach. If the fitness ratio between the two trajectories exceeds a treshold, the mirror operator overwrites the control variables of the unfit trajectory with the values of those of the fit trajectory. Section 7.2.2.

Versatility is defined as applicability beyond the scope of the problems considered in this research. Section 3.1.

Chapter 1

Introduction

Designing propellant-efficient interplanetary spacecraft trajectories is one of the key challenges of space exploration. The vast distances, multitude of possibilities, and large relative velocities between planets push our engineering capabilities to their limit. Generally, trajectory design is approached as an optimization problem, aimed at minimizing the total velocity change (ΔV) during the journey. According to Tsiolkovsky (1903), this guarantees to minimize propellant mass,[1] which in turn reduces complexity and costs. Depending on the mission characteristics, other optimization objectives (such as the time of flight) may also be formulated, but minimizing ΔV forms the core of any trajectory design.

An ingenious technique for reducing ΔV has been found in the application of gravity-assist maneuvers (GAMs). These hyperbolic flybys of third bodies (usually planets) can result in considerable savings of propellant (Minovitch 1963). Together with short periods of thrust during interplanetary coasting (deep space maneuvers or DSMs), they form a powerful combination that has opened up the entire solar system for exploration. Nowadays, GAMs and DSMs are ubiquitous in deep space travel.

A consequence of the application of GAMs and/or DSMs is that the optimization problem becomes more difficult. A specifically challenging set of trajectory optimization problems is encountered in the early conceptual stage of mission design. Here, preliminary options are analyzed and compared, to be potentially further developed into mission concepts. The difficulty is that these trajectories are obtained by solving a global optimization problem, contrary to trajectories of higher fidelity that are obtained through local optimization of a reasonably accurate initial guess. Some complicating factors associated with global trajectory optimization include the following.

- The solution space spans many dimensions, more than twenty is not uncommon.
- The solution space has many local optima.
- Boundary constraints can be nonlinear.
- Expressions may lack a closed form.
- The target and destination bodies' states are functions of time.

To deal with these issues, current methods rely on metaheuristic optimization algorithms.[2] How-

[1] Valid for engines that expel mass.

[2] Metaheuristic optimization algorithms approximate the global minimum solution to a satisfactory degree. They do so by letting the optimization process be (partially) guided by randomly generated variables. Several different metaheuristic optimization techniques are discussed in Chapter 2.

ever, a typical problem solved by NASA's Evolutionary Mission Trajectory Generator, an open-source trajectory optimization tool, may still take several days to compute (Englander 2013).

Besides making it difficult to identify the global minimum in the first place, the issues stated above also make that this global minimum is often not robust.[3] This is an undesirable property, since control variables require a margin for safe and reliable operations. A variable that is specifically prone to uncertainty is departure epoch $T0$. Various reasons may lead to delays, including bad (space) weather, failed tests, programmatic delays, and budget cuts (Biesbroek 2016). It is argued that one can deal with this uncertainty by providing a guaranteed backup departure opportunity for each trajectory. Then, the spacecraft should be designed for the highest ΔV of the two trajectories. The minimization of the maximum ΔV of these two trajectories is the fundamental problem that is addressed in this thesis.

A global optimization algorithm that is capable of finding two such trajectories (a robust trajectory pair) is Python EMTG Automated Trade Study Application (PEATSA) developed by Knittel et al. (2017). This software works as an outer shell around the earlier mentioned EMTG; a series of optimizations yields the minimum ΔV budget for a range of departure epochs (optimal ΔV budget curve). Consequently, the user may select the trajectories that fulfill the requirements for robustness. Remembering the computation times of EMTG, this seems a resource-expensive exercise. In this study, such a method is classified as a posteriori, because it first computes the optimal ΔV budget curve and afterwards lets user select a trajectory pair. Alternatively, the a priori approach has been devised. Here, the user needs to specify the interval between the departure epochs (departure epoch interval $\Delta T0$) beforehand, allowing the control variables of both trajectories to be simultaneously optimized. The a priori approach is hypothesized to require fewer objective function evaluations than the a posteriori approach.

It has been found that departure epoch uncertainty is a largely uncharted field within global optimization of gravity-assist trajectories. No methods capable of selecting robust gravity-assist trajectories, other than PEATSA, have been identified. Because of this blind spot, a specific need for more research has been formulated by Airbus Defence and Space, the commissioner of this research. Taking departure uncertainty into account from an early stage leads to better substantiated decisions in their mission design process. It is therefore desired to gain more knowledge on the different methods that can be used for minimizing the ΔV budget of a robust trajectory pair.

1.1 Purpose

The purpose of this research is to gain knowledge on the relative performance of the a posteriori and a priori approaches[4] to optimizing robust trajectory pairs. To that end, both methods are first developed, then implemented and consequently tested. The following scope and research questions provide a framework for this process.

1.1.1 Scope

A first demarcation is the exclusive focus on high-thrust trajectories. That is, only chemical and thermal rocket propulsion are considered, as opposed to long duration thrust from, for example,

[3] Generally defined as being able to avoid large changes in the output for small changes in the input. A formal definition is provided Chapter 3.

[4] They are mentioned in this (counterintuitive) order because the a posteriori approach refers to an existing method, while the a priori approach is newly proposed.

ion engines. This is at the explicit request of Airbus Defence and Space since the majority of proposals that it receives, is for high-thrust missions. Consequently, two high-thrust trajectory models (developed by ESA's Advanced Concepts Team) are used. Furthermore, optimization is done using a metaheuristic optimizer named differential evolution, which has demonstrated its capability to optimize these trajectory models.

The developed methods are applied to three different problems, all inspired by the Cassini mission to the Saturnian system. The trajectory of Cassini involved four GAMs (Venus-Venus-Earth-Jupiter), leading to a highly efficient ΔV budget of 5.3 km/s (Englander 2013). Much literature is available on the performance of the aforementioned model and optimizer on this trajectory, making it suitable for verifying the algorithms developed in this work.

1.1.2 Research questions

The a posteriori and a priori approaches that are described above demand further research before they are sufficiently specified to be implemented. The following two research questions address two fields of this development.

Development-oriented research questions

1. Into which sub optimization problems can the a posteriori and a priori approaches be decomposed, and what are the associated objective functions?
2. Which solution space pruning[5] techniques and optimizer biasing mechanism can be used to improve computational efficiency,[6] and what are their working principles?

For the second research question, a limited number of five algorithms is developed, leading to different variants of the a posteriori and a priori methods. These variants are sought to be diverse. Having developed and implemented the algorithms, the final three research questions address their performance and characteristics.

Results-oriented research questions

3. How accurate and computationally efficient are the a posteriori and a priori approaches?
4. What are the characteristics of the a posteriori and a priori approaches in terms of algorithm complexity, influence of design parameters, and versatility?
5. How do the results of the a posteriori and a priori approaches compare on the aforementioned fields (4 and 5)?

In the last research question, versatility is defined as applicability beyond the scope of the problems of this research. More definitions are stated in the List of definitions.

1.2 Structure

The structure of this report is as follows. In Chapter 2, a survey of literature on global optimization of gravity-assist trajectories is presented. Also, three concepts that are related to departure epoch uncertainty are discussed. In Chapter 3, the methodology of this research is laid out. This includes

[5] Reducing the size solution space by excluding regions from the search, that are not likely to contain the solution (see Chapter 7).

[6] The number of objective function evaluations that is required to obtain a certain result (see Section 3.5).

formal definitions of the a posteriori and a priori approaches. The two trajectory models that are used in this research are explained in Chapter 4, which is concluded with the characteristics of the three trajectory problems to which the methods are applied. Thereafter, in Chapter 5, an answer to the first research question is presented, through a detailed analysis of both methods. It is followed by Chapter 6, in which the working principle of differential evolution is explained. Also, appropriate settings of the design parameters that determine differential evolution's performance are determined. Consecutively, two pruning techniques and three biasing mechanisms are proposed in Chapter 7, answering the second research question. Finally, answers to the third and fourth research question are found in Chapter 8. Here, the results are presented and discussed. Finally, a conclusion on the comparison between both methods is drawn in Chapter 9. This last chapter ends with suggestions for future research.

Chapter 2

Theoretical context

Global optimization of gravity-assist trajectories is a much researched topic. In this chapter, the reader is guided through the developments in this field, that are most relevant to this research. First, the historical context of global trajectory optimization is (briefly) outlined in Section 2.1. Next, an overview of the current state-of-the art methods is split between the trajectory models, explained in Section 2.2, and the optimization methods, explained in 2.3. Contrary to general trajectory optimization, the effect of departure epoch uncertainty on interplanetary mission design is much less researched. Three relevant concepts have been identified in this field, they are addressed in Section 2.4.

2.1 Historical background

During the early years of spaceflight, mission analysts had to generate a limited number of reasonably accurate suggestions for trajectories, which would then be further optimized using (quasi-) Newton optimization or optimal control theory. The large number of possible flyby sequences, and many parameters that determine a trajectory made generating these initial guesses a challenging task. This issue could be dealt with when, towards the end of the 1980s, computing power had progressed sufficiently to enable optimization algorithms to find optimal trajectories without an initial guess.

The first algorithm that was specifically aimed at global optimization of interplanetary trajectories was the Satellite-Tour Design Program (S-TOUR). This software was originally developed at the Jet Propulsion Laboratory (JPL) for the Galileo mission to the Jovian system[1] and has later been transformed into a general trajectory optimization algorithm at Purdue University (Longuski and Williams 1990). It made use of a grid search method for finding the control variable values. Combined with a graphical approach to identifying possible flyby sequences, S-TOUR formed a chain that was capable of finding an optimal trajectory including the flyby sequence (Strange and Longuski 2002). However, unsurprisingly for a grid search approach, the computation times quickly became intractable for more complex missions.

Since the development of S-TOUR, computation times have been brought down by a combination of further increasing computing power and more intelligent algorithms. Specifically the application of meta-heuristic optimization algorithms has proven to be a lasting paradigm change (Vasile and Pascale 2006; Izzo et al. 2007; Conway 2010). Current methods make use of various different

[1] Galileo departed in 1989 and arrived at Jupiter in 1995 after a Venus-Earth-Earth flyby sequence (Meltzer 2013).

non-deterministic algorithms of which the computation time does not scale exponentially with the dimensionality of the solution space (Vasile, Minisci, and Locatelli 2010).

2.2 Physical models

Before there is zoomed in on the state-of-the-art optimizers, the trajectory models that are the objective of optimization are explained. First, a high-level distinction between analytical and numerical propagation is pointed out in Section 2.2.1. Next, the linked- and patched-conics approximations are explained in Section 2.2.2. This lays the foundation for the analysis of four comprehensive trajectory, presented in Section 2.2.3.

2.2.1 Orbit propagation

The state (position and velocity) of a spacecraft can be propagated (in time) using several techniques. Below, first a distinction between analytical and numerical optimization is pointed out, then two different types of analytical propagation are explained.

Analytical versus numerical propagation

Arguably the most high-level distinction that can be made in orbit propagation, is between an analytical and numerical approach (Vallado 2006). The former requires one to make use of the two-body approximation, thus neglect (gravitational) perturbations. Consequently, the state can be propagated using the computationally inexpensive Lambert and Kepler methods.[2] Contrary, numerical propagation allows any number of forces to be modeled with an accuracy that is inversely related to the propagation step size, leading to potentially very accurate models. However, high accuracy comes at the cost of a long computation time. Current limits on computing power make numerical propagation too expensive to be used in global optimization of trajectories that include multiple GAMs.

Two-body orbit propagation

Two techniques for propagating a two-body orbit are the aforementioned Kepler and Lambert methods. The former propagates the state of a spacecraft given a time of flight. Lambert's problem is concerned with finding the two velocities given two positions and the time of flight. Both Kepler and Lambert propagation are used by the trajectory models that are used in this research, the implementations are explained in Section 4.1. Figure 2.1 shows the inputs and outputs of Kepler and Lambert propagation.

2.2.2 Conic approximations

A trajectory that includes GAMs can be simplified by approximating it as a series of two-body problems. The following two approximations are based in this simplification, they allow for fast propagation of complex trajectories.

Patched-conics approximation

The patched-conics approximation models the trajectory as a planetocentric orbit within the sphere of influence of a planet, and as a heliocentric orbit outside it. This renders the entire trajectory a

[2] The reader may realize that Kepler and Lambert propagation require iterations, thus also exhibit some numerical behavior (see Section 4.1). However, it is common practice to refer to them as analytical methods, this convention adopted in this study.

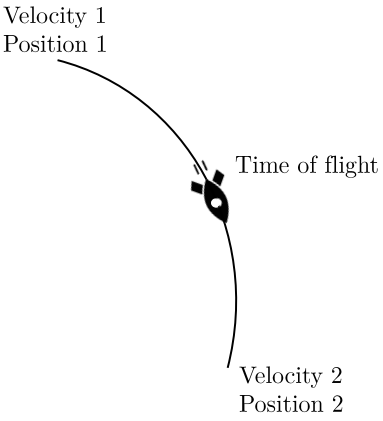


Figure 2.1: An arc (orbital section modeled as a two-body problem) with two states and a time of flight. Kepler propagation calculates a position and a velocity, given the other state and the time of flight. Lambert propagation calculates two velocities given the positions and time of flight.

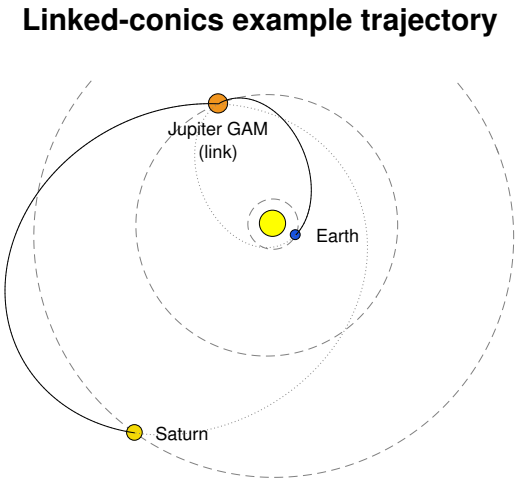


Figure 2.2: Example of a simple linked-conics trajectory transcription. It represents a mission to Saturn with one GAM at Jupiter, the elliptic arcs are 'linked' here. The full ellipses of the orbits between the planets are indicated by dotted lines.

series of patched planetocentric and heliocentric conic sections, hence the name. Both the position and velocity of the spacecraft are continuous in time.

Linked-conics approximation

The spheres of influence of planets are small compared to the semi-major axes of their orbits around the sun. The highest ratio is reached by Jupiter, the radius of its sphere of influence is 6.2% of its semi-major axis (Curtis 2013). Therefore, an additional simplification may be applied: neglecting the radii of the spheres of influence. The result is that the velocity changes due to GAMs are instantaneous, hence they show discontinuities at planetary flybys. The resultant model consists only of heliocentric conic sections and can be propagated faster than the patched-conics model, at the expense of some accuracy (Conway 2010). Figure 2.2 shows an example of a trajectory to Saturn with one GAM at Jupiter, modeled using the linked-conics approximation.

2.2.3 Comprehensive trajectory models

Using the techniques described above, it is possible to formulate comprehensive models of gravity-assist trajectories. Four different models have been identified in literature, all use the linked-conics approximation. They are explained and analyzed below.

Multiple gravity-assist model

The multiple gravity-assist (MGA) model allows for a burn at the perigee of each planetary flyby, as well as at arrival and departure. A GAM that includes a burn is also known as an Oberth maneuver. Applying thrust at the perigee of an orbit is most propellant efficient, since velocity increments at higher velocity yield a larger energy increase (Oberth 1929).

Since the positions of planets are known as a function of time,[3] taking the departure epoch and times of flight as control variables allows each arc between two planets to be solved as a Lambert problem. A clear drawback of this model is that it prohibits DSMs. This limits the types of trajectories that can be modeled. For example, it rules out a V_∞-leveraging maneuver[4] as used in the Cassini mission.

The number of control variables in this model is limited; just one per planetary encounter (including departure and arrival), thus dimensionality D of the solution space is equal to the planetary sequence length (including the departure and arrival planets). This makes the MGA model easiest to optimize of the ones considered here, for a given flyby sequence. Implementations of the MGA model are available for download in several different programming languages from ESA's Advanced Concepts Team web page,[5] where some test case trajectories are also available (Izzo 2010).

Multiple gravity-assist and deep space maneuver model

The multiple gravity-assist and deep space maneuver (MGADSM) model includes one DSM between each planetary transfer, which makes it more flexible than the MGA model. As explained above, a velocity discontinuity occurs at each GAM and at each DSM. A fundamental difference between these maneuvers is that the location of a DSM is not known as a function of time, contrary to the location of a GAM. Therefore, Kepler's problem is solved on the arc from a planet to a DSM. Consecutively, the arc from the DSM to the next planet can be solved using Lambert's method. The GAMs are ballistic, but it is still capable of modeling a much wider variety of trajectories than the MGA model, due to the availability of DSMs (Izzo 2010; Englander 2013).

The number of control variables is significantly higher than for in MGA model, it increases by four with each leg (trajectory section between two planets). The identity of the control variables is explained in Chapter 4. Also, this model is sensitive to small parameter changes, making optimization of this model relatively difficult. Implementations are available at the same source[5] as the MGA model, including several test cases (Izzo 2010).

Multiple Lambert model

As an alternative to the trajectory formulations described above, another formulation for the MGA and MGADSM model has been proposed by Brennan (2015). This formulation only uses Lambert

[3] In the implementation of this model that has been made available by Izzo (2010), the planetary orbital elements are estimated using a fifth-order polynomial least squares fit.

[4] In these maneuvers, a relatively small DSM some time before a GAM enables a high ΔV increase in the maneuver; the effect of the GAM is leveraged (Hollenbeck 1975; Strange and Sims 2002).

[5] `http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html`

arcs, also to DSMs. Therefore, analogously to the MGA model, the control variables of Brennan's model are the departure time and the times of flight between maneuvers. Additional control variables are the positions of the DSM expressed in Cartesian coordinates, leading to four control variables per DSM (three Cartesian coordinates plus the a time of flight).

Although this formulation eliminates the restriction of only one DSM per leg and also allows for powered GAMs, a major defect is that Cartesian coordinates are inconvenient control variables. Since their definition is not relative to a planet or any other part of the trajectory, it is not possible to formulate boundary constraints efficiently, making it unsuitable for global optimization. The source code of this propagator has been made available by Brennan, it has been sent in a personal e-mail.

Multiple-shooting model

An innovative new model for multiple gravity-assist trajectories with an arbitrary number of DSMs per leg has been proposed by Vavrina, Englander, and Ellison (2016). Figure 2.3 provides a schematic interpretation of this multiple-shooting method. Here, $\Delta\mathbf{v}$ is the velocity change at a maneuver (control variable), Δt the time of flight between maneuvers (control variable), $[\mathbf{v}^-, \mathbf{r}^-, m^-]$ and $[\mathbf{v}^+, \mathbf{r}^+, m^+]$ are states at respectively the left and right side of the *match point* (MP).

The method propagates the beginning and end states of the spacecraft in a Keplerian fashion, forward and backward respectively, to the match point in the middle of the trajectory. Continuity of position, velocity and mass are enforced at the match point through nonlinear constraints. Importantly, these are guaranteed through analytical derivatives. Finite differencing would be theoretically possible, but it would involve more calculations.

An advantage of the multiple-shooting method seems to be the lack of a limit on the number of DSMs. However Vavrina, Englander, and Ellison (2016) do not report finding better trajectories when more than one DSM per leg is allowed on a by OSIRIS-REx[6] inspired test case. A less obvious but valuable other advantage of this method is the decreased sensitivity of the cost function, compared to the MGADSM model. Multiple-shooting is the successor of this model in EMTG and brings down the computation time by an order of magnitude when combined with a monotonic basin hopping algorithm (an optimizer, explained in Section 2.3.3).

Despite the superior performance of the multiple-shooting formulation, there is a practical problem. Unfortunately, this formulation is only available in the source code of EMTG. EMTG itself requires licensed optimization software (SNOPT) and neither SNOPT nor an equivalent substitute has been available for this research. Furthermore, interfacing EMTG with other optimization software would be much work for an uncertain outcome. It can be concluded that the multiple-shooting method is promising, but too deeply embedded in the EMTG ecosystem to be used in this research.

2.3 Optimization techniques

Many different optimization algorithms have been used for global optimization of gravity-assist trajectories. In this section, an overview is presented of the ones that have attained noteworthy performances. A general theorem that deserves to be mentioned is the "no free lunch" theorem

[6] An asteroid sample return mission that uses a single Earth GAM. It has been launched in 2016 and is scheduled to return at Earth in 2023 (Lauretta 2012).

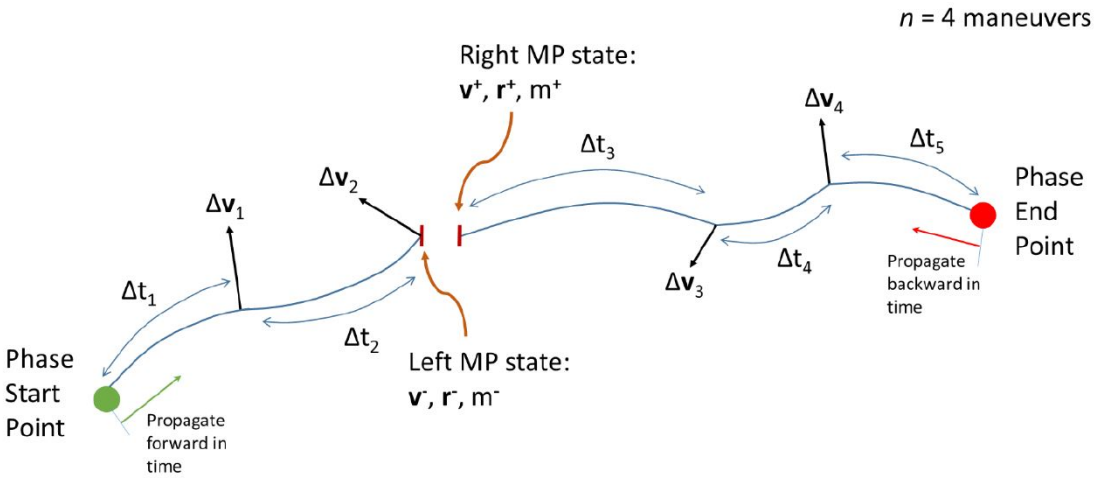


Figure 2.3: Schematic transcription of the multiple-shooting model (Vavrina, Englander, and Ellison 2016)

.

(Wolpert and Macready 1997; Wolpert, Field, and Macready 2005). It states that an improved performance of an optimization algorithm on any class of problems, is paid for with a deteriorated performance on another class of problems. The challenge of formulating an optimization algorithm is therefore to guarantee that the class of optimization problems that pays for the improved performance, does not contain any problem that is of interest. Each of the optimizers that are listed below, handle this challenge differently. They can be categorized in evolutionary algorithms, social behavior algorithms, and repeated local searches. These are addressed respectively in Sections 2.3.1 to 2.3.3. The reader should note that the working principles of the algorithms are not explained in great detail. Instead, the focus is on their performance on gravity-assist trajectory optimization problems.

2.3.1 Evolutionary algorithms

Evolutionary algorithms emulate the mechanism of natural selection in a species. The three variants that have been used most in global trajectory optimization are introduced below.

Genetic algorithm

In a genetic algorithm, solutions are encoded (often using binary bits) in members of a population. Each generation (iteration), members ('parents') are randomly combined to form new members ('children'). The members that are allowed to go to the next generation are selected based on their fitness (objective function value), then the process repeats itself. The genetic algorithm has been proposed by Holland (1975) and its implementation has not changed significantly since then. It was the first evolutionary approach to be suggested for use in trajectory optimization (Janin and Gòmez-Tierno 1985). The performance on fixed GAM sequence trajectory optimization has been demonstrated to be rather poor compared to other metaheuristic methods (Vinko and Izzo 2008). A problem to which it does find much applicability is the generation of flyby sequences. The recombination mechanism makes it capable of dealing with members of variable lengths, which is critical for expressing GAM sequences (Englander 2013).

Differential evolution

Differential evolution is similar to the genetic algorithm, with the key difference that solutions are represented by vectors in a solution space and reproduced through addition and subtraction of random members. It was first proposed by Storn and Price (1997) and has come into wide use for many applications in engineering. It has been proven to be successful on the MGA and MGADSM trajectory models (Vasile, Minisci, and Locatelli 2011). A disadvantage is that it is rather sensitive to the settings that are used. Differential evolution requires the user to select three design parameter values and a strategy, these are often crucial for success. Musegaas (2012) has done an excellent study on the appropriate parameter values to be used on the several problems in the MGA and MGADSM models. Differential evolution is used in this study and explained in detail in Chapter 6.

Covariance matrix adaptation evolutionary strategy

This relatively recent advancement in evolutionary computation uses a multivariate Gaussian distribution for producing offspring, contrary to the recombination strategies of the former two optimizers. It is characterized by a wide applicability to many different objective functions and a major advantage is that it lacks parameters that need to be tuned, arguably except for the population size. Scarce results of CMA-ES on gravity-assist trajectories are ambiguous (Hansen and Ostermeier 2001; Izzo et al. 2013; Izzo, Hennes, and Riccardi 2015).

2.3.2 Social behavior algorithms

Social behavior algorithms are inspired by the interaction between individuals in a group of animals. Indeed, cooperation can lead to results that cannot be obtained by individualistic behavior only.

Particle swarm optimization

In particle swarm optimization, particles move through the solution space, similarly to a flock of birds. The position and velocity of each particle is updated in iterations, based on the fitness of the particle and the fitness of its neighbors. The performance of particle swarm optimization on gravity-assist trajectories has been the topic of various studies, converging to the same conclusion that it performs reasonably, but is significantly outperformed by differential evolution (Vasile, Minisci, and Locatelli 2008; Vasile, Minisci, and Locatelli 2010; Musegaas 2012). An interesting finding by Vinko and Izzo (2008) is that a cooperative (or hybrid, if you will) approach to differential evolution and particle swarm optimization (named COOP), in which the population is passed between both algorithms for a fixed number of iterations, yields better results than both particle swarm optimization and differential evolution individually, on a specific set of problems.

Ant colony optimization

In ant colony optimization, acquired information about the fitness of regions of the solution space is shared by members of the population. Consequently, members base their movement on this information. It has been used by Ceriotti (2010) and forms the core of the MIDACO solver (Schlueter, Egea, and Banga 2009), of which the development has been co-funded by ESA and Astrium (predecessor of Airbus Defence and Space[7]). In a study on its performance on the MGA and MGADSM models, it was found that MIDACO almost always identifies the global minima, but requires a high number of function evaluations as well (Schlueter 2014). It is outperformed (in terms of function evaluations) by differential evolution algorithm by an order of magnitude.

[7] A license for this software has not been available for this study.

2.3.3 Repeated local searches

Repeated local searches consist of, as the name suggest, multiple local optimization efforts. Two algorithms are stated below.

Multi-start

The multi-start principle is simple; it involves starting local optimization efforts from various locations. Its ability to identify the global minimum is inversely related to the size of the basin[8] that contains the global optimum (the global minimum basin). An (possible) advantage is that its computational efficiency can be increased by including numerical derivatives, if available. The reported success rate is outstanding on simpler problems, but deteriorates quickly with more GAMs; it does not perform well Cassini-like trajectories (Vasile, Minisci, and Locatelli 2010).

Monotonic basin hopping

The starting points of the local searches in monotonic basin hopping are generated by randomly perturbing the solution of a previous local optimization effort (Englander and Englander 2014). This optimizer only works well if the fitness values decrease over a long range towards the global minimum, which is the case for many objective functions of gravity-assist trajectories. It is currently used in EMTG (Vavrina, Englander, and Ellison 2016) and has previously demonstrated the ability to handle MGA and MGADSM problems (Vasile, Minisci, and Locatelli 2010). Its performance is greatly influenced by the local optimization method, which poses practical difficulties to this research. Indeed, the state-of-the-art solver used by EMTG is unfortunately not available for this research. Also, it does not cope well with discontinuities of the fitness value; the a priori approach introduces many discontinuities (this is explained in Section 5.2).

2.3.4 Other optimizers

The optimizers mentioned above are not all that have been used in global optimization of gravity-assist trajectories. Other (less successful) methods include a social algorithm named artificial bee colony (ABC) optimization (Karaboga and Basturk 2007), the deterministic divided rectangle (DIRECT) method (Jones, Perttunen, and Stuckman 1993) and the by metallurgy inspired simulated annealing (SA) technique (Kirkpatrick, Gelatt, and Vecchi 1983).

2.4 Sensitivity analyses

Not much literature has been published on the robustness of interplanetary trajectories with respect to the control variables (or the departure epoch specifically). Three relevant concepts have been identified identified in this context: the pork chop plot, a sensitivity analysis, and PEATSA, which uses the EMTG optimization engine. They are respectively addressed in this section.

2.4.1 Pork chop plots

A famous graphical method to gain insight in the characteristics of possible trajectories is the pork chop plot. These graphs plot iso-ΔV contour lines on a background of departure epoch versus time of flight or departure epoch versus arrival epoch. Figure 2.4 shows an example of the latter variant for a ballistic transfer to Mars. The sensitivity to a change of the departure epoch is equivalent to the gradient of ΔV in the horizontal direction. Therefore, where the iso-ΔV contour lines are

[8] A basin contains all the points in a multidimensional space, from which a (local) minimum can be reached via a continuously descending path.

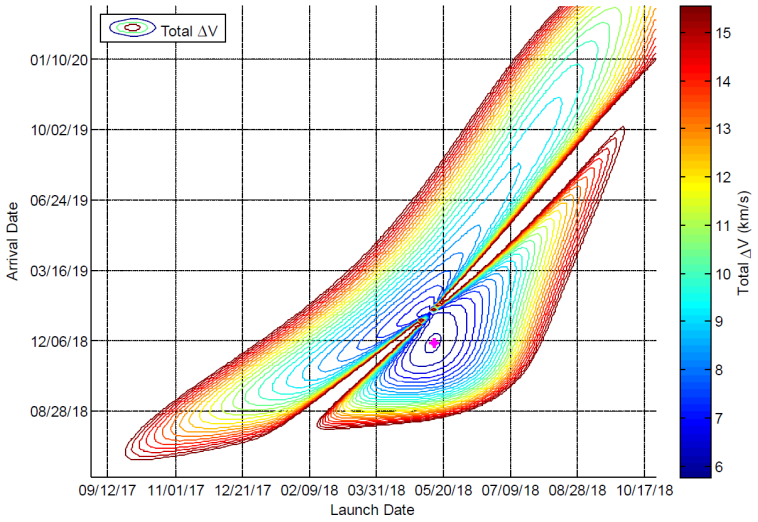


Figure 2.4: Pork chop plot of potential ballistic transfers to Mars (Woolley and Whetsel 2013). The Mars InSight mission used the departure opportunity in May 2018 (Abilleira et al. 2014).

close together in the horizontal direction, sensitivity to departure epoch slip is high.

Using the two-body approximation, a direct ballistic transfer is fully determined by the arrival epoch and the departure epoch; the ΔV budget can be obtained by solving Lambert's problem. However, when the number of control variables is three or more, each combination of a departure and an arrival epoch requires an optimization problem to be solved. This last approach has been taken by Ishimatsu, Hoffman, and Weck (2011), yielding an integrated pork chop plot of both direct and Venus-GAM trajectories to Mars. However, when longer GAM sequences and/or DSMs are considered, the computation time quickly becomes intractable. Therefore no pork chop plots are available for complex interplanetary missions.

2.4.2 Sensitivity analyses

A hybrid global optimization algorithm that uses a combination of differential evolution, CMA-ES and DIRECT has been developed by Stracquadanio et al. (2011). It is named Self-Adaptive Gaussian Evolutionary Strategy (SAGES) and has been applied to the MGA and MGADSM trajectory models. In this same publication, the authors recognize that the solutions of these minimization problems can be very unstable, so a sensitivity analysis is done. The authors research how large perturbations of each variable may be before the increase of the objective function exceeds a predefined limit.

The sensitivity analysis has yielded some insights; it showed for example that the timing of a DSM between Venus and Mercury is particularly sensitive in a trajectory inspired by the Messenger mission.[9] However, from an operational point of view, the effect of a perturbation of a single variable, while all others remain fixed, seems of little interest. It is likely that the effect of a perturbed variable can be mitigated by re-optimizing others. It can be argued that only those control variables that can still be changed when a perturbation in a variable has come to light, may be subjected to re-optimization. For example, if the epoch of the first DSM slips in a MGADSM tra-

[9] A mission to Mercury that departed from Earth in 2004 and injected into orbit in 2011, after an Venus-Venus-Mercury-Mercury GAM sequence, excluding arrival and departure (McAdams et al. 2006; Bedini 2017).

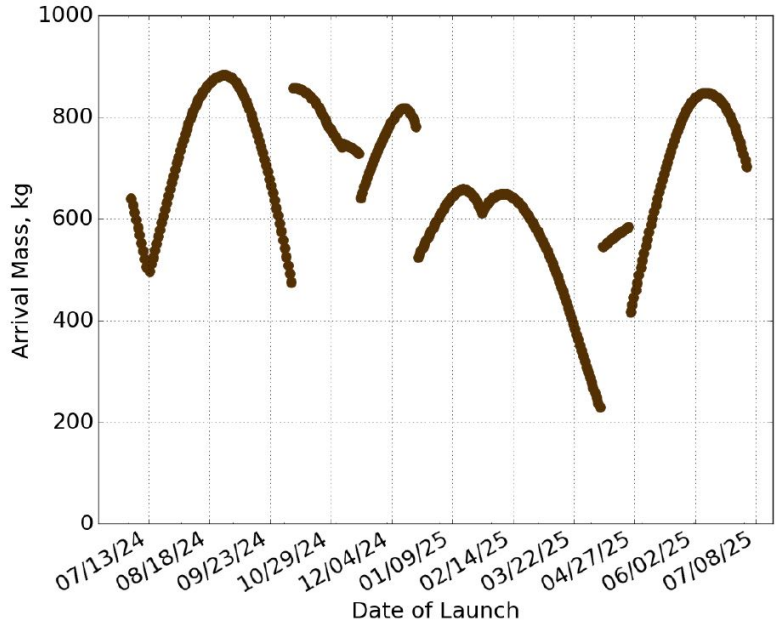


Figure 2.5: Output of a launch window study of a hypothetical mission to Uranus, performed by PEATSA. Different curves correspond to different GAM sequences. Notice that arrival mass is on the y-axis, which is inversely related to the ΔV budget (Knittel et al. 2017).

jectory, a re-optimization effort may include the timing of the second DSM, but not the departure epoch since this cannot be changed anymore. This seems an interesting approach for studies on the robustness of all control variables in a trajectory. Regarding the departure epoch uncertainty that is considered in this study, all variables are free for re-optimization since it is chronologically the first control variable.

2.4.3 Python EMTG automated trade study application

The Python EMTG automated trade study application (PEATSA), developed by Knittel et al. (2017), is the only identified algorithm that can be used for optimizing the robust trajectory pair. It works as a shell around EMTG and (therefore) takes an a posteriori approach. Figure 2.5 is generated by PEATSA and shows the arrival mass of a hypothetical mission to Uranus, for a range of departure epochs. This is the only high-thrust mission of which results of PEATSA have been published, a lack of details on the computational performance makes that they cannot be used for verifying the methods developed in this study.

A noteworthy feature of PEATSA is that it intelligently exploits similarity between trajectories of which the departure epochs are close; a previous solution is used as an initial guess for the current departure epoch. Nonetheless, it took 36 hours of computation time on a 64 core (!) processor to generate the graph in Figure 2.5. In this respect, it is also important to note that Figure 2.5 considers bi-level optimization (determining both the GAM sequence and the control variables), which is much more difficult than optimizing a trajectory of a fixed GAM sequence. Therefore, it can be concluded that PEATSA has capabilities that are outside the scope of the proposed research, since this thesis focuses on fixed GAM sequences exclusively. Still, the use of initial guesses is a powerful mechanism that is further developed in Chapter 7.

Chapter 3

Methodology

The methodology that is used in this thesis research is composed of various different elements that have been adopted from literature, but a substantial part is self-developed as well. The following chapter has six sections that each address distinct elements of the methodology. A section that is very specific to the formulated approach to departure epoch uncertainty, is Section 3.1. It contains formal definitions that are fundamental to the rest of the methodology.

3.1 Definitions

In this section, three central concepts are defined: the robust trajectory pair, the a posteriori approach and the a priori approach. The latter two are based on the definition of the former.

3.1.1 Definition of robustness with respect to departure epoch

The following definition of the robust trajectory pair is used in this research.

> **Definition of a robust trajectory pair:** A trajectory is robust if it has a backup departure opportunity at a user-specified interval $\Delta T0$ after the first departure epoch. ΔV budget ΔV_{rb} of the consequent trajectory pair is equal to the highest of the ΔV budgets of the two trajectories.

The reader should note that minimizing a robust trajectory pair causes the ΔV budgets of both trajectories to converge to the same value (for a continuous optimal ΔV budget curve), this can be observed in Figure 3.1.

Two considerations have led to the definition stated above.

1. **Fixed departure epoch interval.** The fixed departure epoch interval $\Delta T0$ leads to a simpler algorithm than allowing a range for the second departure epoch. Still, the a posteriori and a priori methods can be relatively easily extended to allow $\Delta T0$ to be defined as a range. For the former, the same optimal ΔV budget curve can be used, in case of the latter it would involve an extra optimization variable. This may be done in a future research, the departure epoch interval is kept fixed in this study.

2. **Discrete departure epochs.** This discrete formulation is significantly simpler than a formulation that would demand ΔV_{opt} to be be continuously below ΔV_{rb} during $\Delta T0$. Although

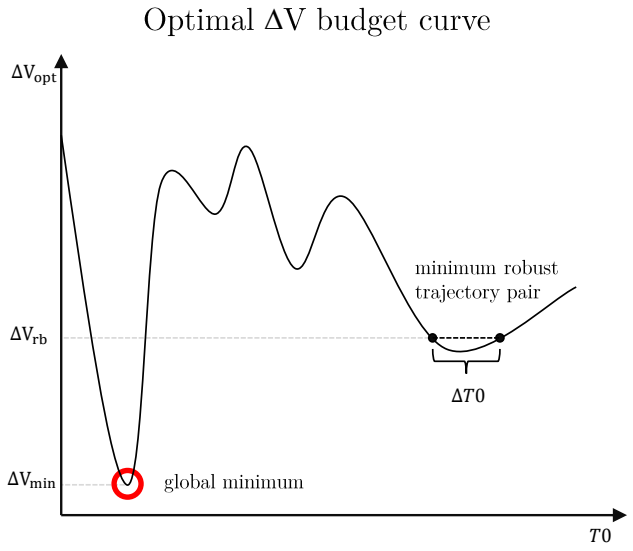


Figure 3.1: A fictitious optimal ΔV budget curve $\Delta V_{opt}(T0)$ and the minimum robust trajectory pair for a specified $\Delta T0$ (not in the global minimum basin).

> ΔV_{rb} is continuously below $\Delta T0$ in the problems studies in this report, and also in Figure 3.1, it is not considered to be required. Two discrete departure opportunities are deemed sufficient.

Correspondence with aerospace engineer Jacob Englander from NASA Goddard Space Flight Center made clear that the order of magnitude of departure epoch interval $\Delta T0$ on which robustness is required is weeks. The critical value is approximately three weeks, according to Mr. Englander. Therefore, in this work departure epoch interval $\Delta T0 = 21$ days unless indicated explicitly otherwise. The $\Delta T0$ value can easily be changed in the developed algorithms.

An observation regarding Figure 3.1 is that it illustrates a situation in which the minimum robust trajectory pair is not in the same basin as the global minimum. Hence, only exploring this global minimum basin would yield a suboptimal robust trajectory pair. Some of the methods proposed in Chapter 7 prune the solution space drastically, thereby potentially missing minimum robust trajectory pairs like the one in Figure 3.1.

A specifically interesting other characteristic of the robust trajectory pair, besides its velocity budget ΔV_{rb}, is the difference in time of flight between paired trajectories. One of the problems considered in this research (Cassini0, Section 4.3) yields a global minimum pair with five years and five months between the two trajectories' times of flight. In this case, choosing a suboptimal pair reduces this difference to one year and three months, at the cost of a mere 67 m/s. One can imagine that this latter trajectory can be found overall more attractive. It illustrates that ΔV_{rb} is not the only relevant property of a robust pair. However, as outlined in Chapter 1, the focus is exclusively on the ΔV budget as a measure of the fitness of the trajectories.

3.1.2 Definitions of the a posteriori and a priori approaches

The following definition of the a posteriori approach is used in this research.

Definition a posteriori approach: The a posteriori approach first computes minimum ΔV budget ΔV_{opt} for a range of departure epochs $T0$ (optimal ΔV budget curve). Thereafter, robust ΔV budget ΔV_{rb} is obtained from pairs of ΔV_{opt} values that are departure epoch interval $\Delta T0$ apart.

Similarly, the definition of the a priori approach is formulated as follows.

Definition a priori approach: The a priori approach minimizes robust ΔV budget ΔV_{rb} directly as a function of the control variables of both trajectories (except the second departure epoch $T0_2$), given a departure epoch interval $\Delta T0$.

In Chapter 5, these definitions are extended into architectures of both approaches.

3.2 Use of existing methods

The trajectory models and the optimization algorithm are selected from the methods listed in Chapter 2. It must be noted that the a posteriori and a priori method are frameworks that can be applied to any trajectory model or optimization algorithm.

3.2.1 Use of MGA and MGADSM trajectory models

Table 3.1 shows a systematic assessment of the four trajectory models that have been identified in Section 2.2, with respect to four criteria. It can be seen that the MGA and MGADSM models have a reasonable overall performance, they can also be relatively easily implemented. Therefore, these have been seleceted. The multiple-shooting model is discarded because its implementation poses difficulties.

Table 3.1: Assessment of the trajectory models on four criteria. Criterion 'flexibility' indicates possibilities in modeling different maneuvers and trajectories.

	MGA	**MGADSM**	**M Lambert**	**M shooting**
Implementability	Easy	Easy	Easy	Difficult
Flexibility	Low	Medium	High	High
Verification problems available	Some	Many	None	Some
Optimizability	Medium	Medium	Bad	Good

3.2.2 Use of differential evolution

There is made use of the differential evolution optimization algorithm. This derivative free optimizer has demonstrated its effectiveness on the MGA and MGADSM models and is derivative-free. A disadvantage is its sensitivity to design parameters, this is specifically addressed in Chapter 6. It must be noted that an implementation of monotonic hopping has been developed as well, which is described in Appendix B. Unfortunately, this did not lead to a performance comparable to values reported in literature. This is attributed to the perturbing operator that was used.

3.2.3 Use of MATLAB and C++ programming languages

MATLAB is used as the programming language for developing and testing algorithms, its superior user-friendliness makes it the method of choice. However, its computational performance is rather poor (Eichorn et al. 2016). Therefore, C++ implementations of the trajectory models are compiled as MEX functions[1], these are interfaced with MATLAB. It has been found that compiling C++ source code as a MEX function decreases the computation time of an objective function evaluation by a factor twenty, compared to the MATLAB implementations of the models. The source code of the MGA and MGADSM functions is available for download from the web page of ESA's Advanced Concepts Team.[2] The used differential evolution code has been made available as `devec3.m` on the web page of Rainer Storn,[3] one of the inventors of differential evolution.

3.3 Development of new algorithms

Several new algorithms are designed and developed, most notably those described in Chapter 7. In this chapter, five different concepts that are aimed at exploiting the similarity between paired trajectories, with a limited $\Delta T0$, are proposed. Two key principles that are adhered to in the generation of the concepts, are the following.

1. Concepts are sought to be diverse, with their distinctions on a high level. This is to maximize the coverage of the 'solution space' of the concepts.
2. Concepts are mutually exclusive. This allows one to distinguish between individual effects on the basis of experimental results.

The latter condition makes that the concepts can be displayed in a tree structure, as done in Figure 7.1 in Chapter 7.

3.4 Algorithm verification

External software is verified by testing it and comparing the results to values reported in literature. Self-developed software is verified by cross-referencing it to known solutions. Also, chapters that describe the implementation of code, notably Chapter 6 and 7, include sections in which verification is specifically addressed.

3.5 Assessment of results

The results of this research consist of performances of both the a posteriori and a priori methods, as well as an analysis of the methods themselves (an algorithm review). The methodology that is used for obtaining results in both categories, is explained below.

3.5.1 Performance

The performance of the methods is split in accuracy and computational efficiency. In evolutionary optimization, both are often communicating vessels. However, due to the decoupling of optimizing ΔV and finding ΔV_{rb} in the a posteriori method, they are not as directly related anymore. Therefore, accuracy and computational efficiency are addressed separately.

[1] More information on MATLAB MEX functions: https://nl.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html

[2] http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html

[3] http://www1.icsi.berkeley.edu/~storn/code.html

Accuracy

The accuracy of an optimization effort is calculated as the difference between the objective function value that is attained, and the best known objective function value on that problem. Since robust trajectory pairs on the MGA and MGADSM models have not been researched before, the best solutions attained in this research are the best known solution in general.

Computational efficiency

In many optimization problems, the lion's share of the computational effort is taken by the numerous objective function evaluations. Therefore, the number of trajectory function evaluations (a trajectory function calculates the ΔV budget as a function of control variables) is taken as a measure of the computational effort, rather than computation time. This approach allows for comparison between computers of different processing power. On the other hand, one should realize that this method implies that any difference in the overhead computation time of the optimizer is not taken into account. This simplification is common in literature and also used in this research. In case of the a posteriori method, the total number of trajectory function evaluations N_T is documented. It consists of the optimization runs that are required for computing the optimal ΔV budget curve.

A first note on the a priori method, is that its objective function calls twice to the trajectory function per objective function evaluation (see Section 5.2). Furthermore, calculating the computational efficiency is slightly more complicated than for the a posteriori method because not every differential evolution run converges to the global minimum. The here proposed measure is named performance indicator $N95$ and is closely related to the number of trajectory function evaluations that is required to be 95% confident that a user-specified target value VTR has been found at least once Olds, Kluever, and Cupples (2007). It is calculated as follows. Let p be the chance that a certain optimizing run of N_{fe} function evaluations is successful, N_{runs} the number of runs and p_{thresh} the desired confidence interval, thus here $p_{thresh} = 0.95$. Then the maximum allowed chance that out of N_{runs} runs, none reach VTR is $1 - p_{thresh}$. This is mathematically transcribed below.

$$(1-p)^{N_{runs}} = 1 - p_{thresh} \tag{3.1}$$

Rewriting as function of N_{runs} leads to the following formulation.

$$N_{runs} = \log_{(1-p)}(1 - p_{thresh}) \tag{3.2}$$

Changing the base of the logarithm yields the following equation. One should notice that the logarithms in Equation 3.3 can have any base, as long as they are the same for the numerator and denominator.

$$N_{runs} = \frac{\log(1 - p_{thresh})}{\log(1-p)} \tag{3.3}$$

Finally, to find $N95$, Equation 3.3 is multiplied with the average number function evaluations per run.

$$N95 = \overline{N_{fe}} \cdot N_{runs} = \overline{N_{fe}} \cdot \frac{\log(1 - p_{thresh})}{\log(1-p)} \tag{3.4}$$

The reason why $N95$ is closely related to, and not exactly, the number of function evaluations that is required to be 95% confident that VTR has been reached, is because the distribution of N_{fe} is

neglected. The added complexity of including the distribution of N_{fe}, which has been observed to be asymmetrical, would make the performance indicator opaque, while the current formulation of $N95$ already provides sufficient measure of relative performance. Therefore, the approximation using $\overline{N_{fe}}$ is made. Furthermore, a result of the formulation of $N95$ is that the stopping criterion of differential evolution plays a role in the quantification of its performance. This is inevitable when comparing the a posteriori and a priori approach. Stopping criteria are addressed in Chapter 6.

Standard deviation of the success rate

The uncertainty of the results is quantified using the standard deviation σ_p of the success rate. Given that the success rate is the same for each run, any series of N_{runs} can be modeled as a binomial experiment. The standard deviation of the success rate is calculated as follows (Dekking et al. 2005).

$$\sigma_p = \sqrt{\frac{p\,(1-p)}{N_{runs}}} \tag{3.5}$$

The standard deviation can consequently be used to calculate high and low estimates of the performance indicator.

3.5.2 Algorithm review

Besides the quantitative comparison, the following features of the developed algorithms will be compared qualitatively.

1. **Algorithm complexity.** Besides an analysis of the different subroutines of an algorithm, the need for human oversight is also taken into account. This indicates that some task are not automated because of complexity.

2. **Influence of design parameters.** These parameters need to be set by the user. Although they provide control over an algorithm, they are generally considered unfavorable features. Referring to the no free lunch theorem (Wolpert and Macready 1997), optimal settings on one problem lead to suboptimal settings on another. Moreover, the non-physical nature of design parameters makes it difficult to estimate reasonable values. This makes an algorithm prone to suboptimal settings.

3. **Versatility.** This concept is defined as the applicability beyond the strict scope of the problem formulation. It is further split into the following two categories.

   (a) **Results.** It is analyzed if the results of a robust optimization campaign have any additional value for analysis of the mission.

   (b) **Method.** It is assessed if the methods are applicable to similar (isomorphic) problems.

3.6 Application to the Cassini mission

Although the a posteriori and a priori approach are designed to work on any gam sequence, they are specifically applied to three problems that are inspired the trajectory of Cassini. This mission to Saturn and its moons was launched on October 17th 1997 and arrived at Saturn on July 1st 2004, after consecutive flybys of Venus (twice), Earth and Jupiter. Only on September 15th 2017 has the orbiter been decommissioned after many ground-braking discoveries. It has been one of

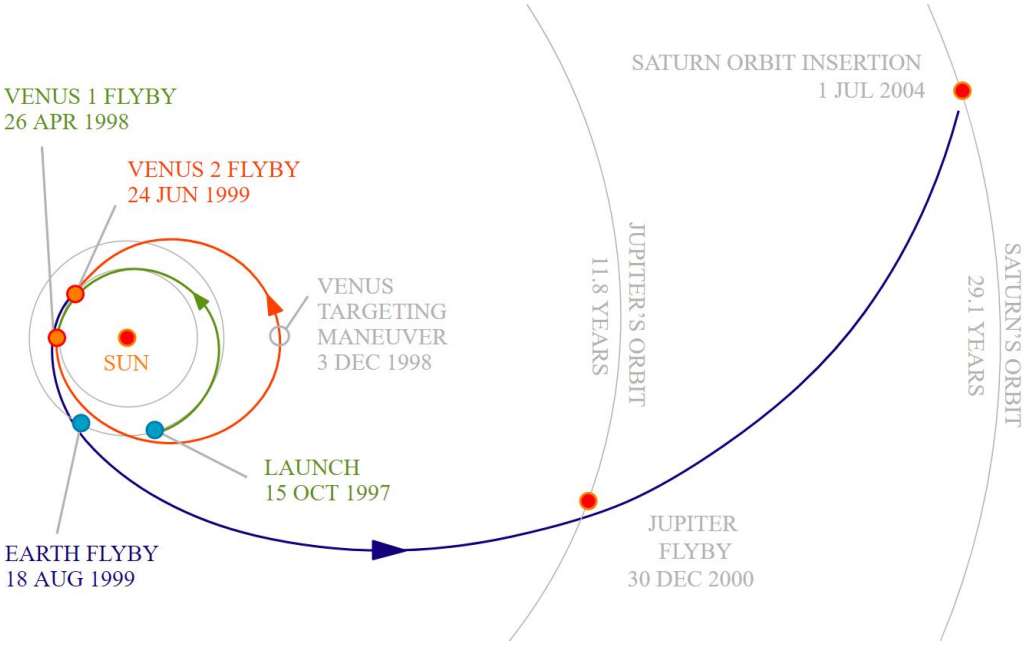


Figure 3.2: Overview of the Cassini trajectory.[4]

the most successful interplanetary missions in history (Coates 2017).

Figure 4.6 shows the trajectory of Cassini. The following three reasons have led to selecting it as a case study.

1. It is a challenging trajectory with four gravity-assist maneuvers (GAMs).
2. Although the actual trajectory contains deep space maneuvers (DSMs), it can be approximated without DSMs as well.
3. It has been well researched, therefore sufficient verification material is available in literature.

In Chapter 4, three variants of the Cassini trajectory are proposed as test problems.

[4] https://saturn.jpl.nasa.gov/resources/1776/cassini-trajectory/

Chapter 4

Trajectory models

To understand the capabilities and limitations of the MGA and MGADSM trajectory models, it is vital to gain insight into their working principles. This chapter explains how they calculate the ΔV budget from a set of control variables (together contained in the decision vector). The structure of this chapter is threefold. First, the iterative nature of Kepler and Lambert propagation is addressed in Chapter 4.1. In the following section, it is explained how the events of departure, DSMs, GAMs, and arrival are modeled. Finally, a description of the three test problems (Cassini0, Cassini1, and Cassini2[1]) is provided in Section 4.3.

4.1 Two-body problem propagators

Kepler propagation is explained first, thereafter Lambert propagation. Full analytical relations are omitted since they are not required for understanding how the a posteriori and a priori approach work, they can be found in Bate, Mueller, and White (1971) and Izzo (2014) respectively. The focus of this section is to point out the key challenges and characteristics of both propagation methods.

4.1.1 Kepler propagation

Propagating a spacecraft's position and velocity given a time of flight involves transforming the Cartesian state $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ to an orbit in terms of Kepler elements $(a, e, i, \omega, \Omega, E)$ and back (the quantities can be found in the List of symbols). The analytical relations for both transformations are well known, the implementation used in the implementation by Izzo (2010) are based on Bate, Mueller, and White (1971). A limitation of this method is that there are two singularities in the transformation from Kepler elements to Cartesian coordinates (for $i = 0$ and when an intermediate parameter equals π), but no errors due to these singularities have occurred during this research. To find the new position after the transformation to Kepler elements, mean anomaly M is required. It can be calculated as follows.

$$M = E - e \cdot \sin E \tag{4.1}$$

The difference between two mean anomalies M_1 and M_2 is calculated as a function of time of flight T. The sign before gravitational parameter μ is positive for ellipses and negative for hyperbolas.

$$M_2 - M_1 = \sqrt{\frac{\pm\mu}{a^3}} \cdot T \tag{4.2}$$

[1] The self-developed problem Cassini0 is simpler than both known problems, Cassini1 and Cassini2 (Izzo 2010). Therefore, the self-developed problem is given a lower number, hence counting starts at zero.

Using the inverse of Equation 4.1 leads to the new state in Kepler elements. However, there is no closed-form expression for eccentric anomaly E, therefore its value is determined iteratively using the Newton-Rhapson method (Raphson 1702). This is one of the issues that hinder analytical optimization (see Chapter 1). Finally, the new Cartesian coordinates can be calculated using the new eccentric anomaly.

4.1.2 Lambert propagation

The reader is reminded that Lambert's problem is concerned with solving the boundary value problem of finding two velocities, given the two associated positions and a time of flight. Since the first accurate solution was found by Gauss (1857), various different approaches have been suggested. The key challenges are, analogously to Kepler propagation, singularities (see below) and the lack of explicit relations. The method that is implemented in the trajectory models has been developed by Izzo (2014). The main innovation of Izzo's approach is the introduction of a new variable k that is updated in iterative loops. It is defined as follows.

$$k = \log\left(1 + \cos\left(\frac{\beta}{2}\right)\right) \tag{4.3}$$

Angle β is a function of two angles ϕ and ψ.

$$\beta = \phi + \psi \tag{4.4}$$

Furthermore, ϕ and ψ are functions of the eccentricity and eccentric anomalies.

$$\psi = \frac{E_2 - E_1}{2} \tag{4.5a}$$

$$\cos\phi = e \cdot \cos\left(\frac{E_2 - E_1}{2}\right) \text{(elliptic)} \tag{4.5b}$$

$$\cos\phi = e \cdot \cosh\left(\frac{E_2 - E_1}{2}\right) \text{(hyperbolic)} \tag{4.5c}$$

Then, parameter k is calculated as a function of the time of flight (see Equation 4.1 and 4.2) and updated in an iterative loop, until convergence is declared. The Newton-Rhapson method is used here as well (Raphson 1702).

A result of this new formulation is an improved accuracy of the initial guess for the iterations, with respect to previous solution methods. Also, the velocity estimates do not show singular behavior for a transfer angle close to π (half an orbit), but k remains undefined for a transfer angle of zero.

4.2 Trajectory events

Figure 4.1 shows schematic overviews of the MGA and MGADSM trajectories. It can be seen that, from left to right (thus forward in time), the following types of events are encountered.

1. Departure
2. A DSM (only in the MGADSM model)
3. A GAM (powered in the MGA model and ballistic in the MGADSM model)
4. Arrival

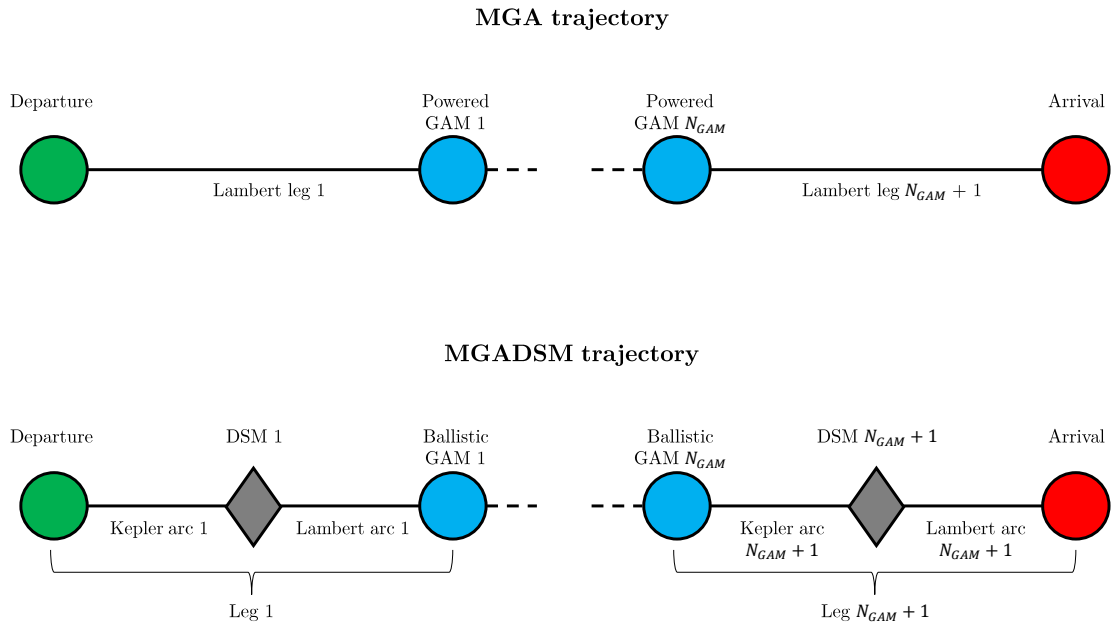


Figure 4.1: Schematic description of the MGA and MGADSM trajectory models.

Table 4.1: Control variables of the MGA model. The last column indicates the number of the control variables as function of the number of GAMs N_{GAM}.

Symbol	Quantity	Units	Number
$T0$	Departure epoch	MJD2000	1
T	Time of flight between planets	days	$N_{GAM}+1$

The control variables of both trajectory models are stated in Table 4.1 and 4.2. In the following sections, the formulas that express ΔV in terms of these variables are explained, by analyzing the events in the respective order.

4.2.1 Departure

Departure is modeled as an impulse that contributes directly to the ΔV budget, its magnitude is denoted by $\Delta V0$. Both trajectory transcriptions only account for the hyperbolic excess velocity in $\Delta V0$, assuming that escape from Earth's gravity well is handled by a launch vehicle.

In the case of the MGA model, the magnitude of $\Delta V0$ follows from solving Lambert's problem between departure and the first GAM. In the MGADSM model, propagation from the departure planet to the first DSM is done using Kepler's method, requiring the velocity vector at departure to be fully determined. This is done by including right ascension α_{LA} and declination δ_{LA} (both of the launch asymptote) as well as $\Delta V0$ in the decision vector. Definitions of the former two variables, relative to a non-rotating Earth, are shown in Figure 4.2.

4.2.2 Deep space maneuvers (MGADSM only)

DSMs are short bursts of thrust that are modeled as discrete changes of the velocity vector. The location of a DSM is determined by ratio η; it states at which fraction of time of flight T between

Table 4.2: Control variables of the MGADSM model. The last column indicates the number of the control variables as function of the number of GAMs N_{GAM}.

Symbol	Quantity	Units	Number
$T0$	Departure epoch	MJD2000	1
$\Delta V0$	Departure impulse magnitude	km/s	1
α_{LA}	Right ascension of the launch asymptote	radians	1
δ_{LA}	Declination of the launch asymptote	radians	1
T	Time of flight between planets	days	$N_{GAM}+1$
η	Fraction of T at which the DSM takes place	ratio	$N_{GAM}+1$
R	Ratio between the flyby perigee and planetary radius	ratio	N_{GAM}
γ	b-plane insertion angle	radians	N_{GAM}

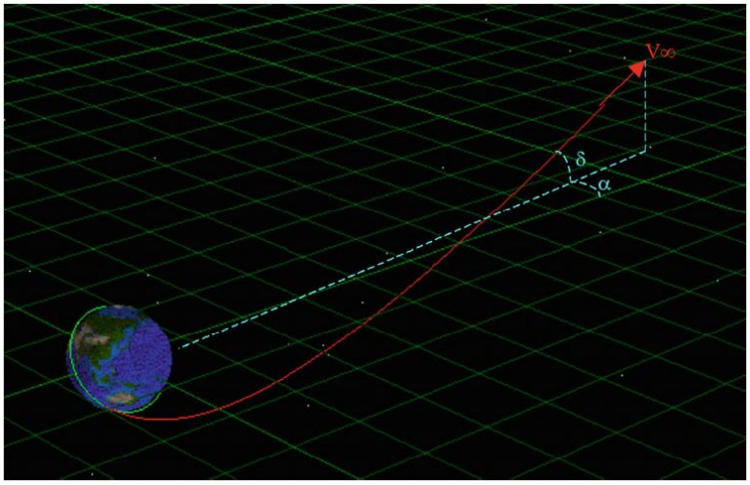


Figure 4.2: Departure trajectory with right ascension α_{LA} and declination δ_{LA} (both of the launch asymptote). The subscripts are omitted in this figure (Biesbroek 2016).

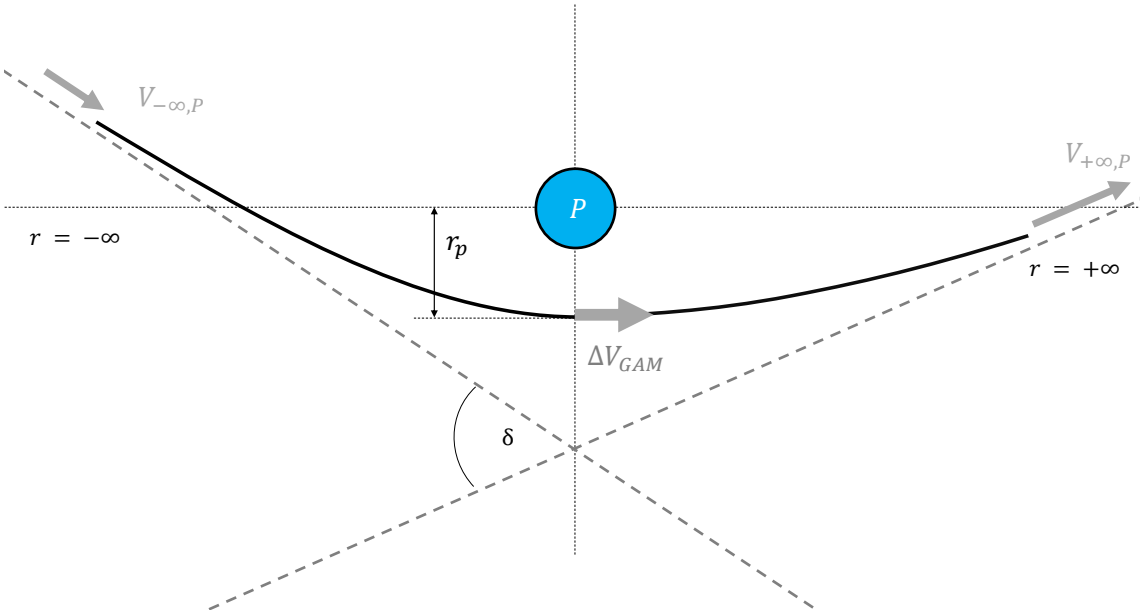


Figure 4.3: Schematic drawing of a powered GAM.

two planets a DSM takes place. Therefore, Kepler propagation from the last planet over an interval ηT yields the location and incoming velocity of the DSM. Then, Lambert's problem can be solved between the DSM and the next planet, the time of flight is $(1-\eta)T$, yielding the outgoing velocity of the DSM. Simple vector calculus reveals the ΔV contribution of the DSM.

4.2.3 Gravity-assist maneuvers

Two different types of GAMs are considered in this work: the powered variant in the MGA model and the ballistic variant in the MGADSM model. They are explained in this order because the equations of the ballistic GAM can be easily derived from the relations that hold for the powered variant.

Powered gravity-assist maneuver (MGA only)

Figure 4.3 shows a GAM with an impulse (parallel to the direction of flight) at the perigee of the flyby. The general expression for the outer asymptote angle δ is stated in Equation 4.6 (Englander 2013), in which μ_{pl} is the gravitational parameter of the planet. Furthermore, the magnitudes of incoming and outgoing velocities at an infinite radial distance from the planet are denoted by $V_{-\infty,P}$ and $V_{+\infty,P}$.

$$\delta = \sin^{-1}\left(1 + \frac{r_p V_{-\infty,P}^2}{\mu_{pl}}\right)^{-1} + \sin^{-1}\left(1 + \frac{r_p V_{+\infty,P}^2}{\mu_{pl}}\right)^{-1} \tag{4.6}$$

The incoming an outgoing heliocentric velocities are obtained by solving Lambert's problem on the trajectory legs between planets. Transforming both velocities as well as the angle between them to the planetocentric reference frame yields $V_{-\infty,P}$, $V_{+\infty,P}$ and δ. This leaves r_p the only unknown in Equation 4.6. There is no closed-form expression for r_p, consequently the Newton-Rhapson method is used again (Raphson 1702). Then, after the flyby radius is determined, the magnitude of the burn at the perigee ΔV_{GAM} can be obtained by applying conservation of angular momentum in ballistic flight. When r_p violates a predefined constraint relative to the planetary

radius (to avoid atmospheric entry or collision), the magnitude of the violation is multiplied with a factor and added as a penalty to the ΔV budget.[2] This steers any optimizer away from infeasible flyby geometries.

Ballistic gravity-assist maneuver (MGADSM only)

Since no burn is applied during the GAMs in the MGADSM model, the magnitudes of the incoming and outgoing velocities are equal. This leads to the following equality.

$$V_{-\infty,P} = V_{+\infty,P} \tag{4.7}$$

This means that both terms of Equations 4.6 are equal. Since the arc after the GAM is propagated using Kepler's method (see Figure 4.1), the direction of outgoing velocity vector $\mathbf{V}_{+\infty,\mathbf{P}}$ needs to calculated. This is done as follows.

The ratio between r_p and the planetary radius is denoted as R and is included in the decision vector.[3] Since $V_{-\infty,P}$ has been obtained by solving the Lambert arc between the DSM and the GAM, δ can be calculated. Next, the plane in which δ is defined, is obtained from the last optimization variable, b-plane insertion angle γ. This b-plane is perpendicular to the incoming velocity asymptote and passes through the center of the body. Figure 4.4 shows its orientation with respect to a hyperbolic flyby trajectory. The angle between the spacecraft's velocity and vector $\mathbf{B}$ at the moment of crossing the b-plane, is γ. The outgoing velocity is calculated as follows, with $\left[\hat{i}, \hat{j}, \hat{k}\right]$ unit vectors describing the b-plane.

$$\mathbf{V}_{+\infty,\mathbf{P}} = V_{-\infty,P} \cdot \left(\cos\delta\hat{i} + \cos\gamma\cos\delta\hat{j} + \sin\gamma\sin\delta\hat{k}\right) \tag{4.8}$$

The unit vectors of the b-plane in the heliocentric frame are obtained as follows, where $\mathbf{V}_{\mathbf{P},\mathbf{H}}$ is the heliocentric velocity vector of the planet (Englander 2013).

$$\hat{i} = \frac{\mathbf{V}_{-\infty,\mathbf{P}}}{V_{-\infty,P}} \tag{4.9a}$$

$$\hat{j} = \frac{\hat{i} \times \mathbf{V}_{\mathbf{P},\mathbf{H}}}{||\hat{i} \times \mathbf{V}_{\mathbf{P},\mathbf{H}}||} \tag{4.9b}$$

$$\hat{k} = \hat{i} \times \hat{j} \tag{4.9c}$$

The MGADSM model can be relatively easily extended to allow for powered flybys. This would require a single extra control variable per GAM (for a perigee burn parallel to the direction of flight), for example the magnitude of the impulse. This can be implemented as part of a follow-up study, but it is not expected to play a significant role in the relative performance of the a posteriori and a priori methods.

4.2.4 Arrival

There are two different manners in which arrival is modeled. A first option is that the spacecraft is injected into an arrival orbit that is determined by its eccentricity and perigee (where it also joins the orbit), the inclination is equal to the that of the incoming trajectory. The magnitude of the contribution to the ΔV budget is obtained by equating potential, kinetic and orbital energy in the planetocentric frame. The second option is a 'rendez-vous' arrival, which eliminates the

[2] The values that are used have been suggested by Izzo (2010). Jupiter has a penalty factor of 0.001 and all other planets 0.01.

[3] One should note that this makes the penalty function redundant.

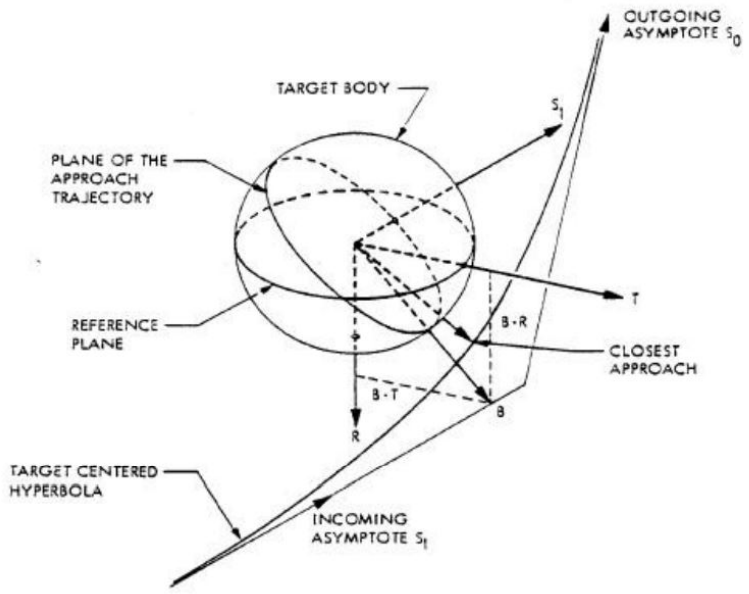


Figure 4.4: Schematic drawing of the b-plane (Jones 2016). Vector **B** points through the b-plane to the incoming asymptote of the spacecraft.

relative velocity between the spacecraft and the target (planet), the gravity of the arrival body is neglected. Although this last arrival type is not realistic for planetary missions, it is used in the Cassini2 verification problem (Izzo 2010).

4.3 Application to the trajectory of Cassini

Three different problems have been formulated, each of different optimization difficulty. While the first (Cassini0) is self-developed, the second and third problems (Cassini1 and Cassini2) are obtained from the web page of ESA's Advanced Concepts Team[4] and have also been used to verify the performance of the implementations of the trajectory models. The exact same values as reported by ESA's Advanced Concepts Team have been obtained, therefore the trajectory models are considered verified.

4.3.1 Cassini0 MGA trajectory

The Cassini0 trajectory is a highly simplified version of the actual trajectory: it omits the GAMs of the inner planets and does not allow for DSMs. At arrival, the spacecraft injects into an eccentric orbit with $e = 0.98$ and perigee radius $r_p = 108950$ km, these are the same values as in Cassini1. The minimum ΔV trajectory is displayed in Figure 4.5, the parameter boundaries and numeric solution are shown in Table 4.3. Figure 4.5 shows that the Jupiter GAM is virtually ballistic. In total, 95% of the ΔV budget stems from the departure impulse. Impulses yield more energy gain at higher velocities; in the trajectory in Figure 4.5 the heliocentric velocity is highest near departure.

[4] http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html

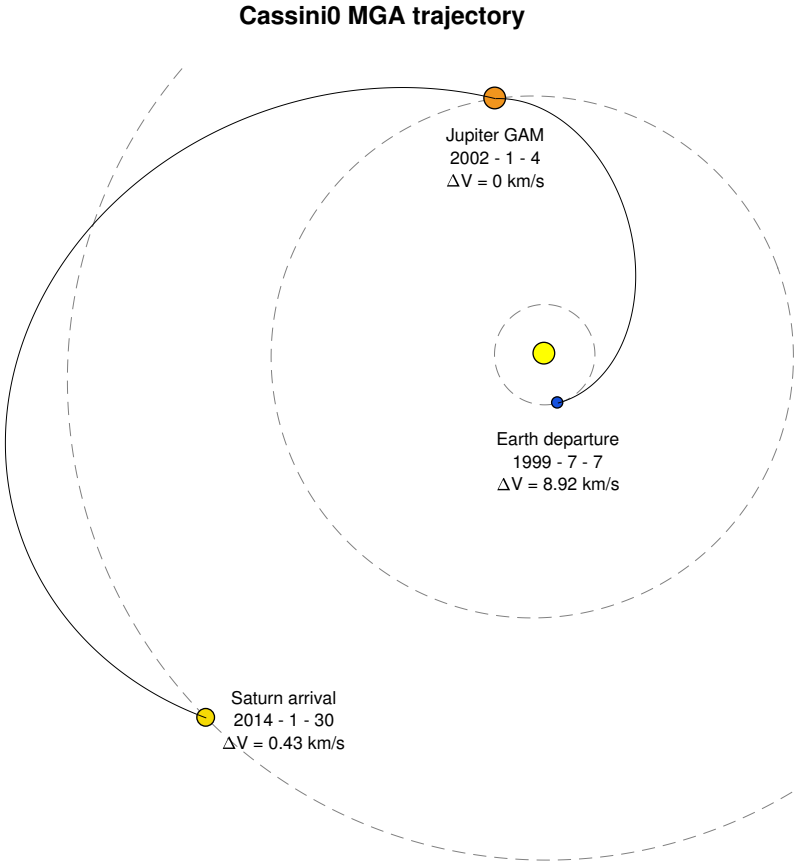


Figure 4.5: Global minimum trajectory of the Cassini0 MGA problem.

Table 4.3: Best attained solution on the Cassini0 MGA problem. The limited dimensionality makes it highly likely that it is the global minimum solution.

Variable	**Units**	**Lower boundary**	**Upper boundary**	**Solution**
T0	MJD2000	-1000	0	-177.810
T1	days	400	2000	911.903
T2	days	2000	6000	4409.37
ΔV	km/s			9.352

4.3.2 Cassini1 MGA trajectory

The Cassini1 MGA problem has the same planetary sequence as the actual mission: Earth, Venus, Venus, Earth, Jupiter, Saturn. Its global minimum trajectory is displayed in Figure 4.6. The first two flybys, both at Venus, are separated by a resonant leg; its period is twice the orbital period of Venus; both GAMs occur at the same location. In the solution space analysis in Appendix A it is demonstrated that the ΔV budget is very sensitive to the duration of this second leg. Also, at the fourth GAM (Jupiter), a low flyby radius results in a sharp transfer angle. Table 4.4 shows the control variable values of the global minimum solution. The associated ΔV budget is 4.931 km/s, which is considerably lower than the 9.352 km/s of the Cassini0 problem. This illustrates the effectiveness of GAMs.

Table 4.4: Global minimum solution to the MGA Cassini1 problem (Izzo 2010).

Variable	Units	Lower boundary	Upper boundary	Solution
$T0$	MJD2000	-1000	0	-789.812
$T1$	days	30	400	158.302
$T2$	days	100	470	449.386
$T3$	days	30	400	54.7490
$T4$	days	400	2000	1024.36
$T5$	days	1000	6000	4552.31
ΔV	km/s			4.931

4.3.3 Cassini2 MGADSM trajectory

Figure 4.7 shows the global minimum solution of the Cassini2 MGADSM problem in the inner solar system. It resembles the actual trajectory of Cassini more than the MGA transcription, due to the V_∞-leveraging maneuver between the two successive Venus flybys. Interestingly, the direction of this DSM 2 is opposite to the heliocentric velocity of the spacecraft. Intuitively, it seems counterproductive to reduce the orbital velocity en-route from Earth to the Jovian system. Apparently, the leveraging effect outweighs the orbital energy decrease of DSM 2. Furthermore, Figure 4.8 shows the rest of the trajectory as well, DSMs 3, 4 and 5 are negligible.

Table 4.4 shows the numerical values of the control variables of the best known solution. Two values lie on a boundary constraint, $T5 = 2200$ days and $R2 = 1.05$. One can imagine that relaxing the constraint on $T5$ may be feasible and worth the extra flight time, while the lower limit on $R2$ is physical. It stems from the radius of Venus and its atmosphere.

The global minimum ΔV budget is 8.385 km/s. The reason that this is higher than the ΔV budget of the MGA trajectory, is because of a rendez-vous arrival at Saturn. The ΔV budget including arrival orbit injection (into the same orbit as Cassini0 and Cassini1) would be 4.610 km/s, showing the merit of the MGADSM over the MGA trajectory, which has a ΔV budget of 4.931 km/s and a longer time of flight.

4.3.4 Comparison with the actual trajectory

Cassini0 has a higher ΔV budget than the actual Cassini mission, due its shorter flyby sequence. However, both Cassini1 and Cassini2 are more propellant-efficient than the actual mission, with ΔV budgets of respectively 4.931 and 4.610 km/s (when orbit injection is considered instead of

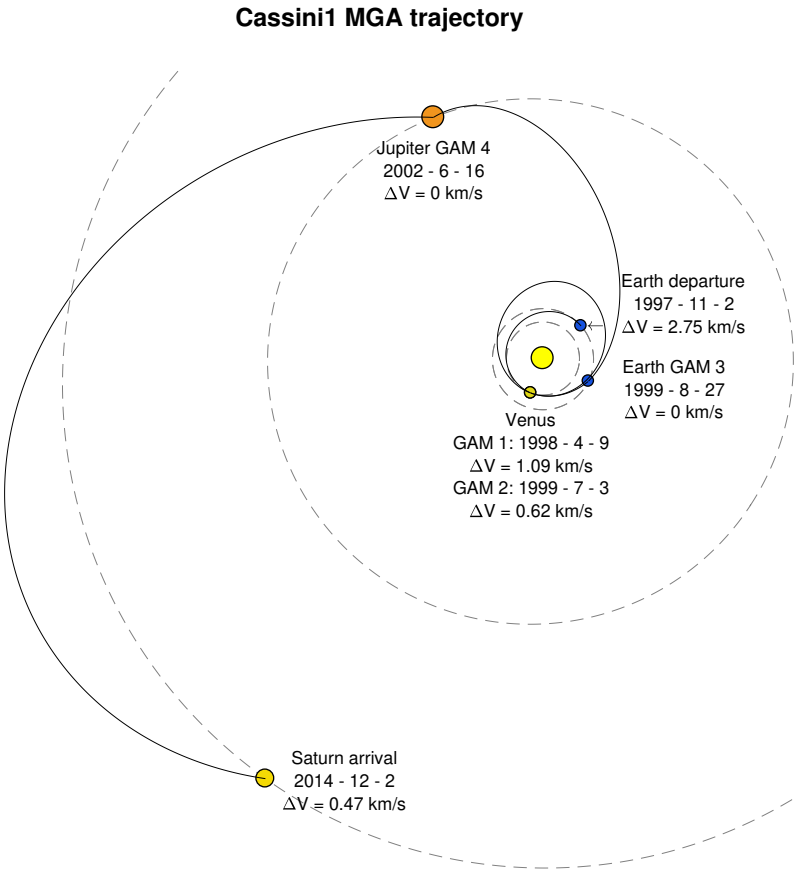


Figure 4.6: Global minimum trajectory of the Cassini1 MGA problem.

Table 4.5: Global minimum solution to the MGADSM Cassini2 problem (Izzo 2010).

Variable	**Units**	**Lower boundary**	**Upper boundary**	**Solution**
$T0$	MJD2000	-1000	0	-779.047
ΔV_{dep}	km/s	3	5	3.25911
α_{LA}	radians	0	1	0.52598
δ_{LA}	radians	0	1	0.38086
$T1$	days	100	400	167.379
$T2$	days	100	500	424.028
$T3$	days	30	300	53.2897
$T4$	days	400	1600	589.767
$T5$	days	800	2200	2200.00
$\eta 1$	ratio	0.01	0.9	0.76948
$\eta 2$	ratio	0.01	0.9	0.51329
$\eta 3$	ratio	0.01	0.9	0.02742
$\eta 4$	ratio	0.01	0.9	0.26399
$\eta 5$	ratio	0.01	0.9	0.59998
$R1$	ratio	1.05	6	1.34878
$R2$	ratio	1.05	6	1.05000
$R3$	ratio	1.15	6.5	1.30730
$R4$	ratio	1.7	291	69.8090
$\gamma 1$	radians	$-\pi$	π	-1.59374
$\gamma 2$	radians	$-\pi$	π	-1.95953
$\gamma 3$	radians	$-\pi$	π	-1.55499
$\gamma 4$	radians	$-\pi$	π	-1.51346
ΔV	km/s			8.385

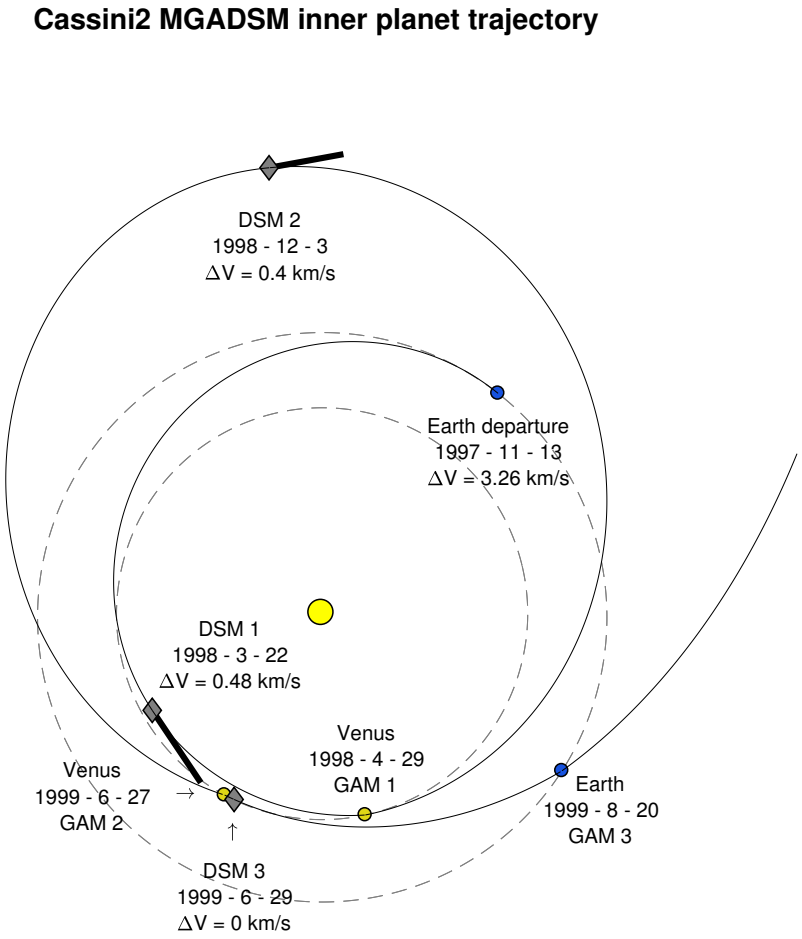


Figure 4.7: Zoomed-in view of the global minimum trajectory of the Cassini2 MGADSM problem.

rendez-vous) versus 5.33 km/s on the actual mission (Englander 2013). This can be ascribed to the significantly shorter time of flight of the latter: 2451 days. The Cassini1 and Cassini2 times of flight are respectively 7028 (!) and 3434 days. This illustrates the trade between conflicting objectives of minimizing both ΔV and time of flight.

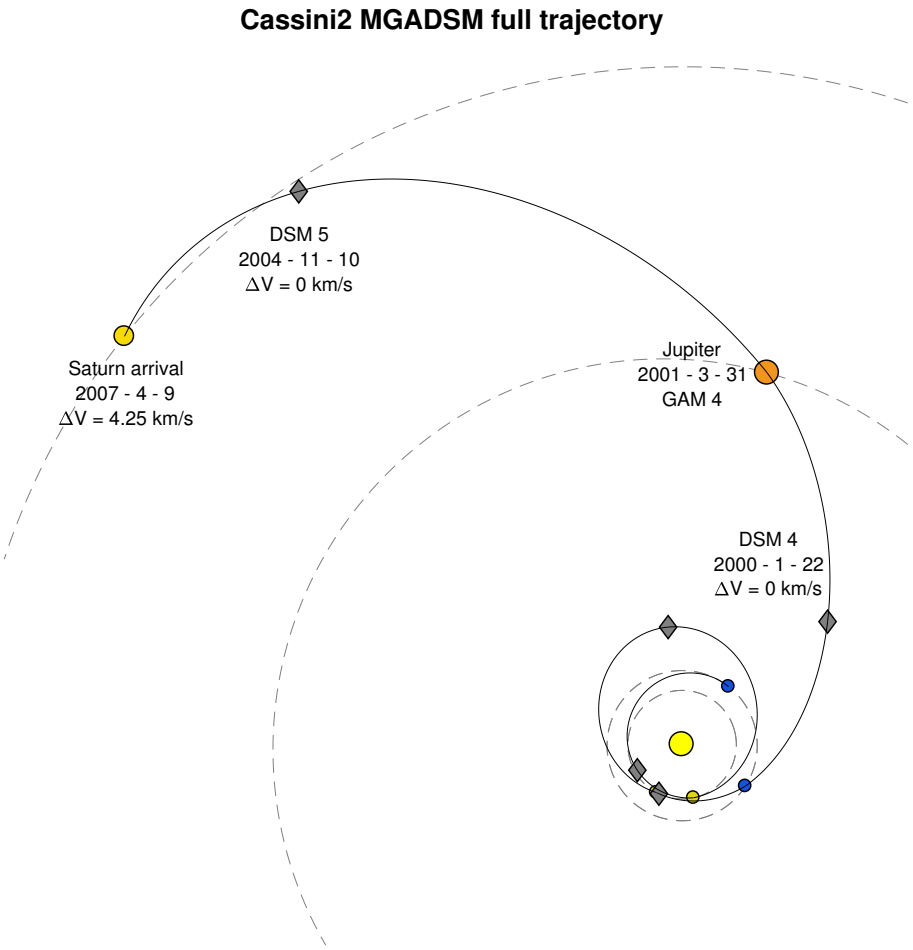


Figure 4.8: Full view of the global minimum trajectory of the Cassini2 MGADSM problem.

Chapter 5

Objective functions

Both the a posteriori and a priori methods can be decomposed into several sub optimization problems. These are identified through a problem analysis of both approaches, consequently it is assessed which of these optimization problems is the most challenging part of the respective method. Finally, objective functions and the associated decision vectors are formulated. In case of the a priori approach, two different formulations are proposed.

5.1 A posteriori problem analysis

The a posteriori approach has been defined in as follows.

> **Definition a posteriori approach:** The a posteriori approach first computes minimum ΔV budget ΔV_{opt} for a range of departure epochs $T0$ (optimal ΔV budget curve). Thereafter, robust ΔV budget ΔV_{rb} is obtained from pairs of ΔV_{opt} values that are departure epoch interval $\Delta T0$ apart.

This definition shows the separation between optimizing ΔV and identifying the minimum robust trajectory pair. This section is structured analogously.

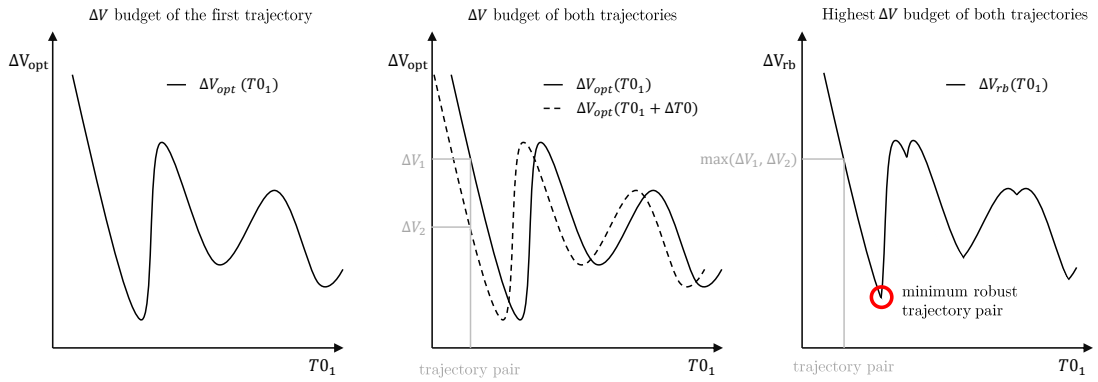


Figure 5.1: Visual representation of the algorithm that finds the minimum robust trajectory pair from the optimal ΔV budget curve $\Delta V_{opt}(T0)$.

5.1.1 Computing the optimal ΔV budget curve

Due to the lack of several closed-form expressions, no general analytical expression for optimal ΔV budget curve $\Delta V_{opt}(T0)$ can be obtained. Therefore, the following numerical approach is taken. Departure epoch $T0$ is discretized into N_{dps} values, for each of which ΔV is optimized (yielding ΔV_{opt}).

Minimization of ΔV with the departure epoch fixed is an optimization problem of dimensionality $D-1$ because $T0$ is not an optimization variable. Here, D is the number of control variables that are provided to the trajectory function (the function that calculates a ΔV budget, given control variables). The dimensionalities of the optimization problems are therefore 2, 5 and 21 respectively on the Cassini0 to Cassini2 problems. The associated objective function is named the a posteriori objective function and the decision vector is denoted by $\mathbf{x_{pt}}$. It can be concluded that to compute the optimal ΔV budget curve, one has to solve N_{dps} times a $D-1$ dimensional optimization problem.

5.1.2 Determining the minimum robust trajectory pair

The robust trajectory pair can be relatively easily obtained from the optimal ΔV budget curve. Figure 5.1 shows graphically how it is found. This figure should be interpreted as follows.

1. The first graph shows a optimal ΔV budget curve of the first departure opportunities.
2. The second figure includes the optimal ΔV budget curve of the associated backup trajectories.
3. The maximum of both ΔV budgets is ΔV_{rb}, depicted in the rightmost graph.

Consequently, minimizing $\Delta V_{rb}(T0_1)$ yields the minimum robust trajectory pair. Taking a strict mathematical definition, the step from the optimal ΔV budget curve to the minimum robust pair comprises of two optimization problems: first a maximization of $\Delta V_{opt}(T0_1)$ and $\Delta V_{opt}(T0_1+\Delta T0)$ for each discrete departure epoch, to generate $\Delta V_{rb}(T0_1)$, and then a minimization of this last function. However, both are one-dimensional optimizations of discrete values and can be solved rapidly by a simple algorithm that finds the extreme values of an array. The challenge of the a posteriori method lies in the N_{dps} optimization problems of dimensionality $D-1$.

The problem analysis is synthesized into the flow diagram of Figure 5.2 and the pseudo code of Algorithm 1.

Algorithm 1 Pseudo code of the a posteriori approach

establish a departure epoch vector **T0**
for each entry of **T0 do**
    $\Delta V_{opt}(T0) := \min(\Delta V(T0, \mathbf{x_{pt}}))$ ▷ differential evolution
end for
establish a first departure epoch vector $\mathbf{T0_1}$
for the each entry of $\mathbf{T0_1}$ **do**
    $\Delta V_{rb} := \max(\Delta V_{opt}(T0_1),\ \Delta V_{opt}(T0_1 + \Delta T0))$ ▷ maximum of a two-value array
end for
min $\Delta V_{rb}(T0_1)$ ▷ minimum value of an array

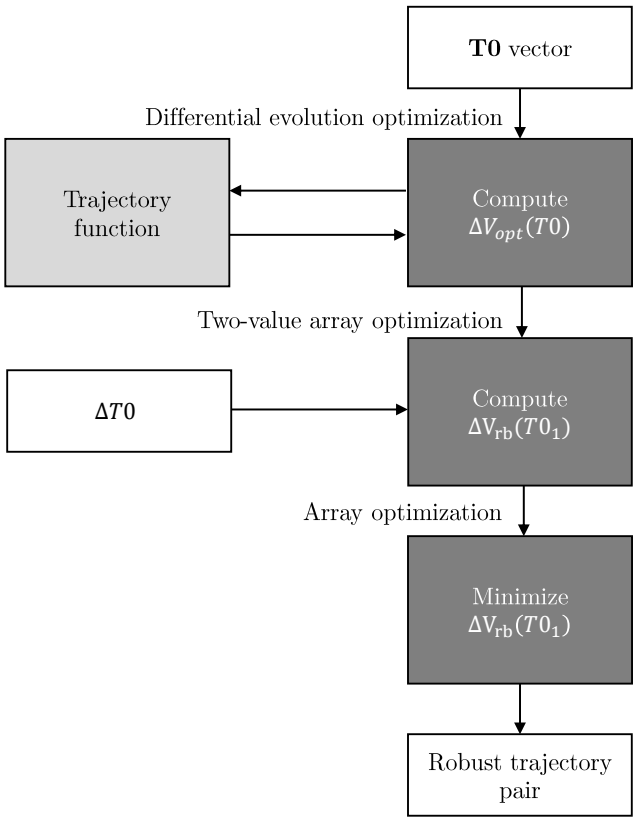


Figure 5.2: Flow diagram of the a posteriori approach.

5.2 A priori problem analysis

The a priori approach has been defined as follows.

> **Definition a priori approach:** The a priori approach minimizes robust ΔV budget ΔV_{rb} directly as a function of the control variables of both trajectories (except the second departure epoch $T0_2$), given a departure epoch interval $\Delta T0$.

By definition, ΔV_{rb} is the maximum of the ΔV's the two trajectories (indicated by ΔV_1 and ΔV_2). These trajectories are each determined by D control variables, thus $2D$ in total. However, $T0_2$ is fixed at an interval $\Delta T0$ from $T0_1$, therefore either one is not free for optimization; $T0_2$ is chosen. This leads to an optimization problem of dimension $2D-1$, so 5, 11, and 43 respectively on the Cassini0 to Cassini2 problems. The associated objective function is named the a priori objective function.

The result is a minimax[1] optimization problem, due to the nested maximization of the two trajectories inside the minimization of ΔV_{rb}. The minimization problem is a one dimensional optimization of two discrete values, arguably the simplest possible form of optimization. Figure 5.3 shows a flow diagram of the a priori approach.

A high-level distinction between two types of formulations for the decision vector of the a priori objective function has been identified. The first formulation describes both trajectories using absolute values of the control variables. It is named the absolute decision vector formulation. An alternative approach is found in the decision vector formulation, where the control variables of the second trajectory are defined relative to the those of the first. This allows one to prune the solution space efficiently, which is addressed in Chapter 7.2.3.

The implementations of both objective functions have been verified by feeding them decision vectors of which the individual ΔV budgets are calculated as well.

5.2.1 Absolute decision vector formulation

The control variables of both trajectories are contained in absolute decision vector $\mathbf{x_{ab}}$, with the exception of $T0_2$. This is illustrated by the following example for the MGA case. The MGADSM problem follows the same structure (it is provided in Table 5.1 at the end of this chapter).

$$\mathbf{x_{ab}} = \begin{bmatrix} T0_1, & T1_1, & T2_1, & \ldots, & Tn_1, & T1_2, & T2_2, & \ldots, & Tn_1 \end{bmatrix} \tag{5.1}$$

The control variables of two trajectories are then extracted as follows. Note that by definition $T0_2 = T0_1 + \Delta T0$.

$$\mathbf{x_1} = \begin{bmatrix} T0_1, & T1_1, & T2_1, & \ldots, & Tn_1 \end{bmatrix} \tag{5.2a}$$

$$\mathbf{x_2} = \begin{bmatrix} T0_1 + \Delta T0, & T1_2, & T2_2, & \ldots, & Tn_2 \end{bmatrix} \tag{5.2b}$$

The procedure is schematically described in Algorithm 2.

[1] A minimax optimization problem is a minimization problem with a maximization problem nested in it. This type of problem is common in game theory where it is often desired to minimize the maximum score of a competitor (Du and Pardalos 2013).

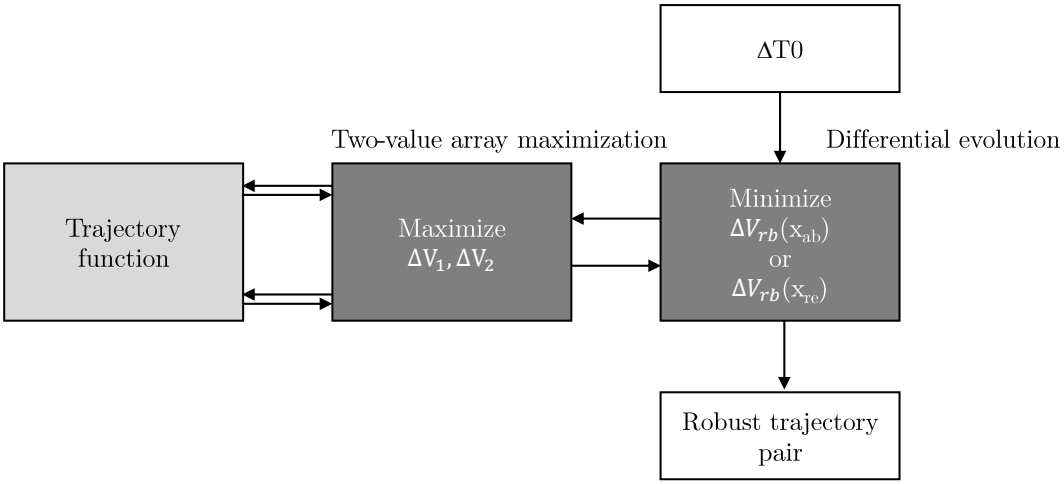


Figure 5.3: Flow diagram of the a priori approach. The difference between $\mathbf{x_{ab}}$ and $\mathbf{x_{re}}$ (middle right block) is explained in the text.

Algorithm 2 Absolute objective function formulation

function $\Delta V_{rb}(\mathbf{x_{ab}})$
    get parameters of first trajectory $\mathbf{x_1} := \mathbf{x_{ab}}\,[1 \text{ to } D]$
    get parameters 2 to last of second trajectory $\mathbf{x_2} := \mathbf{x_{ab}}\,[D+1 \text{ to last}]$
    calculate $T0_2$: $\mathbf{x_2}[1] := \mathbf{x_1}[1] + \Delta T0$
    evaluate ΔV budgets $\Delta V_1 := \Delta V\,(\mathbf{x_1})$ and $\Delta V_2 := \Delta V\,(\mathbf{x_2})$
    define $\Delta V_{rb} := \max\,(\Delta V_1, \Delta V_2)$
end function

5.2.2 Relative decision vector formulation

Instead of using absolute values for the control variables of the second trajectory, one can define them relative to $\mathbf{x_1}$. The resultant variables are named the relative optimization variables. Illustrating this formulation also with an MGA trajectory leaves the following formulation for relative decision vector $\mathbf{x_{re}}$.

$$\mathbf{x_{re}} = \begin{bmatrix} T0, & T1, & T2, & \ldots, & Tn, & \Delta T1, & \Delta T2, & \ldots, & \Delta Tn \end{bmatrix} \tag{5.3}$$

Consequently, the MGA control variables are obtained as follows. Algorithm 3 shows a pseudo code description of the relative objective function.

$$\mathbf{x_1} = \begin{bmatrix} T0, & T1, & T2, & \ldots, & Tn \end{bmatrix} \tag{5.4a}$$

$$\mathbf{x_2} = \begin{bmatrix} T0_1 + \Delta T0, & T1 + \Delta T1, & \ldots, & Tn + \Delta Tn \end{bmatrix} \tag{5.4b}$$

The solution space of the $T0$ to Tn control variables is named the absolute solution space, the solution space of the $\Delta T1$ to ΔTn variables is named the relative solution space. The elegance of this formulation is that it allows boundary constraints on the differences between the control variables. This property is further explored in Section 7.2.3.

Algorithm 3 Relative objective function formulation

1: **function** $\Delta V_{rb}(\mathbf{x}_{re})$
2: get parameters for first trajectory $\mathbf{x_1} := \mathbf{x_{re}}\,[1 \text{ to } D]$
3: calculate $\mathbf{x_2}$ [2 to last] := $\mathbf{x_1}$ [2 to last] + $\mathbf{x_{re}}\,[D+1 \text{ to last}]$
4: calculate $T0_2$: $\mathbf{x_2}[1] := \mathbf{x_1}[1] + \Delta T0$
5: evaluate ΔV budgets $\Delta V_1 := \Delta V\,(\mathbf{x_1})$ and $\Delta V_2 := \Delta V\,(\mathbf{x_2})$
6: define $\Delta V_{rb} := \max\,(\Delta V_1, \Delta V_2)$
7: **end function**

5.3 Conclusion

Both the a posteriori and the a priori approach include multiple sub optimization problems. For the a posteriori approach, the highest dimension is $D-1$, while for the a priori approach it is $2D-1$. An overview of the trajectory, a posteriori, and a priori objective functions is provided in Table 5.1. The entries of each of the decision vectors considered in this work can be found in Table 5.2.

Table 5.1: Overview of the characteristics of three different objective functions.

Function name	**Trajectory**	**A posteriori**	**A priori**
Description	Function that evaluates the ΔV budget of a trajectory (MGA or MGADSM)	Trajectory function with the departure epoch fixed	Objective function that evaluates two trajectories with $\Delta T0$ between the departure epochs
Input	Control variables of the trajectory	Control variables except for $T0$	Control variables of both trajectories except for second departure epoch $T0_2$
Output	ΔV of the trajectory	ΔV of the trajectory	ΔV_{rb}, equal to the highest of the ΔV's of the trajectories
Dimensionality	D	$D-1$	$2D-1$

Table 5.2: Specification of the optimization variables of the various objective functions that are considered in this work.

Cassini0 MGA	**Symbol**	**Dimensionality**	**Optimization variables**
Trajectory	$\mathbf{x}$	3	$T0$ $T1$ $T2$
A posteriori	$\mathbf{x_{pt}}$	2	$T1$ $T2$
A priori absolute	$\mathbf{x_{ab}}$	5	Trajectory 1 optimization variables $T0$ $T1_1$ $T2_1$ Trajectory 2 optimization variables $T1_2$ $T2_2$
A priori relative	$\mathbf{x_{re}}$	5	Absolute optimization variables $T0$ $T1$ $T2$ relative optimization variables $T1_{re}$ $T2_{re}$
Cassini1 MGA			
Trajectory	$\mathbf{x}$	6	$T0$ $T1$ $T2$ $T3$ $T4$ $T5$
A posteriori	$\mathbf{x_{pt}}$	5	$T1$ $T2$ $T3$ $T4$ $T5$
A priori absolute	$\mathbf{x_{ab}}$	11	Trajectory 1 optimization variables $T0$ $T1_1$ $T2_1$ $T3_1$ $T4_1$ $T5_1$ Trajectory 2 optimization variables $T1_2$ $T2_2$ $T3_2$ $T4_2$ $T5_2$
A priori relative	$\mathbf{x_{re}}$	11	Absolute optimization variables $T0$ $T1$ $T2$ $T3$ $T4$ $T5$ relative optimization variables $T1_{re}$ $T2_{re}$ $T3_{re}$ $T4_{re}$ $T5_{re}$
Cassini2 MGADSM			
Trajectory	$\mathbf{x}$	22	$T0$ $\Delta V0$ δ_{LA} α_{LA} $T1$ $T2$ $T3$ $T4$ $T5$ $\eta1$ $\eta2$ $\eta3$ $\eta4$ $\eta5$ $R1$ $R2$ $R3$ $R4$ $\gamma1$ $\gamma2$ $\gamma3$ $\gamma4$
A posteriori	$\mathbf{x_{pt}}$	21	$\Delta V0$ δ_{LA} α_{LA} $T1$ $T2$ $T3$ $T4$ $T5$ $\eta1$ $\eta2$ $\eta3$ $\eta4$ $\eta5$ $R1$ $R2$ $R3$ $R4$ $\gamma1$ $\gamma2$ $\gamma3$ $\gamma4$
A priori absolute	$\mathbf{x_{ab}}$	43	Trajectory 1 optimization variables $T0$ $\Delta V0_1$ δ_{LA1} α_{LA1} $T1_1$ $T2_1$ $T3_1$ $T4_1$ $T5_1$ $\eta1_1$ $\eta2_1$ $\eta3_1$ $\eta4_1$ $\eta5_1$ $R1_1$ $R2_1$ $R3_1$ $R4_1$ $\gamma1_1$ $\gamma2_1$ $\gamma3_1$ $\gamma4_1$ Trajectory 2 optimization variables $\Delta V0_2$ δ_{LA2} α_{LA2} $T1_2$ $T2_2$ $T3_2$ $T4_2$ $T5_2$ $\eta1_2$ $\eta2_2$ $\eta3_2$ $\eta4_2$ $\eta5_2$ $R1_2$ $R2_2$ $R3_2$ $R4_2$ $\gamma1_2$ $\gamma2_2$ $\gamma3_2$ $\gamma4_2$
A priori relative	$\mathbf{x_{re}}$	43	Absolute optimization variables $T0$ $\Delta V0$ δ_{LA} α_{LA} $T1$ $T2$ $T3$ $T4$ $T5$ $\eta1$ $\eta2$ $\eta3$ $\eta4$ $\eta5$ $R1$ $R2$ $R3$ $R4$ $\gamma1$ $\gamma2$ $\gamma3$ $\gamma4$ relative optimization variables $\Delta V0_{re}$ δ_{LAre} α_{LAre} $T1_{re}$ $T2_{re}$ $T3_{re}$ $T4_{re}$ $T5_{re}$ $\eta1_{re}$ $\eta2_{re}$ $\eta3_{re}$ $\eta4_{re}$ $\eta5_{re}$ $R1_{re}$ $R2_{re}$ $R3_{re}$ $R4_{re}$ $\gamma1_{re}$ $\gamma2_{re}$ $\gamma3_{re}$ $\gamma4_{re}$

Chapter 6

Optimization

In the previous chapter, the sub optimization problems of the a posteriori and a priori methods have been identified. The objective functions will be optimized using differential evolution.[1] This optimizer has demonstrated its capability solve various MGA and MGADSM trajectories, but it is sensitive to the settings of its parameters. Therefore, two different variants are tested, leading to the selection of a single variant to be used in the remainder of the research. The structure of this chapter is as follows. First the working principle and implementation of differential evolution is explained in Section 6.1. Thereafter, the performance of the two variants is reported and discussed in Section 6.2, followed by a conclusion in Section 6.3.

6.1 Differential evolution

Differential evolution is a simple, yet very effective, metaheuristic optimization algorithm. It is important to understand how differential evolution works, to appreciate the different performances of the a posteriori and the a priori approach. Therefore, a description of the core optimization loop is provided in Section 6.1.1. Next, Section 6.1.2 and 6.1.3 respectively handle how constraints and stopping criteria are applied. Verified results of the used implementation are presented in Section 6.1.4.

6.1.1 Algorithm description

Differential evolution emulates evolutionary behavior in a population $\mathbf{P}$ with N_P vectors $\mathbf{u}$ that each represent a solution in the N dimensional solution space. They are also referred to as members. In each generation (iteration), a mutant vector $\mathbf{m}$ is created for each member $\mathbf{u}$. The creation of mutants happens through addition and subtraction of random members of the previous generation, although several slightly different procedures for recombination are available. These various procedures are referred to as different strategies, they are explained in more detail below. When a mutant has been created for each member, crossover ratio $CR \in [0, 1]$ determines which entries of the mutant are placed in competing member $\mathbf{c}$. The entries of $\mathbf{c}$ that are not filled by entries of $\mathbf{m}$ are filled with the respective entries of original member $\mathbf{u}$. Two distinct crossover operators are possible, *bin* (binary) and *exp* (exponential) crossover. The *bin* operator does a binomial experiment for each entry individually, the latter assigns entries of $\mathbf{m}$ to $\mathbf{c}$ until the first binomial experiment fails. The experiments consist of checking if a random value between 0 and 1 is lower than crossover ratio CR. Also, crossover is guaranteed for one random entry per member. In

[1] Appendix B describes the implementation of a monotonic basin hopping variant that has been tested on the objective functions but did not lead to any success. This is attributed to the perturbing operator.

general, the difference between the crossover operators is marginal (Price, Storn, and Lampinen 2005). Finally, the fitness of each **c** is compared to the fitness of the associated **m**; when it has improved, **c** will replace **m**. This scheme is looped until a user-defined stopping criterion is satisfied (Storn and Price 1997). A pseudo code transcription of the differential evolution variant that is used in this work is included as Algorithm 4.

Algorithm 4 Pseudo code of differential evolution best/2/exp

```
generate initial population P
evaluate fitness of each member in population P
identify best member b
while both stopping criteria are not satisfied do
    for for each member u do
        pick randomly four members d, e, f and g
        create mutant m := b + F (e − f + g − h)
        for each dimension, starting randomly, do
            while a random value between 0 and 1 < CR do
                assign value of mutant m to competing member c
            end while
            assign values of u to c for remaining entries
            ensure crossover at random entry
        end for
        calculate fitness of competing member c
        if fitness c is better than fitness u then
            replace u by c
        end if
    end for
    identify and save best member b
    check stopping criterion 1: calculate the average distance
    check stopping criterion 2: count number of stalled generations
end while
```

Strategies for mutant creation

Strategies for differential evolution are indicated in the format *a/n/b*. Here, *a* refers to the base member of each mutant (**b** in Algorithm 4), which can either be a random member (*rand*), the best member of the population (*best*) or a combination of both (*rand-to-best*). Next, *n* indicates using how many vector pairs the perturbing term of **m** is created. Generally *n* is either 1 or 2, this corresponds to respectively 2 or 4 vectors (**e** to **h** in Algorithm 4). Finally, *b* indicates the crossover strategy: either *bin* or *exp*.

Design parameters

Besides selecting the strategy, differential evolution requires the user to set values for three parameters. They are explained below.

1. Scaling factor $F \in [0, 2]$ determines the magnitude of the perturbing term of **m** (see Algorithm 4). Higher scaling vectors correspond to more exploratory behavior.

2. Crossover ratio $CR \in [0, 1]$ determines the chance that an entry from **m** goes to **c**. Lower values result in a higher convergence (Zaharie 2009). This potentially reduces the computational

effort but also increases the chance on premature convergence.

3. Population size N_P. Many different population sizes for the MGA and MGADSM trajectories have been proposed, from a fixed number of 20 (Vinko and Izzo 2008) to $20N$ (Englander 2013) and many values in between. Small populations have little diversity and will experience difficulty exploring the solution space exhaustively. On the other hand, large populations have a tendency to converge into local minima (Price, Storn, and Lampinen 2005). This behavior has been observed during this study as well.

Proposed variants

The strategy and design parameter settings have a decisive impact on the performance of differential evolution. Values that are generally considered good first guesses are $F = 0.8$, $CR = 0.9$ and $N_P = 10N$ combined with the *DE/rand/1/bin* strategy. However, several studies have yielded significantly enhanced performance on the MGA and MGADSM trajectories, either by tuning parameters to a specific problem or by creating self-adapting schemes (Vasile, Minisci, and Locatelli 2011; Musegaas 2012; Zuo, Dai, and Peng 2017). In this respect, especially the work of Musegaas (2012) is valuable. He was able to improve the performance of differential evolution significantly by tuning the design parameters. These results still represent the best attained performance of differential evolution on the Cassini1 and Cassini2 trajectories. Therefore, the two best performing variants of Musegaas are selected to be tested for use in this work. The settings are summarized in Table 6.1

Table 6.1: The two most promising differential evolution variants, proposed by Musegaas (2012).

	Strategy	**F**	**CR**	$\mathbf{N_P}$
DE variant 1	*best/1/exp*	0.7	0.9	$4.5 \cdot N$
DE variant 2	*best/2/exp*	0.5	0.94	$3 + 3.7 \cdot N$

6.1.2 Constraint handling

The MATLAB implementation of differential evolution that has been made available by Rainer Storn (one of its inventors) is used.[2] A limitation of this implementation is that it does not for enforce boundary constraints. Therefore, a bounce-back mechanism is implemented; members are bounced back into the solution space by a magnitude equal to the boundary constraint violation. Trials showed that the effect of using of a different constraint handling strategy, reinitialization in the full range of the respective dimension (as used in PaGMO[3]), is negligible on the considered problems. This insignificance of the precise boundary handling strategy has also been described by Price, Storn, and Lampinen (2005). The nonlinear constraints on the flyby radius in the MGA trajectory model are handled by a penalty function, described in Section 4.2.3.

Algorithm 5 Equal bounce-back constraint handling

for each dimension i **do**
  if entry c_i of competing member **c** violates entry q_i of boundary constraint **q then**
    $c_i := 2q_i - c_i$
  end if
end for

[2] It is named `devec3.m` and can be downloaded from http://www1.icsi.berkeley.edu/~storn/code.html.
[3] A bundle of implementations of optimizers that has been developed by ESA https://esa.github.io/pagmo2/

6.1.3 Stopping criteria

Differential evolution converges after a number of generations. The definition of the performance indicator stated in Chapter 3 rewards quick termination of runs after convergence, because $N95$ is linearly dependent on the average number of function evaluations $\overline{N_{fe}}$. Two stopping criteria are used in this research. The first criterion is the average distance between population members, the loop is terminated when it drops below a threshold. The second criterion counts the number of generations without significant improvement and stops the optimizer when it exceeds a limit. Both stopping criteria are explained below.

Average distance calculation

A comparison of the average distance between the members of the population to a user-specified limit, is a very powerful criterion that can be used to declare convergence (Vasile, Minisci, and Locatelli 2008). Since reproduction occurs by adding and subtracting members (vectors), the population is unable to move away from a minimum after the average distance between members drops below a certain threshold. The computational effort of calculating this average distance is considerable since the number of combinations of members N_c is a function of the population size N_P, it adheres to the following relation.

$$N_c = \frac{N_P\,(N_P - 1)}{2} \tag{6.1}$$

To mitigate the computational effort, the average distance between the members is calculated once every ten generations. For proper comparison, the solution space needs to be scaled. In the algorithm developed for this research, it is scaled down to the range [-1, 1]. One should also note that the distance between members increases with the dimensionality of the solution space. Distance d between a member $\mathbf{e}$ and a member $\mathbf{f}$ in an N dimensional space is calculated as follows.

$$d = \sqrt{\sum_{i=1}^{N}(e_i - f_i)^2} \tag{6.2}$$

Pseudo code of the average distance stopping criterion is included as Algorithm 6.

Algorithm 6 Algorithm that calculates the average distance between members

```
let population matrix P have N_P rows and N columns
initialize total distance as 0
calculate number of combinations N_c (Equation 6.1)
for i = 1 to N_P − 1 do
    A := P(first to last - i row)
    B := P(first + i to last row)
    C := A - B
    for each row of C do
        take the root-sum-square (rss) and add to total distance
    end for
end for
divide the total distance by number of combinations
verify that the number rss values is equal to N_c
```

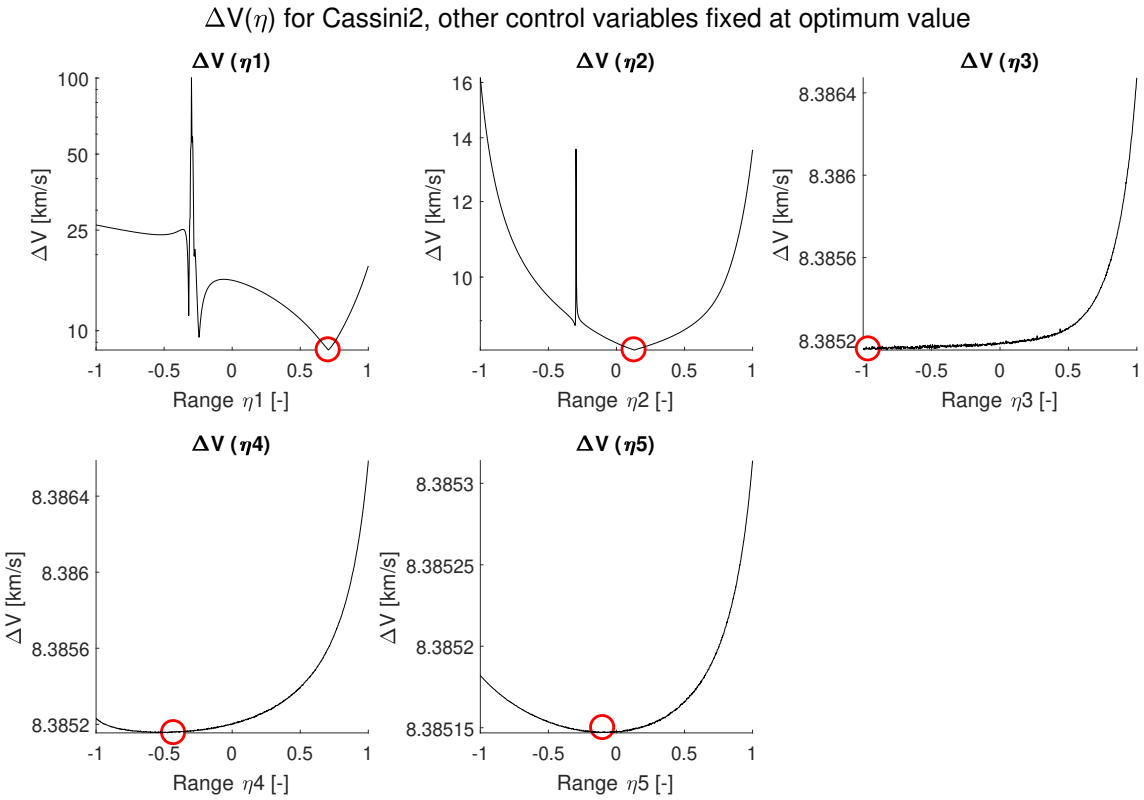


Figure 6.1: Cross-sections of the solution space along the DSM ratios of the global optimum. The (small) oscillations near the minima of $\eta 3$ to $\eta 5$ are likely due to the iterations of the Lambert and Kepler propagators (see Section 4.1).

Insensitivity of DSM ratios

During trials with the stopping criterion on the Cassini2 trajectory, an undesirable property of the MGADSM trajectory function revealed itself. As shown in Chapter 4, the solution of the Cassini2 problem includes three DSMs of negligible magnitude; the velocity increments range between 0.036 m/s and 0.20 m/s. The result of these low velocity increments is that the associated DSM ratios η have no significant impact on the ΔV budget, thereby eliminating the mechanism that leads to convergence of the DSM ratios. That in turn makes that the average distance has much difficulty reaching values lower than 0.01 in the normalized solution space. This insensitivity of $\eta 3$ to $\eta 5$ is illustrated by Figure 6.1. It shows cross sections of the solution space along the DSM ratios through the global optimum. This issue has been solved by not taking the DSM ratios into account in the average distance stopping criterion. This reduces the number of dimensions in which the average distance is calculated from 22 to 17 for the Cassini2 trajectory.

Convergence of only one trajectory in the a priori approach

The fact that ΔV_{rb} is determined only by the highest ΔV of the robust trajectory pair has a similar effect on the minimum distance stopping criterion, as the DSM ratios. The control variables of the trajectory with the lowest ΔV do not affect ΔV_{rb}, as long as this trajectory remains fitter than the other trajectory. Therefore, these control variables can navigate relatively freely through the solution space. Because this hinders simultaneous convergence of all control variables, the stopping criterion only takes the distances between control variables of the momentarily fittest trajectory into account, where $T0_1$ is considered to be a control variable of both trajectories. Therefore,

the numbers of control variables that are considered for calculating the average distances are respectively 3, 6 and 17 for the test problems.

Average distance limit values

The numerical values of the average distance stopping limits are 0.001, which is adopted from Musegaas (2012). These values are somewhat conservative as many observed objective functions reach the global minimum when the average distance is around 0.01. However, most populations reach an average distance of 0.001 soon after 0.01. An exception is Cassini2, of which convergence is very slow below 0.01. Therefore, the following additional stopping criterion is defined as well.

Maximum stall generations stopping criterion

A second stopping criterion is implemented in the differential evolution algorithm in the form of a maximum number of stalled generations MSG. That is, the algorithm stops if it fails to improve by more than fitness threshold FT over MSG generations. The following values are used in this research, they are adopted from Englander (2013).

1. Minimum fitness improvement threshold $FT = 1 \cdot 10^{-6}$ km/s.
2. Maximum number of stalled generations $MSG = 10 \cdot N$.

This stopping criterion has only been satisfied frequently on the Cassini2 problem.

Discussion on stopping criteria

The stopping criteria have a direct influence on the performance of the algorithm, through average number of function evaluations $\overline{N_{fe}}$. If they are too lenient, the average number of function evaluations is unnecessarily high, but when they are too strict, the success rate suffers. This suggests that an optimum exists. Specifically optimizing the stopping criteria has not been addressed in this research, but it is a field in which further incremental improvement of the performance is expected to be possible. For a fair comparison between the two different differential evolution variants that are tested in this chapter, it is deemed most important that all are subjected to the same stopping criteria, which requirement has been fulfilled by the approach proposed in this section.

6.1.4 Verification of the differential evolution algorithm

Although `devec3.m` has been used in many studies, a peculiar difference between this code and the description provided by Storn and Price (1997) came to light when it was attempted to reproduce results on two standard optimization testing functions, the Rosenbrock saddle and the Rastrigin function.[4] That is, `devec3.m` lacks the mechanism that ensures that at least one random entry from $\mathbf{m}$ is crossed over to $\mathbf{c}$. This does not lead to large performance differences for high CR and D values, but is critical when CR is equal to zero. Inspection of the source code of the differential evolution implementation in PaGMO revealed that here the minimum crossover mechanism is used only in the exponential crossover variant. Again, this differs from the implementation proposed by Storn and Price (1997). We may conclude that there is no consensus in the scientific community whether or not to use the minimum crossover mechanism. In this work it is included, but the high dimensionality of the solution spaces and high CR values selected in Section 6.1.1 limit its effect.

After having modified `devec3.m` such that it matches the descriptions in the respective publications precisely, it was possible to reproduce results from literature. An overview can be found

[4] Equations are provided in Appendix C

in Table 6.2. The first problem counts the number of function evaluations to reach a value (0.9), the second reports the fitness after 1000 iterations. The performance measure n_{95} that is used on the third verification function is not the same as $N95$. The difference is that n_{95} is calculated for each successful run individually, using the number of function evaluations of that run until VTR is reached (Olds, Kluever, and Cupples 2007).

The stopping criterion calculation scheme (Algorithm 6) is verified in the code itself (line 13), as well as by a thorough review and by feeding a population matrix $\mathbf{P}$ of four members with all combinations of coordinates -1 and 1 in two dimensions, yielding an average distance of 2.2761. This is verified analytically. Algorithm 5 is of such limited complexity that careful inspection of the code suffices as verification of its functionality.

Table 6.2: Three problems using which the implementation of differential evolution is verified. Sources: Storn and Price (1997), Mezura-Montes, Velázquez-Reyes, and Coello Coello (2006), and Musegaas (2012).

Settings	**Rastrigin**	**Rastrigin**	**Cassini1**
Strategy	rand/1/bin	best/1/exp	best/2/exp
Dimension	20	30	6
Initial parameter range	[-600, 600]	[-5.12, 5.12]	Section 4.3.2
Bounds	No	No	Section 4.3.2
Value-to-reach	0.9	$-\infty$	4.98
Population size	25	60	25
Scaling factor	0.5	0.5	0.5
Crossover ratio	0	0.8	0.94
Max iterations	∞	1000	∞
Performance measure	N_{fe}	fitness	n_{95}
Verification			
Average value	13171	39.9391	431
Sample size	1000	100	100
Literature			
Average value	12971	40.0040	412
Sample size	20	100	400

6.2 Performance on the objective functions

Both proposed differential evolution variants have been tested on objective functions that are used in this study. In this section, the setup of these trials is explained first and consequently the results are presented in Section 6.2.2.

6.2.1 Experimental set-up

The two variants have been tested on the trajectory and a priori objective functions of each of the three test problems (so six in total). The solution spaces of the a posteriori objective functions are subsets of those of the trajectory functions and therefore not tested separately. The VTR that should be reached for declaring success varies per objective function, these values are included as well as the ΔV budget of the global minimum ΔV_{min}. It can be observed that the margins are

reasonably strict on the Cassini0 and Cassini1 trajectories, while a more lenient VTR is applied to the Cassini2 trajectory because of its high dimensionality.

Table 6.3: Performance indicator N95 for the two variants of differential evolution.

Objective function		N	ΔV_{min}	VTR	**DE1**	**DE2**
		-	km/s	km/s	$N95$	$N95$
Cassini0 MGA	Trajectory	3	9.352	9.4	$5.34 \cdot 10^3$	$5.01 \cdot 10^3$
	A priori	5	9.517	9.6	$2.86 \cdot 10^4$	$2.46 \cdot 10^4$
Cassini1 MGA	Trajectory	6	4.931	5.0	$8.15 \cdot 10^5$	$8.64 \cdot 10^5$
	A priori	11	5.028	5.1	$2.10 \cdot 10^8$	$1.52 \cdot 10^8$
Cassini2 MGADSM	Trajectory	22	8.385	8.5	$6.47 \cdot 10^8$	$3.81 \cdot 10^8$
	A priori	43	8.455	8.8	No success	No success

6.2.2 Optimization results and discussion

The results in terms of $N95$ are shown in Table 6.3. Between the differential evolution variants, it can be concluded that DE2 performs overall better than DE1. The slightly better performance of DE1 on the Cassini1 trajectory function is deemed insignificant, therefore DE2 is selected as the differential evolution variant to be used in this work. Table 6.3 also shows that both variants are capable of reaching the VTR values on three out of four problems, but not on the 43-dimensional Cassini2 a priori objective function. The fact that neither variant found a solution that exceeded the VTR (the best results were 10.309 km/s and 9.681 km/s respectively), despite the lenient threshold, is somewhat disappointing. On the other hand, this 43 dimensional optimization problem is rather difficult.

In Table 6.4, several extra parameters are provided for both differential evolution variants. These are success rate p, standard deviation σ_p of the success rate and the average number of function evaluations $\overline{N_{fe}}$. The following observations have been made.

1. Uncertainty (σ_p) is relatively large due to the low success rates on the higher dimensional objective functions. This makes the decision between DE1 or DE2 less certain on those objective functions. Still larger samples lead to intractable computation times.

2. Performance indicator $N95$ increases with N. For the trajectory function, the increase is approximately a factor 200 from Cassini0 to Cassini1, and approximately 500 from Cassini1 to Cassini2 (both for DE2). This increase of $N95$ is limited, compared to the exponentially expanding solution spaces. It illustrates the favorable behavior of metaheuristic optimization on solution spaces of high dimensionality.

3. The number of function evaluations of DE1 and DE2 are similar on the Cassini0 and Cassini1 trajectories, but different on the Cassini2 objective functions. This can be attributed to the fact that one the former two, most optimization runs are terminated by the average distance stopping criterion, while on the Cassini2 problem the maximum number of stalled generations is leading. DE2 stalled sooner than DE1.

The Cassini0 problem has been generated for this study, so no verification data is available. However, the Cassini1 and Cassini2 problems have been generated by Izzo (2010) and the performance of differential evolution on their trajectory functions has been studied before (Vasile, Minisci, and Locatelli 2010; Musegaas 2012). The low success rates of both are subscribed. As the verification of Section 6.1.4 has demonstrated, the success rate reported for DE2 on Cassini1 complies with value

described by Musegaas (2012). The maximum number of stalled generations stopping criterion that is applied to Cassini2 makes that a direct comparison is not possible for Cassini2.

Table 6.4: Success rate p, its standard deviation σ_p and the number of function evaluations N_{fe} for both differential evolution variants that have been tested.

Objective function				**DE1**			**DE2**		
		N	Samples	p	σ_p	N_{fe}	p	σ_p	N_{fe}
Cassini0 MGA	Trajectory	3	$5 \cdot 10^3$	44.0%	0.70%	$1.03 \cdot 10^3$	49.4%	0.71%	$1.14 \cdot 10^3$
	A priori	5	$5 \cdot 10^3$	32.4%	0.66%	$3.73 \cdot 10^3$	37.6%	0.69%	$3.88 \cdot 10^3$
Cassini1 MGA	Trajectory	6	$2 \cdot 10^4$	1.45%	0.08%	$3.97 \cdot 10^3$	1.37%	0.08%	$3.98 \cdot 10^3$
	A priori	11	$2 \cdot 10^4$	0.040%	0.014%	$2.80 \cdot 10^4$	0.055%	0.017%	$2.79 \cdot 10^4$
Cassini2 MGADSM	Trajectory	22	$1 \cdot 10^4$	0.120%	0.035%	$2.36 \cdot 10^5$	0.100%	0.032%	$1.22 \cdot 10^5$
	A priori	43	$1 \cdot 10^4$	0%	-	$7.89 \cdot 10^5$	0%	-	$6.85 \cdot 10^5$

6.3 Conclusion

In this chapter, the differential evolution optimization algorithm that is used on the objective functions has been explained. Optimization is terminated if the average distance between the members drops below 0.1% of the normalized side length of the solution space, or if the fitness has not improved more than 10^{-6} km/s over $10N$ generations. Also, appropriate settings for the differential evolution optimizer have been selected. Two different variants have been tested, leading to a small performance difference. The optimal settings have been found to be scaling factor $F = 0.5$, crossover ratio $CR = 0.94$ and population size $N_P = 3 + 3.7 \cdot N$ for a *best/2/exp* strategy. An area of improvement is the formulation of the stopping criteria.

Chapter 7

Pruning and biasing

An important characteristic of the robust trajectory pairs has, until now, remained unexplored: the similarity between paired trajectories. The extend of this similarity depends on the departure epoch interval; for the three weeks considered in this study, it promises potential for improving the optimization performance. Five methods for pruning the solution space and biasing the optimizer are proposed. An overview of these similarity exploitation methods is presented in Figure 7.1, the methodology stated in Section 3.3 has been used to generate the concepts. A notable feature of Figure 7.1 is that there is no method that biases the optimization process of the a posteriori approach; no viable concept has been generated in this category. Furthermore, there are two algorithms of which the design is based on a performance comparison of several options. The first is the initial guess generator of the a posteriori method, here four different variants in two degrees of freedom are compared. The second is the variable mirror operator; it includes a parameter of which the value needs to be tuned. This is done through a grid search on Cassini1.

It is challenging to improve the computational efficiency while avoiding to exclude potentially optimal solutions. Referring back to the no free lunch theorem (Wolpert and Macready 1997), it is desired that improved performance on robusttrajectorypairs is payed for by a worse performance on irrelevant problems. However, a difficulty is that it is *likely* that the trajectories of a robust trajectory pair are similar, but not guaranteed. The proposed methods each handle this ambiguity differently. In the following sections, first the a posteriori (green blocks) and then the a priori (blue blocks) methods are explained.

7.1 A posteriori approach

Two measures are proposed to enhance the performance of the a posteriori approach. The first biases optimization by providing an initial guess for the generation of ΔV_{opt} values of neighboring departure epochs. The second prunes the solution space to a section around the global minimum ΔV_{min}. Both are discussed respectively in the following two sections.

7.1.1 Generating an initial guess (biasing)

The similarity of trajectories with a limited interval between their departure epochs has been recognized by Knittel et al. (2017) and used in PEATSA (remember that this is an automated sensitivity analysis algorithm that calls to EMTG). However, the precise implementation of this algorithm remains unclear. Therefore, in this section an independently developed method is proposed.

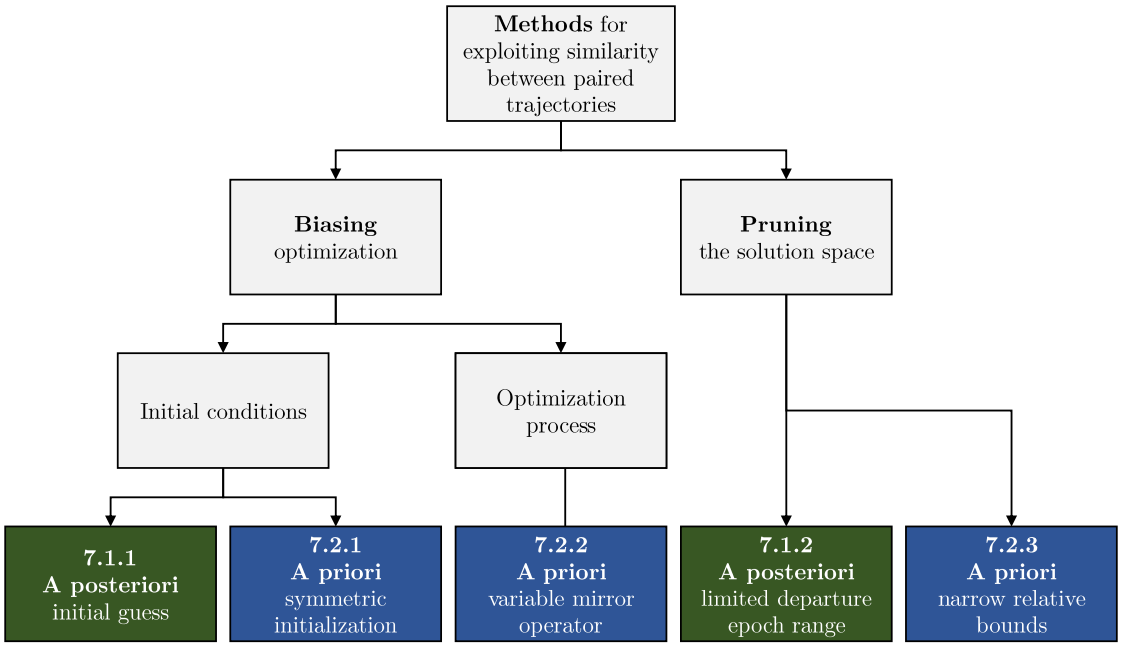


Figure 7.1: Tree structure with the similarity exploitation methods presented in this chapter, the methods are ordered conceptually. The green blocks are a posteriori methods, the blue blocks a priori methods.

The impact of the following two aspects on the quality of the initial guess is researched.

1. The extrapolation method used to generate the initial guess
2. The departure epoch step size $\delta T0$

Both characteristics are addressed below. After that, the performances of four algorithms with different settings, are compared. The results are used to synthesize a final initial guess algorithm.

Extrapolation method

Two different extrapolation methods are considered. The first strategy simply uses the solution of the previously optimized departure epoch $\mathbf{x_{i-1}}$ as initial guess $\mathbf{x_{i_{ig}}}$ for the current time step. It is named the previous solution strategy. For completeness, it is mathematically transcribed below.

$$\mathbf{x_{i_{ig}}} = \mathbf{x_{i-1}} \tag{7.1}$$

A second proposed strategy extrapolates the past two solutions linearly to make an initial guess for the current departure epoch, hence it is named the linear extrapolation strategy. It is implemented as follows.

$$\mathbf{x_{i_{ig}}} = \mathbf{x_{i-1}} + (\mathbf{x_{i-1}} - \mathbf{x_{i-2}}) \tag{7.2}$$

The strategies are graphically illustrated in Figure 7.2 for both forward and backward propagation. Note that this optimal ΔV budget curve does not correspond to an actual trajectory.

Departure epoch step size

A second factor that has a major impact on the quality of initial guesses is step size $\delta T0$ of the departure epoch; the smaller, the more accurate initial guesses are. On the other hand, each

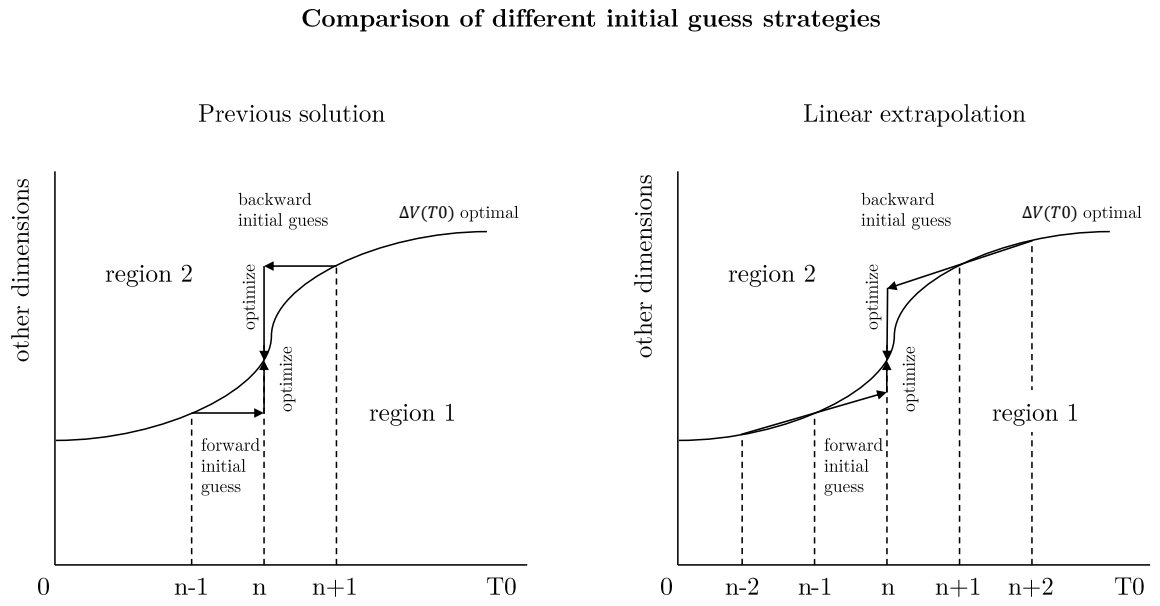


Figure 7.2: Graphical interpretation of the previous solution an linear extrapolation initial guess strategies, for generating an initial guess for departure epoch n.

discrete departure epoch value requires an optimization problem to be solved. Also, the departure epoch interval imposes a requirement on the step size, the latter should fit a positive integer times in the former. These considerations can be synthesized into the hypothesis that an optimum value (in terms of minimum computation time) for the step size exists, although this optimum depends on the local properties of the solution space. Since a variable step size would lead to high complexity, a constant departure epoch step size is used. Two step sizes are tested: 1 and 0.1 day.

Comparison of different initial guess algorithms

Taking each combination of the two extrapolation methods and two step sizes yields four different initial guess algorithms. Their performances been evaluated on a 12 week interval around the Cassini1 global minimum. The performance measure of the algorithms is the difference between the ΔV budget (fitness) of the initial guesses and the minimized ΔV budget at that epoch,[1] also referred to as the error. Furthermore, each variant is tested using both forward and backward propagation of $T0$; differences between the performances in different directions can be large. Consequently, the conclusions drawn from the performance on this section are generalized for the Cassini0, Cassini1 and Cassini2 solution spaces. This extrapolation avoids a costly assessment of all methods on all trajectories, but introduces the assumption that the twelve week window of Cassini1 is representative for the solution spaces. This window contains two different basins and on one switch between these basins; the local characteristics of the solution space have influence on the results. Anticipating the results of Chapter 8, no clear indications of the invalidity of the extrapolation have surfaced. Still, there are various starting points for future, more elaborate, studies on initial guess algorithms. The results on the twelve week window are graphically displayed in Figure 7.3. In this figure, the optimal ΔV budget curve has a basin switch at around $T0 = -770$ MJD2000. The following observations have been made.

[1] It can be claimed with confidence that the obtained curve (see Figure 7.3) is globally optimal, it is smooth and includes the global minimum described by (Izzo 2010).

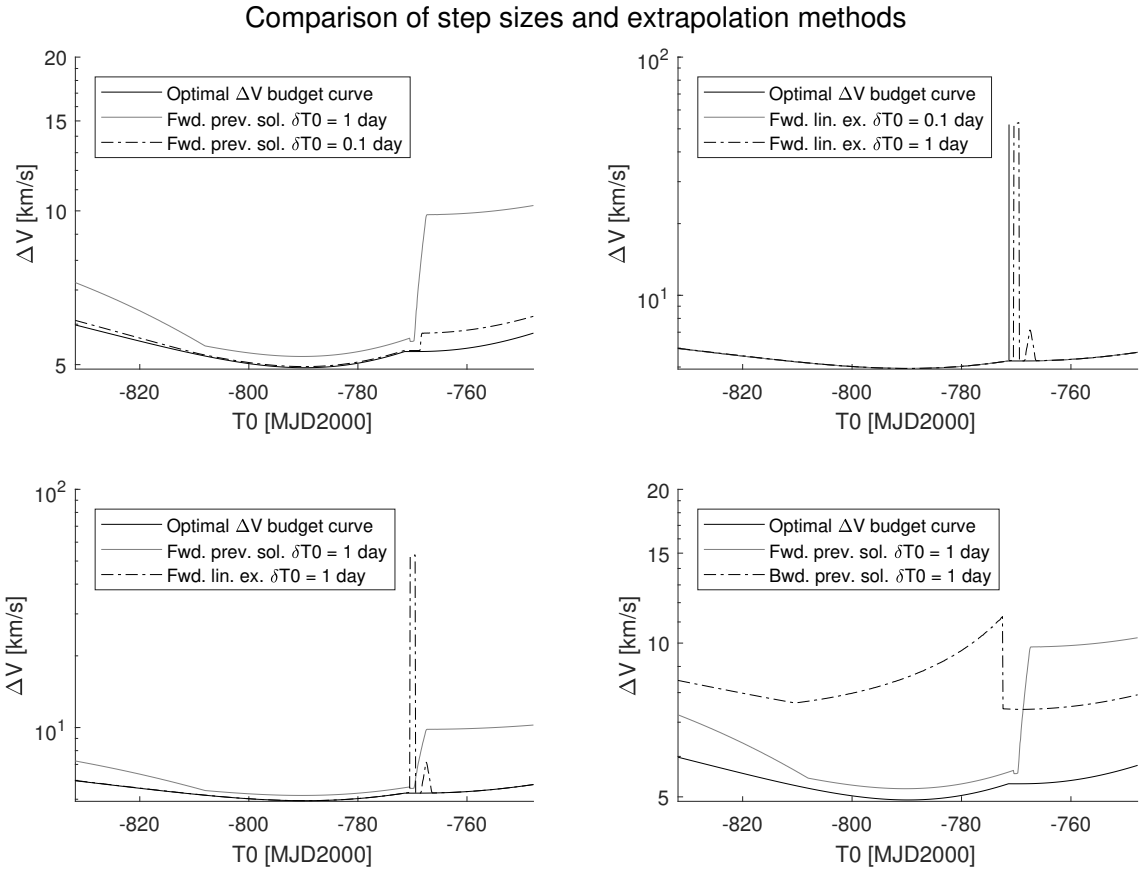


Figure 7.3: Comparison of step sizes and extrapolations methods on a twelve week window around the global minimum of Cassini1.

1. The accuracy changes with the basin switch. In the top left graph it can be seen that the resolution of 1 day yields rather poor initial guesses for $T0 > -770$ MJD2000.
2. The top right figure illustrates that linear extrapolation provides a poor initial guess around the basin switch. The magnitude of the error is about the same for both 1 and 0.1 day resolution.
3. The bottom left figure shows that the error of the linear extrapolation strategy is much larger than the error of the previous solution strategy around the basin switch (note the logarithmic y-axis).
4. The bottom right figure shows that the at the basin switch, the relative performance of the propagation direction switches. In other words: in the left basin, forward propagation performs best while in the right basin, backward propagation performs best.

Table 7.1 provides quantitative insight into the four different methods. Two important metrics are the median and maximum errors. The former is an indicator of the accuracy within a basin while the latter indicates how bad the performance is around basin switches. It is desired to have a low value for both.

It is concluded that the previous solution strategy with a resolution of 0.1 days provides the best performance. It generates reasonably accurate initial guesses in both propagation directions.

Table 7.1: Comparison between the median and maximum errors of eight different initial guess strategies around the global minimum of Cassini1

Direction	**Strategy**	**Resolution**	**Median error**	**Max error**
-	-	days	km/s	km/s
Forward	Previous solution	0.1	$3.2 \cdot 10^{-2}$	$4.6 \cdot 10^{1}$
Forward	Previous solution	1.0	$4.2 \cdot 10^{-1}$	$4.5 \cdot 10^{0}$
Forward	Linear extrapolation	0.1	$3.6 \cdot 10^{-5}$	$4.7 \cdot 10^{1}$
Forward	Linear extrapolation	1.0	$1.3 \cdot 10^{-3}$	$4.8 \cdot 10^{1}$
Backward	Previous solution	0.1	$2.5 \cdot 10^{-1}$	$5.9 \cdot 10^{1}$
Backward	Previous solution	1.0	$2.4 \cdot 10^{0}$	$6.0 \cdot 10^{0}$
Backward	Linear extrapolation	0.1	$3.7 \cdot 10^{-5}$	$1.5 \cdot 10^{2}$
Backward	Linear extrapolation	1.0	$1.2 \cdot 10^{-3}$	$1.5 \cdot 10^{2}$

Linear extrapolation is much more accurate within a basin, but performs worse around the basin switch. Therefore, the previous solution strategy is selected as the method of choice.

Initializing method

The use of an initial guess is somewhat uncommon in differential evolution, or any global optimization method indeed. Often, global optimization efforts are undertaken precisely when there is no initial guess. Still, the algorithm allows fairly easily for an initial guess to be included. In this study it is simply inserted into the initial population, the rest of the initial members are still uniformly generated over the solution space. This assures a very diverse population to maximize the chances of identifying a potentially more optimal basin. An alternative method would be a Gaussian initialization around the initial guess (Kazimipour, Li, and Qin 2014a). Two disadvantages are that it includes a parameter (the standard deviation) that needs to be tuned, and centering the population around the initial guess reduces diversity, thus the chance to identify a more optimal basin. For these reasons, the first approach is used.

Automated retries

While the initial guess strategy increases the chances of finding the global minimum ΔV for a given departure epoch, the success rate is still not 100% for each departure epoch. Several measures for dealing with this, while minimizing number of function evaluations, are proposed. The first is focused on retrying optimization. This is done if the following conditions are true.

1. The slope between the current and the previous departure epoch (numerical approximation of the first derivative $\frac{\partial \Delta V_{opt}}{\partial T0}$) exceeds 1 km/s per day.

2. The change between the current and previous slopes (numerical approximation of the second derivative $\frac{\partial^2 \Delta V_{opt}}{\partial T0^2}$) exceeds 1 km/s per day per day.

The magnitudes of the thresholds have been obtained through trial and error. The conditions are only true for a sudden and large deflection upwards, which is interpreted as a failure to solve the optimization problem. Sudden and large deflections downwards are allowed, these occur when there is switched from a suboptimal curve to a more optimal one (see Figure 7.5, it is explained below). To avoid getting stuck in an infinite loop of retries, the number of retries is limited, the limit is set on ten. Areas in which this limit is reached are later revisited for propagation using a different extrapolation method and/or from a different direction.

Entry points

The initial guess algorithm makes ΔV_{opt} values of the optimal ΔV budget curve consecutively dependent: following departure epochs rely on previous ones for their initial guesses. This consecutive generation of the optimal ΔV budget curve is referred to as propagation. Furthermore, the points from which propagation is started, are named entry points.

While the initial guess algorithm has a good performance for propagation within a basin, it has been seen that it does not always cope well with basin switches. An accurate initial guess gives the optimizer a strong bias to remain in the current basin, while, as propagation advances, other basins may become optimal. The initial guess propagation technique has been found to identify more optimal basins late, and sometimes not at all. This weakness is handled by starting propagation from multiple entry points. These points are generated as follows.

The departure epoch range is divided into N_s (10 in this study) sections and on each of these sections a number of trajectory function optimization runs is done. The number of entry points is decided to be equal to the number of runs required to be 99% sure that the VTR has been reached by the trajectory function. Using the relations described in Section 3.5, number of entry points N_{eps} is calculated as follows, with p the success rates (respectively 49.4%, 1.37%, and 0.1%) and $p_{thresh} = 0.99\%$.

$$N_{eps} = \frac{\log\left(1 - p_{thresh}\right)}{\log\left(1 - p\right)} \tag{7.3}$$

Rounding leads to 10, 320 and 4600 entry points (in total) on Cassini0 to Cassini2. Propagation of the optimal ΔV budget curve begins at the fittest entry point (the global minimum) and moves to the upper boundary of the departure epoch range, thereafter the procedure is repeated from the fittest entry point to the lower boundary. When, during propagation, entry points with a better ΔV than the initial guess are encountered, these replace the initial guess. Figure 7.4 shows an example of a (fictional) fully optimized, optimal ΔV budget curve and the entry points from which it has been generated. Not every entry point has converged to the optimal ΔV budget curve. This is because differential evolution not always converges to the global minimum.

The aforementioned consecutive dependency of the optimizations is also a vulnerability. Failure to converge to the minimum of certain departure epoch will affect all following departure epochs through bad initial guesses. This is an inherent weakness of the initial guess method.

Smoothening

When the ΔV has been minimized for each departure epoch, there may remain spiky regions as well as too late transitions between basins, recognizable by the sudden drops of the optimal ΔV budget curve. In these areas, the optimization algorithm is rerun in a different direction and/or using a different propagation strategy, until all the transitions are smooth. Figure 7.5 shows an example of smoothening of a too late basin transition. This procedure has not been fully automated, smoothening is started manually.

Algorithm verification

The algorithm has first been verified by doing a thorough inspection. Furthermore, the extrapolation mechanisms are implemented modularly, they are verified by feeding simple examples of which the outcome is known. Also, the algorithm has been designed such that the values of variables (the fitness of the initial guess, number of tries per departure epoch, value of the minimized

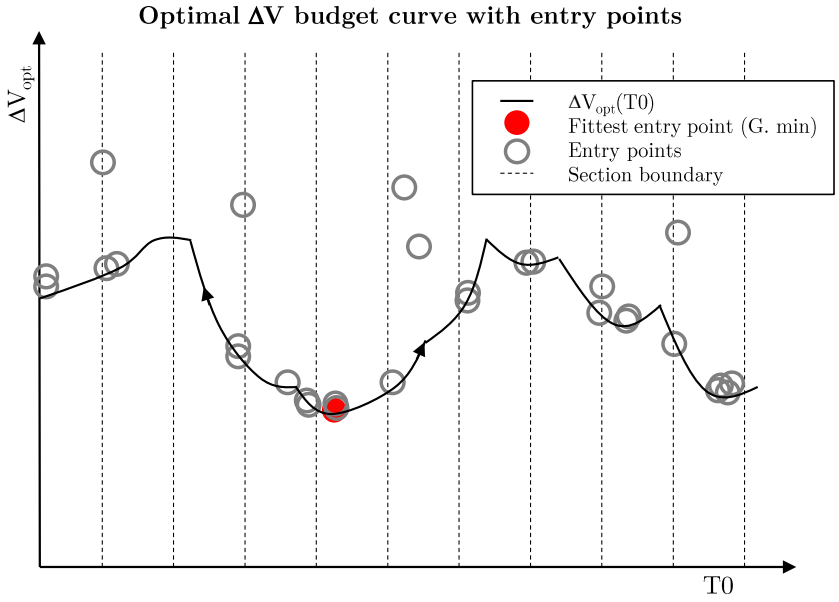


Figure 7.4: Schematic interpretation of an optimal ΔV budget curve $\Delta V_{opt}(T0)$. It can be seen that not all entry points converge to the $\Delta V_{opt}(T0)$ curve.

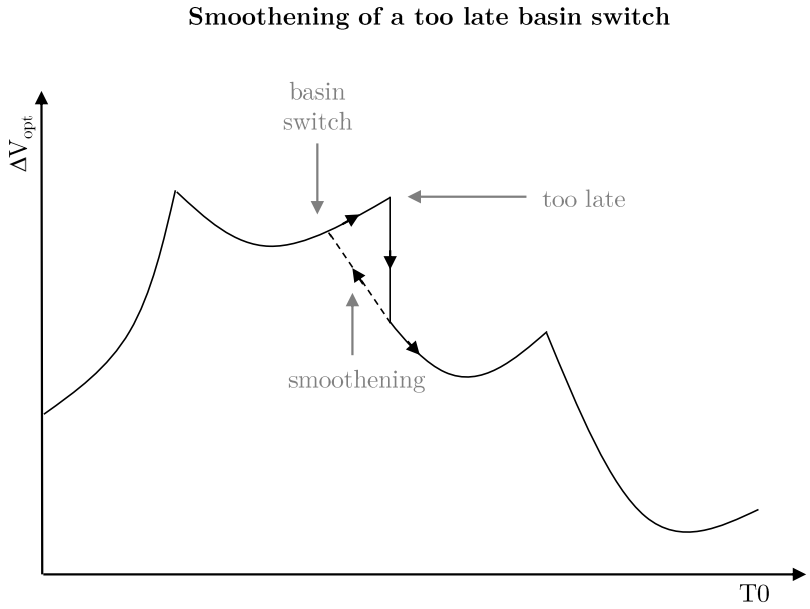


Figure 7.5: Example of smoothening. Propagating to the right results in a too late basin switch. By propagating to the left, the transition is smoothened.

Algorithm 7 Pseudo code of the a posteriori approach including initial guesses

```
discretize the range of departure epochs in N_dps values of T0
discretize the range of departure epochs in N_s sections
initialize matrix M of size N_dps × D
for each section do
    generate entry points
    round T0 of the solution to the nearest discretized value
    place the solution in the corresponding row of M
end for
start at the row after (forward in time) the fittest entry point
for current row to last row do
    while criteria for terminating retrying are not satisfied do
        initial guess is previous row from M
        optimize a posteriori objective function
        place solution in M
    end while
    go to next departure epoch
end for
repeat rows 10 to 17 the other way
re-optimize on spiky sections using different direction and/or propagation (human work)
find the minimum robust pair (Algorithm 1)
```

velocity budget) can be continuously monitored during test runs. This enables one to determine if performance is as expected. Lastly, plots such as Figure 7.3, but also Figure 8.2 and 8.3 in the next chapter, allow for confirmation of the functionality of the full algorithm.

Conclusion of the initial guess approach

The pseudo code of the a posteriori approach including the initial guess method can be found in Algorithm 7. A resolution of 0.1 day is combined with the previous solution extrapolation method for propagation, the linear extrapolation can be used to smoothen spiky sections of the optimal ΔV budget curve.

The proposed algorithm is the result of several comparisons and analyses, the chosen approach has been found to work satisfactory. However, it is likely that the performance can be further improved by assessing more extrapolation mechanisms, initialization methods and automated retry strategies on a wider sections than the twelve week window around the global minimum of Cassini1. The most comprehensive approach would be to compare the average number of function evaluations on various objective functions. This would be an elaborate study, but could yield substantial efficiency gains. It is recommended as a follow-up research.

7.1.2 Limited departure epoch range (pruning)

The second strategy applied to the a posteriori method limits the departure epoch range to $[-\Delta T0, \Delta T0]$ around the global minimum.[2] This means that the solution space for finding the entry point (global minimum) remains intact, while the solution space for the robust trajectory is pruned relative to the global minimum. The number of function evaluations used for identifying the global minimum (the entry point) is (decided to be) the value of performance indicator $N95$.

[2] For $\Delta T0 = 21$ days, the limited range is 4.2% of the full range.

This strategy only yields the minimum robust pair if it is near the global minimum, which is not guaranteed (illustrated in Figure 3.1). Still, it is interesting to research how much computational efficiency can be gained by this drastic pruning mechanism. Furthermore, the initial guess algorithm is applied to this method as well. This conflicts somewhat with the second principle formulated in the methodology of this chapter.[3] However, it is clear that not using the initial guess algorithm leads to exorbitant computation times, since this implies that a large series of $D-1$ dimensional optimization problems needs to be solved. Therefore, the limited departure epoch range algorithm also includes the propagation technique described in the previous section. A pseudo code tanscription of the limited departure epoch range method is provided in Algorithm 8. it has been verified in the same manner as described in Section 7.1.1.

Algorithm 8 Pseudo code of the limited departure epoch range method

```
initialize matrix M of size N_dps × D
minimize ΔV to find the entry point (the global minimum)
place the solution of the entry point in the middle row of M
discretize T0 in a range [−ΔT0, ΔT0] around the entry point
start at the row after the entry point (forward in time)
for current row to last row do
    while criteria for terminating retrying are not satisfied do
        initial guess is previous row from M
        optimize a posteriori objective function
        place solution in M
    end while
    go to next departure epoch
end for
repeat rows 4 to 8 the other way
find the minimum robust pair (Algorithm 1)
```

7.2 A priori approach

Three different algorithms that exploit similarity between trajectories, have been developed for the a priori approach. Section 7.2.1 argues for a specific initialization method and in Section 7.2.2, an additional operator is added to the differential evolution optimizer. Both leave the solution space intact, but bias the optimizer. The third method, which is proposed in Section 7.2.3, prunes the solution space.

7.2.1 Symmetric initialization (biasing)

The effect of the initialization method on the performance of differential evolution can be significant (Kazimipour, Li, and Qin 2014b). In this section, a tailored initialization strategy for the a priori approach, is proposed. It applies to the absolute objective function formulation.

The principle is simple: initialize decision vector $\mathbf{x_{ab}}$ symmetrically. That is, assign identical values to pairs of $T1$ to Tn, so $T1_1 = T1_2$, $T2_1 = T2_2$ et cetera. Taking the MGA decision vector

[3] The second principle is that concepts should be mutually exclusive.

as example, this yields the following structure for each initial member, where the magnitudes of the variables are still uniformly randomly generated.

$$\mathbf{x_{ab}} = \begin{bmatrix} T0_1 & T1_1 & T2_1 & \dots & Tn_1 & T1_1 & T2_1 & \dots & Tn_1 \end{bmatrix} \tag{7.4}$$

The rationale behind this method is the hypothesis that a population with members that start with the same control variable values for both trajectories, are also likely to find trajectories with similar control variable values.

In differential evolution, a mutant vector $\mathbf{m}$ is created for each member $\mathbf{u}$ by adding and subtracting random members of the population (see Section 6.1). Thus, a symmetrical population leads to symmetrical mutants $\mathbf{m}$. Then, crossover ratio CR determines the chance for individual entries of $\mathbf{m}$ to go into competing member $\mathbf{c}$; for each entry that is not crossed over from $\mathbf{m}$ to $\mathbf{c}$, the resultant $\mathbf{c}$ becomes less symmetrical. It can thus be concluded that the magnitude of CR plays a role in the rate of dilution of the symmetry in the population. Since CR plays a role in the effectiveness of symmetric initialization, it is likely that a different optimum value for it exists than in differential evolution without symmetric initialization. In this work, CR is left on the value determined in Chapter 6.

The form of symmetric initialization that is suggested above, is not possible for the relative objective function formulation. It would result in the initial values of the optimization variables of the second trajectories being all zeros. No addition or subtraction could lead to any of these parameters becoming non-zero. Research on alternative approaches that would avoids this characteristic, like Gaussian initialization, are recommended for future studies. Lastly, the functionality of the symmetrical initialization algorithm has been verified by inspecting initial populations that are generated by it. It has been verified that the populations are indeed symmetric.

7.2.2 Variable mirror algorithm (biasing)

A second manner to bias the optimization process towards solutions with similar values for both trajectories, is found in the new variable mirror operator. It is applied to the absolute objective function formulation. In this section, the working principle of the operator is explained first. After that, a design parameter named mirror threshold MT is tuned to the Cassini1 problem. It determines the threshold for executing the variable mirror operations.

Variable mirror algorithm description

By definition, fitness ΔV_{rb} of a robust trajectory pair is determined by the highest of the two ΔV's of the trajectories. This implies the classification method in the left graph of Figure 7.6. Only members that have a fit value for both trajectories, are classified as overall fit. For example, a very fit ΔV_1 combined with a very unfit ΔV_2 leads to a very unfit $\Delta V_{rb} = \Delta V_2$. One could criticize this method for wasting of potentially interesting (partial) solutions, since it also discards an entire member when just one the two trajectories is unfit.

The following remedy is proposed. It is suggested that for each member, the ratio between the two ΔV's is calculated. If the highest divided by the lowest exceeds a threshold MT, then the variables of the fit trajectory are copied to the entries of the unfit trajectory. This procedure is referred to as variable mirroring. The new classification is schematically displayed in the right graph of Figure 7.6. Blue areas indicate members that qualify for mirroring. The reader can view pseudo code of the mirror operator in Algorithm 9. It should be noted that application of the

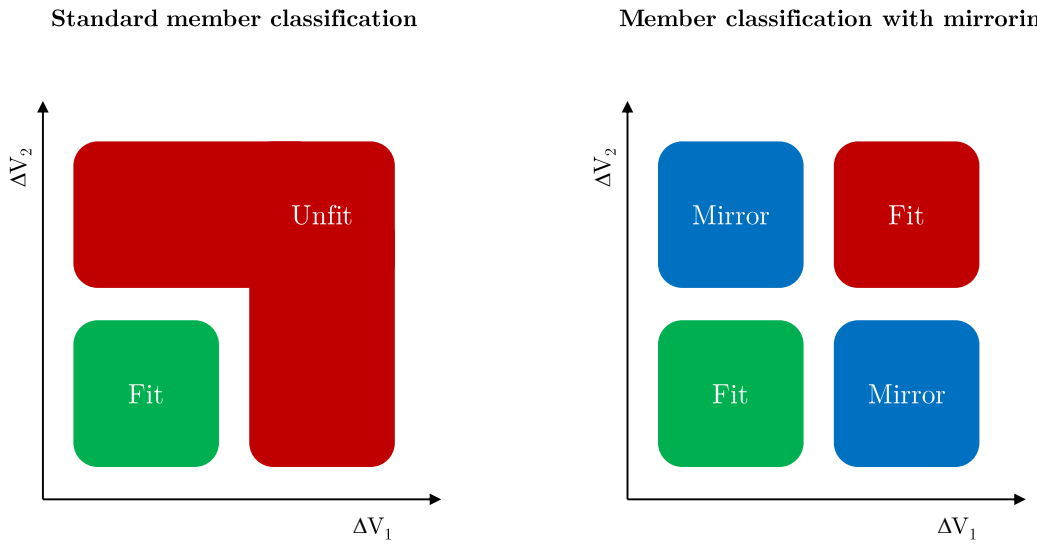


Figure 7.6: Classification of members with and without the variable mirror operator. The ΔV_1 and ΔV_2 axes correspond to the ΔV budgets of the respective trajectories that are represented by a single member, so every member corresponds to a location in the plot.

mirror operator does not depend on the fitness of the resultant (mirrored) member; there is no check after mirroring if the member as improved or not.

The variable mirror operator is applied in the iterative loop, before the standard differential evolution operations, as seen in Algorithm 10. The introduction of a new operator makes the resultant algorithm closely related to hybrid evolutionary algorithms. These techniques combine or alternate multiple (evolutionary) optimizers to achieve a better performance than the optimizers can achieve individually (Grosan and Abraham 2007). Often, it can only be confirmed experimentally if a specific combination of works or not. Hybrid evolutionary algorithms have been applied to spacecraft trajectories as well (remember COOP in Section 2.3.1 and SAGES in Section 2.4.2).

A disadvantage of the mirror operator is that it requires a design parameter that needs to be tuned, namely the mirror threshold MT. This is addressed in the following section.

Algorithm 9 Variable mirror operator

for each member in population **P do**
    get ΔV_1 and ΔV_2
    if $\Delta V_1/\Delta V_2$ exceeds mirror threshold MT **then**
        mirror variables: $\mathbf{x}_{rb}$ [2 to D] := $\mathbf{x_{rb}}[D+1$ to $D-1]$
    else if $\Delta V_2/\Delta V_1$ exceeds mirror threshold MT **then**
        mirror variables: $\mathbf{x}_{rb}$ [$D+1$ to $D-1$] := $\mathbf{x_{rb}}$[2 to D]
    end if
end for

Algorithm 10 Differential evolution including variable mirroring

1: **while** stopping criteria are not satisfied **do**
2: execute variable mirror function (Algorithm 9)
3: apply standard differential evolution for one loop
4: **end while**

Methodology mirror threshold optimization

To determine the optimal value of mirror threshold MT, twenty sets of 10^4 optimization runs have been performed for the following MT values.

$$\begin{array}{cccccccccc} [& 1.01 & 1.02 & 1.03 & 1.04 & 1.05 & 1.06 & 1.07 & \dots \\ \dots & 1.08 & 1.09 & 1.1 & 1.4 & 1.8 & 2.3 & 3 & \dots \\ \dots & 4 & 6 & 8 & 10 & 20 & \infty &] & \end{array} \tag{7.5}$$

One should note that $MT = \infty$ corresponds to standard differential evolution optimization, this value is included for verification. Of each set of 10^4 runs, the success rate p, average number of function evaluations $\overline{N_{fe}}$ and the number of mirror operations N_{mirror} are reported. This implies that the performance of the mirror algorithm is (partially) reported in this chapter already (the following chapter presents all results); this is inevitable due to the tuning of MT.

The tuning study has been done on the Cassini1 trajectory. This trajectory involves reasonable computation times, while it is still relatively difficult. Furthermore, a similar research is done on Cassini0 for verification (see Appendix D). It is not practically feasible to tune the mirror threshold to on the Cassini2 problem, because of the long computation times. In this respect, it must also be said that the variable mirror algorithm is only of value if it does not *need* to be tuned to the Cassini2 problem, or any individual trajectory. Tuning MT takes a lot more time than optimization itself.

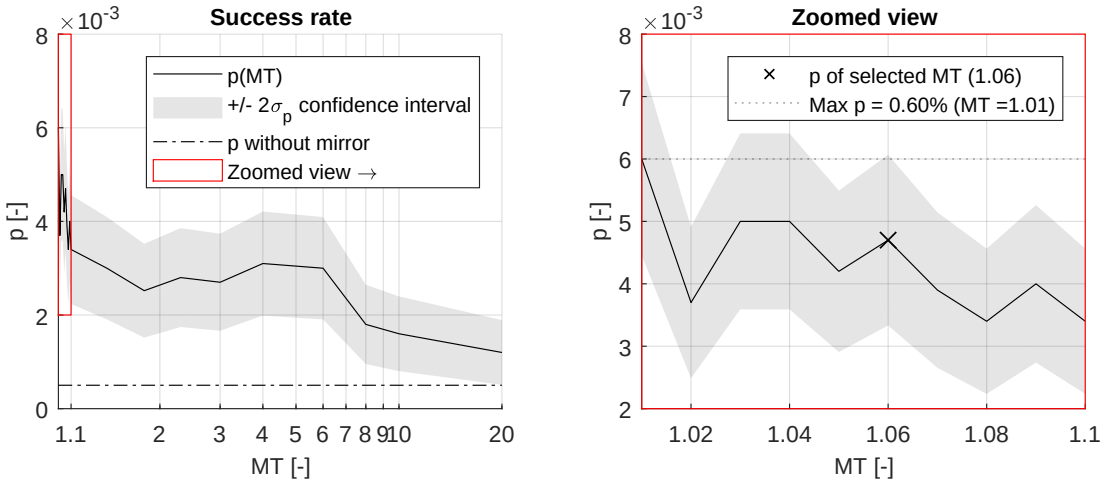


Figure 7.7: The success rate for various mirror threshold values on the Cassini1 trajectory.

Observations on the success rate

It can be seen in Figure 7.7 that the success rate is inversely correlated to the mirror threshold value. The graph is volatile (especially so for $MT = [1.01, 1.1]$), but this volatility remains within

the range of the +/-2σ_p confidence interval. In general, it is concluded that there is a significant improvement of the success rate for a decreasing mirror threshold. For comparison, the success rate of the standard a priori algorithm is included as a dashed line in the left graph.

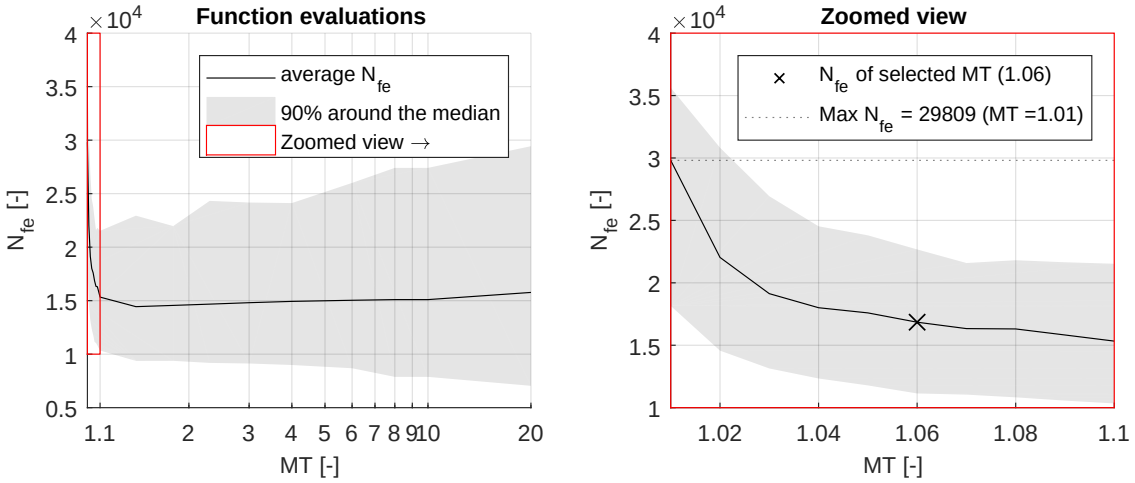


Figure 7.8: The average number of function evaluations for various mirror threshold values on the Cassini1 trajectory.

Observations on the number of function evaluations

Figure 7.8 shows that the average number of function evaluations $\overline{N_{fe}}$ is approximately constant for the MT range [1.1, 20], but increases soon after MT drops below 1.1 (middle two graphs). The right graph shows that the maximum $\overline{N_{fe}}$ is attained for $MT = 1.01$, $\overline{N_{fe}}$ is expected to increase further when the mirror threshold approaches 1. The increase of the average number of function evaluations with a decreasing mirror threshold comes from two sources.

1. Mirror operations directly lead to function evaluations. Namely, each member that has undergone a mirror operation needs its fitness to be re-evaluated.
2. Populations that that are subjected to mirror operations require more iterations to converge.

It has been found that both sources contribute about equal to the increase in function evaluations.

Observations on the performance indicator

Performance indicator $N95$ is shown in Figure 7.9. The inverse relation with the success rate is clearly recognizable and the volatility of the success rate is transferred to the performance indicator. The lowest performance indicator value is attained for $MT = 1.06$, namely $1.15 \cdot 10^7$. This is a very significant improvement compared to the $1.52 \cdot 10^8$ value of $N95$ of the standard a priori approach. It must be mentioned that the difference with $MT = 1.04$ is very small and all values are well within the +/-2σ confidence interval, making the choice for the precise value uncertain.

Observations on the mirror ratio

The mirror ratio divides the number of mirror operations N_{mirror} by the number of function evaluations N_{fe}.

$$MR = \frac{N_{mirror}}{N_{fe}} \tag{7.6}$$

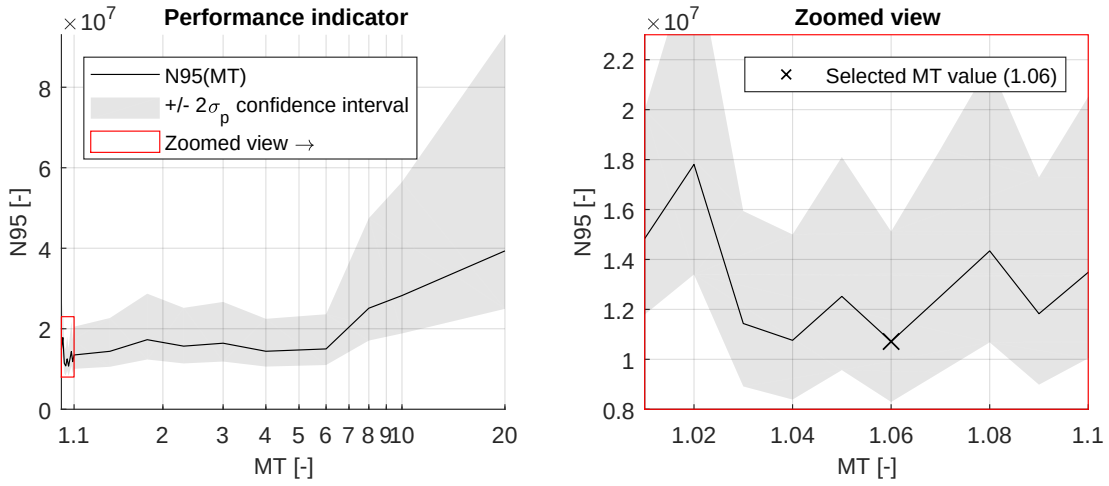


Figure 7.9: The performance indicator for various mirror threshold values on the Cassini1 trajectory.

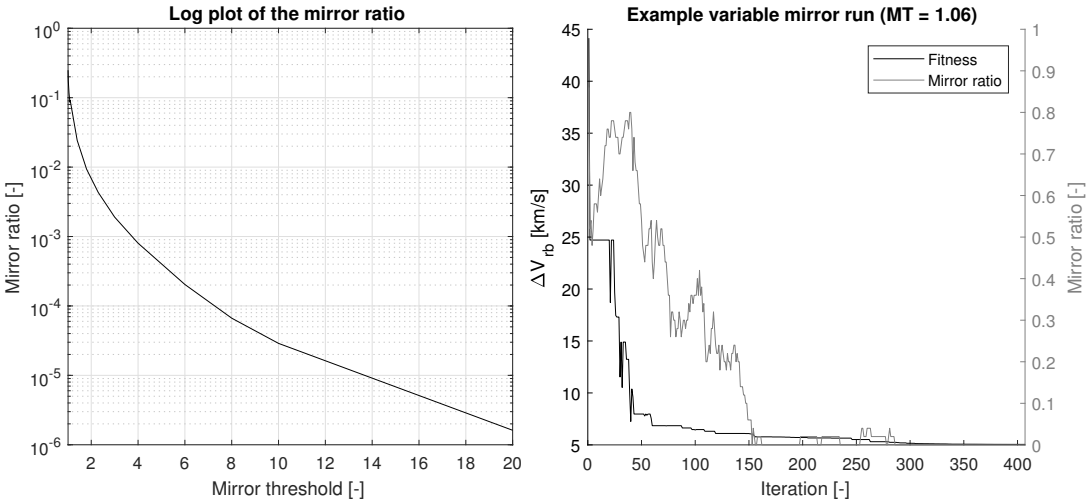


Figure 7.10: Mirror ratio as function of the mirror threshold (left) and the fitness and mirror ratio per iteration for a typical successful run on Cassini1 ($MT = 1.06$).

Figure 7.10 shows that the curve $MR(MT)$ is extremely steep for low MT values (note the logarithmic y-axis). There are very few mirror operations for the high $MT = 20$; the mirror ratio is in the order of magnitude of 10^{-6}. However, the success rate for this mirror threshold value is still significantly higher than for the a posteriori approach without the variable mirror operator. It is remarkable that very few mirror operations already have a significant impact on the results. A potential explanation is that mutant vectors are composed of five different population members. In Algorithm 4, it is stated that the following vector equation is used to generate mutants.

$$\mathbf{m} = \mathbf{b} + F \cdot (\mathbf{e} - \mathbf{f} + \mathbf{g} - \mathbf{h}) \tag{7.7}$$

The fact that each mutant is composed of five other members, may result in symmetrical members spreading quickly through the population. An interesting experiment to test this hypothesis would be to verify if the $best/1/exp$ strategy (see Section 6.1) performs worse; its mutants are composed of only three other vectors.

In the right graph of Figure 7.10, the development of both the fitness of a member and the mirror

ratio per iteration are displayed for a successful run with $MT = 1.06$. The maximum mirror ratio is approximately 0.8, so about 80% of the members underwent a mirror operation in that iteration. Furthermore, the fitness does not monotonically decrease, which it normally does in differential evolution. A decrease of the fitness of the population occurs when the best member undergoes a mirror operation that deteriorates its fitness. It has been analyzed what the effect is of discarding mirror operations that lead to worse fitness values. This meant that the vast majority of mirror operations was rejected and led to both a lower success rate and a lower number of function evaluations. These have opposite effects on the performance indicator, but the net effect of the lower success rate proves dominant. Therefore, only applying mirror operators when it leads to a better fitness, results in a worse overall performance.

Verification

The variable mirror operator has been implemented modularly. Its functionality has been confirmed by feeding it (small) populations of ten members and checking if the expected mirror operations have been executed. Furthermore, the results of the mirror threshold tuning study have been verified by doing a similar campaign on the (much simpler) Cassini0 problem. The results are included in Appendix D. An optimal mirror threshold of 1.05 has been observed, which is very close to the 1.06 on Cassini1.

Conclusion mirror threshold research

A mirror threshold MT of 1.06 leads to an $N95$ value that is an order of magnitude lower than without the mirror algorithm, on Cassini1. Other values in the range $[1.03, 1.1]$ produce similar results, volatility and uncertainty remain large. Verification on Cassini0 led to an MT of 1.05, while computation times prohibit a tuning study on Cassini2. The value of 1.06 is used in the remainder of this study.

7.2.3 Narrow relative boundary constraints (pruning)

The final method that is proposed in this chapter prunes the solution space of the relative objective function. This objective function has the convenient property that the control variables of the second trajectory can be constrained relative to the first. Therefore, a set of narrow boundary constraints is applied to the relative optimization variables. When these relative boundary constraints violate the absolute ones, the latter prevail.

The values of the narrow boundary constraints have been derived from the minimum robust trajectory pair, they are formulated moderately tightly around its coordinates. More conservative boundaries may be used in actual mission design, since one wants to avoid pruning the minimum robust trajectory pair. However, since this study is interested in the potential of this method, the values below are used; it provides an indication of the upper limit of the performance of the narrow relative constraints algorithm.

Cassini0 relative boundary constraints

Table 7.2 provides an overview of how the boundary constraints have been generated on Cassini0. The first five columns stem from the problem definition in Section 4.3. Column six and seven show the numerical values of the control variables of both trajectories of the minimum robust pair.[4] In

[4] These values have been obtained in the study on the optimal mirror threshold value of in Section 7.2.2.

Table 7.2: Overview of the values that are used for generating the narrow relative boundary constraints for the Cassini0 problem.

Column 1	2	3	4	5	6	7	8	9	10	11	12	13
Variable	**Units**	**Absolute boundaries**			**Solutions**				**Relative boundaries**			
		Lower	**Upper**	**Range**	**1**	**2**	**Difference**	**Normalized**	**Lower**	**Upper**	**Range**	**Ratio**
$T0$	MJD2000	-1000	0	1000	-583.81	-562.81	21.00	0.021	NA	NA	NA	NA
$T1$	days	400	2000	1600	736.42	1181.7	445.3	0.278	-500	500	1000	0.625
$T2$	days	1000	6000	5000	2133.0	3668.2	1535	0.307	-2000	2000	4000	0.80
Normalized volume of the relative solution space												0.5

Table 7.3: Overview of the values that are used for generating the narrow relative boundary constraints for the Cassini1 problem.

Column 1	2	3	4	5	6	7	8	9	10	11	12	13
Variable	**Units**	**Absolute boundaries**			**Solutions**				**Relative boundaries**			
		Lower	**Upper**	**Range**	**1**	**2**	**Difference**	**Normalized**	**Lower**	**Upper**	**Range**	**Ratio**
$T0$	MJD2000	-1000	0	1000	-800.81	-779.81	21.00	0.021	NA	NA	NA	NA
$T1$	days	30	400	370	167.63	149.87	-17.75	-0.048	-50	50	100	0.27
$T2$	days	100	470	370	449.39	449.39	$1.1 \cdot 10^{-4}$	$3.1 \cdot 10^{-7}$	-50	50	100	0.27
$T3$	days	30	400	370	55.84	53.70	-2.146	-0.006	-50	50	100	0.27
$T4$	days	400	2000	1600	1012.11	1034.18	22.07	0.014	-50	50	100	0.06
$T5$	days	1000	6000	5000	4532.96	4566.59	33.63	0.007	-50	50	100	0.02
Normalized volume of the relative solution space												$2.47 \cdot 10^{-5}$

column eight, the differences between the variables are provided: 21 days for T0 (as expected), 445 days for $T1$ and 1535 days for $T2$.[5] The relative boundary constraints are obtained by rounding these differences up with a resolution of 500 days, yielding 500 days for $T1$ and 2000 days for $T2$ (remember that $T0$ does not have a relative optimization variable). The lower and upper boundary constraints are equal in magnitude, the values are displayed in columns ten and eleven.

The rightmost column (Ratio) divides relative optimization variable ranges by the absolute ones. Taking the product of this column yields the ratios of the volumes of the relative solution space and the absolute solution space, 0.5 for this problem.

Cassini1 relative boundary constraints

The relative boundary constraints of Cassini1 are generated in a similar manner, as can be seen in Table 7.3. Here, the highest difference is rounded to 50 days, this value is uniformly adopted as both a lower and upper boundary constraint. This means that the solution space of the relative boundary constraints is a hypercube with side lengths of 100 days. The ratio between the volumes of the relative solution space and the absolute solution space is $2.468 \cdot 10^{-5}$. This is a significant reduction.

Cassini2 relative boundary constraints

The relative boundary constraints of the Cassini2 trajectory can be found in Table 7.4 and have been established in a slightly different fashion. The key difference is here that, because the control variables have various different units, it was found convenient to derive the relative boundary constraints from the normalized values. That is, the normalized values in column nine have been rounded up and multiplied by two (because the lower and upper boundaries are chosen to be equal), yielding the values in the rightmost column. Consequently, the ranges of the lower and upper boundaries are obtained by multiplying the values of column thirteen by the ranges in column five, yielding column twelve. Finally, the lower and upper boundary constraints are obtained by

[5] These differences are so large because trajectories of the robust pair each lie in different basins (this will be shown in Section 8.1.1).

centering the ranges around zero. The size of the relative solution space is in this case $5.033 \cdot 10^{-18}$, compared to a normalized absolute solution space of volume 1.

Hypercube side length comparison

An interesting analysis is to calculate the side lengths of a hypercube of the same volume as the normalized relative solution space, by taking the D_{re}-th root of the volumes, with D_{re} the dimensionality of the relative solution space. Realizing that $D_{re} = D - 1$ (since $\Delta T0$ is not a relative optimization variable), D_{re} is 2, 5 and 21 respectively for Cassini0 to Cassini2. This leads to the following hypercube side lengths for the respective trajectories.

$$\begin{aligned} &\text{Cassini0:} && 0.5^{1/2} && = 0.707 \\ &\text{Cassini1:} && \left(2.47 \cdot 10^{-5}\right)^{1/5} && = 0.120 \\ &\text{Cassini2:} && \left(5.03 \cdot 10^{-18}\right)^{1/21} && = 0.150 \end{aligned} \tag{7.8}$$

As expected, the side lengths are large for Cassini0. Furthermore, the side lengths on the Cassini1 problem (0.12) are slightly larger than those on the Cassini2 problem (0.15). Still, the relative solution space of Cassini2 is several orders of magnitude smaller due to its higher dimensionality. In absolute terms, $2.47 \cdot 10^{-5}$ and $5.03 \cdot 10^{-18}$ are both very small, one can conclude that the boundary constraints proposed in this chapter roughly half the volume of the global (11 and 43 dimensional) solution spaces for Cassini1 and Cassini2.

Table 7.4: Overview of the values that are used for generating the narrow relative boundary constraints for the Cassini2 problem.

Column 1	2	3	4	5	6	7	8	9	10	11	12	13
Variable	**Units**	**Absolute boundaries**			**Solutions**				**Relative boundaries**			
		Lower	**Upper**	**Range**	**1**	**2**	**Difference**	**Normalized**	**Lower**	**Upper**	**Range**	**Ratio**
$T0$	MJD2000	-1000	0	-1000	-791.45	-770.45	21.00	0.021	NA	NA	NA	NA
$\Delta V0$	km/s	3	5	-2	3.4945	3.2778	-0.22	-0.108	-0.4	0.4	0.8	0.40
δ_{LA}	radians	0	1	-1	0.5702	0.4923	-0.08	-0.078	-0.1	0.1	0.2	0.20
α_{LA}	radians	0	1	-1	0.4026	0.3653	-0.04	-0.037	-0.1	0.1	0.2	0.20
$T1$	days	100	400	-300	178.47	160.38	-18.09	-0.060	-30	30	60	0.20
$T2$	days	100	500	-400	425.28	422.43	-2.84	-0.007	-40	40	80	0.20
$T3$	days	30	300	-270	53.335	53.288	-0.05	$-1.7 \cdot 10^4$	-27	27	54	0.20
$T4$	days	400	1600	-1200	589.78	589.77	-0.01	$-7.1 \cdot 10^6$	-120	120	240	0.20
$T5$	days	800	2200	-1400	2200.0	2200.0	$-2.2 \cdot 10^4$	$-1.6 \cdot 10^7$	-140	140	280	0.20
$\eta1$	ratio	0.01	0.9	-0.89	0.7946	0.7421	-0.05	-0.059	-0.089	0.089	0.178	0.20
$\eta2$	ratio	0.01	0.9	-0.89	0.5496	0.5101	-0.04	-0.044	-0.089	0.089	0.178	0.20
$\eta3$	ratio	0.01	0.9	-0.89	0.1333	0.0105	-0.12	-0.138	-0.178	0.178	0.356	0.40
$\eta4$	ratio	0.01	0.9	-0.89	0.4578	0.0100	-0.45	-0.503	-0.534	0.534	1.068	1.20
$\eta5$	ratio	0.01	0.9	-0.89	0.0201	0.0100	-0.01	-0.011	-0.089	0.089	0.178	0.20
$R1$	ratio	1.05	6	-4.95	1.2072	1.5117	0.30	0.062	-0.495	0.495	0.99	0.20
$R2$	ratio	1.05	6	-4.95	1.0512	1.0500	$-1.2 \cdot 10^3$	$-2.4 \cdot 10^4$	-0.495	0.495	0.99	0.20
$R3$	ratio	1.15	6.5	-5.35	1.3060	1.3074	$1.4 \cdot 10^3$	$2.6 \cdot 10^4$	-0.535	0.535	1.07	0.20
$R4$	ratio	1.7	291	-289.3	69.819	69.808	-0.01	$-3.9 \cdot 10^5$	-28.93	28.93	57.860	0.20
$\gamma1$	radians	$-\pi$	π	2π	-1.5816	-1.6062	-0.02	-0.004	-0.01π	-0.01π	0.02π	0.02
$\gamma2$	radians	$-\pi$	π	2π	-1.9598	-1.9596	0.00	0.000	-0.01π	-0.01π	0.02π	0.02
$\gamma3$	radians	$-\pi$	π	2π	-1.5545	-1.5550	0.00	0.000	-0.01π	-0.01π	0.02π	0.02
$\gamma4$	radians	$-\pi$	π	2π	-1.5134	-1.5134	0.00	0.000	-0.01π	-0.01π	0.02π	0.02
Normalized volume of the relative solution space												$5.03 \cdot 10^{-18}$

Chapter 8

Results

In this chapter, the results attained by six different algorithms (shown in Figure 8.1) are presented and discussed. Besides the five pruning and biasing variants described in Chapter 7, a 'standard' a priori approach is included as well, to quantify the improvements of the pruning and biasing mechanisms. A variant of the a posteriori approach without the initial guess mechanism is omitted, because of its evident computational inefficiency. The results of the two a posteriori variants are first presented, followed by the four different a priori variants.

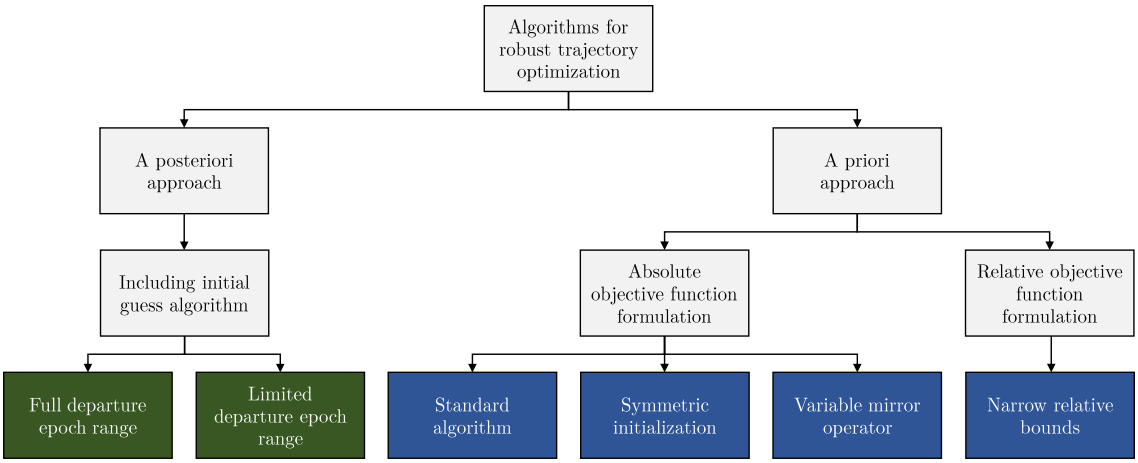


Figure 8.1: Overview of all tested algorithms.

8.1 Results of the a posteriori approach

The presentation of the results of the a posteriori approach follows a threefold structure. First, several graphs of the optimal ΔV budget curve $\Delta V_{opt}(T0)$ are presented in Section 8.1.1. Next, the performance is analyzed and finally the algorithm is reviewed.

8.1.1 Visualization of the optimal ΔV budget curve

Figure 8.2 and 8.3, printed on the next two pages, show the optimal ΔV budget curve for all three problems. These graphs form the core of the a posteriori approach, each is accompanied by several observations that have been made.

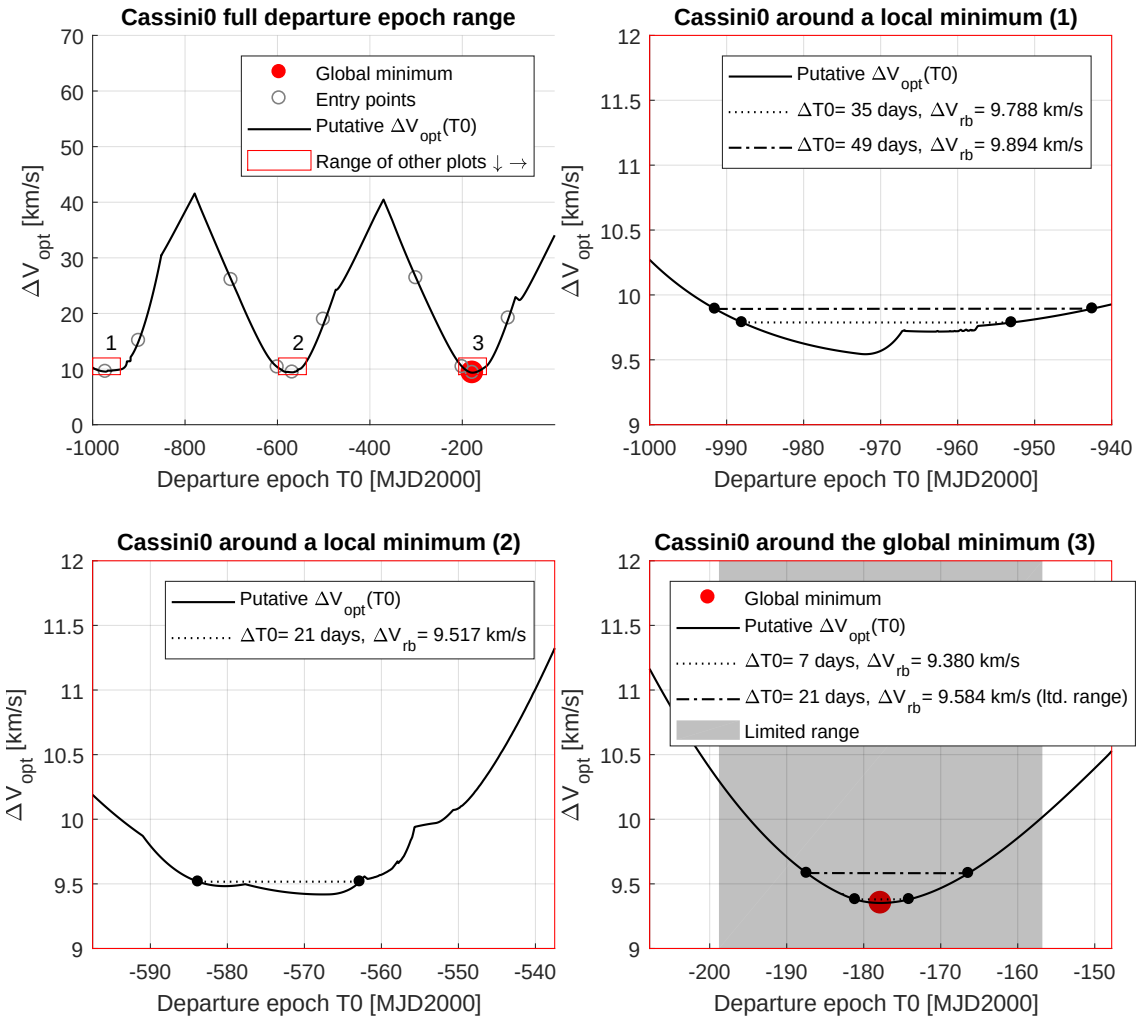


Figure 8.2: The putative optimal ΔV budget curve for Cassini0 and minimum robust trajectory pairs for several different different departure epoch intervals.

– The top left graph in Figure 8.2 shows that Cassini0 has three minima with very similar ΔV budgets. In the three close-up plots, it can be seen that depending on the value of the departure epoch interval, the minimum ΔV robust pair is at opposing sides of a different minimum. As a result, the limited range method has found a suboptimal robust pair near the global minimum ($\Delta V = 9.584$ km/s), with a ΔV budget that is 67 m/s higher than the global minimum robust pair (near the second local minimum, $\Delta V = 9.517$ km/s).

– Both trajectories of the $\Delta T0 = 21$ days minimum robust pair lie in different basins (note that the graph is not smooth between them). The differences between T1 and T2 are respectively 445 and 1535 days. In other words, one trajectory has a time of flight that is five years and five months longer than the other. It seems likely that in mission design, such differences are avoided, if possible. For comparison, the suboptimal robust pair near the global minimum has two trajectories that differ only one year and three months in time of flight, at the cost of a mere 67 m/s higher ΔV budget.

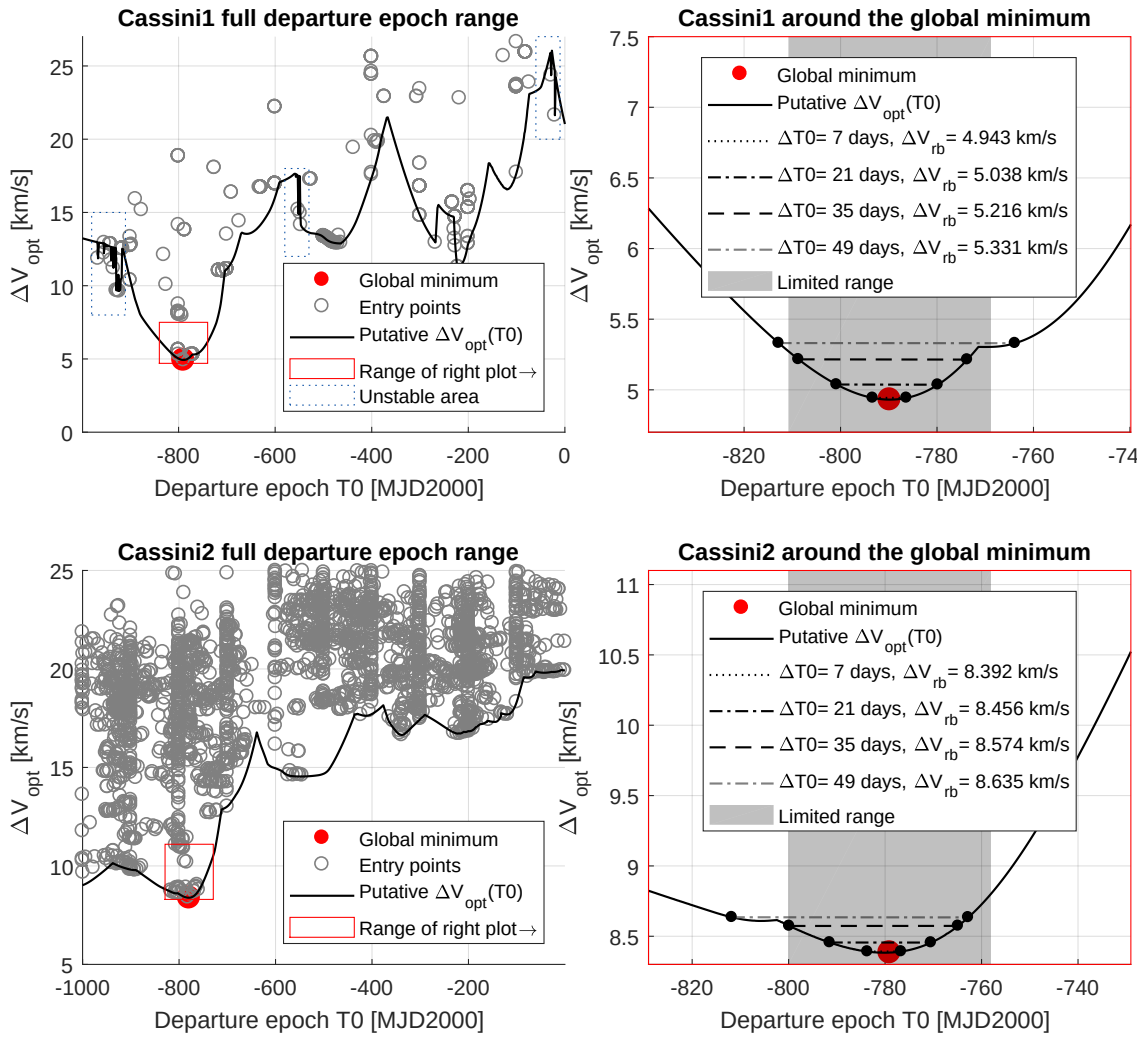


Figure 8.3: The putative optimal ΔV budget curve for Cassini1 and Cassini2 and minimum robust trajectory pairs for several different different departure epoch intervals.

- The departure epochs of the global minima of Cassini1 and Cassini2 in Figure 8.3 are similar. Away from this minimum, the resemblance between the graphs is much less outspoken; their shapes differ significantly, despite having the same flyby sequence.
- For both Cassini1 and Cassini2, the minimum robust trajectory pair is in the global minimum basin.
- During the generation of the optimal ΔV budget curve for Cassini1, there were three regions identified that did not allow a smooth line to be established, indicated as blue dotted rectangles in Figure 8.3. The trajectory function is unstable in these areas because the penalty function is very sensitive to changes in the control variables here. Since the MGADSM model lack a penalty function, no sections of similar (severe) instability have been encountered in the optimal ΔV budget curve of Cassini2, but it is still generally sensitive to changes in the control variables.

In the following pages, there will be periodically referred back to Figure 8.2 and 8.3.

8.1.2 Accuracy

Table 8.1 shows that the full range a posteriori method comes to approximately 0.5 m/s from the best attained solutions on each trajectory. The limited range method reaches the same values, with the exception of Cassini0, where a suboptimal solution (described above) led to a 67 m/s higher ΔV_{rb}.

The differences between ΔV_1 and ΔV_2 of the trajectories illustrate that the remaining errors are due to the limited resolution of $T0$ and the slope of the optimal ΔV budget curve. They prohibit the ΔV budgets to converge further. It can be concluded that the used resolution of 0.1 day leads to an accuracy in the order of magnitude of 0.5 m/s, which seems sufficient for a trajectory that is modeled using the linked-conics approximation.

Table 8.1: The accuracy of the a full range a posteriori method for $\Delta T0 = 21$ days. One should note that $\Delta V_{rb} = \max(\Delta V_1, \Delta V_2)$. Furthermore, ΔV_{best} is the ΔV budget of the best obtained solution in this research (including by the a priori method).

Problem	$\boldsymbol{\Delta V_1}$ km/s	$\boldsymbol{\Delta V_2}$ km/s	$\boldsymbol{\Delta V_{rb}}$ km/s	$\boldsymbol{\Delta V_{best}}$ km/s	**Error** km/s
Cassini0	9.51658	9.51719	9.51719	9.51671	0.00049
Cassini1	5.03751	5.03620	5.03751	5.03702	0.00049
Cassini2	8.45566	8.45527	8.45566	8.45513	0.00054

8.1.3 Computational efficiency

The numbers of function evaluations N_T of the full and limited range methods are provided in Table 8.2 and 8.3. In the following section, these numbers are discussed. Also, an analysis of the influence of the dimensionality on the accuracy of initial guesses is included.

The number of function evaluations of the full range method

Table 8.2 shows the specification of the number of function evaluations of the full range method. In the second last row, the total number of function evaluations N_T is provided. The last row calculates the ratio between N_T, and $N95$ on the trajectory function optimization problem (see Section 6.2.2). This figure compares the number of function evaluations that is required for finding a single trajectory ($N95$), and for finding the robust trajectory pair (N_T). There is a clear decreasing trend, from $4.49 \cdot 10^3$ to 2.86. This is because propagation and smoothening are relatively cheap on the more difficult problems, while $N95$ increases. The following two reasons have been identified.

1. The number of runs per departure epoch is of the same order of magnitude for each of the problems. Therefore, the number of a posteriori objective function optimization runs does not increase when problems become more difficult (see Table 5.1 for the objective function definitions).

2. The average number of function evaluations $\overline{N_{fe}}$ per run of the a posteriori objective function (for propagating and smoothening) is lower than $\overline{N_{fe}}$ per run of the trajectory function (for generating entry points). This reduction is greatest on Cassini2 with 83%, compared to 22% and 18% on Cassini0 and Cassini1. As a result, propagation and smoothening are cheaper on

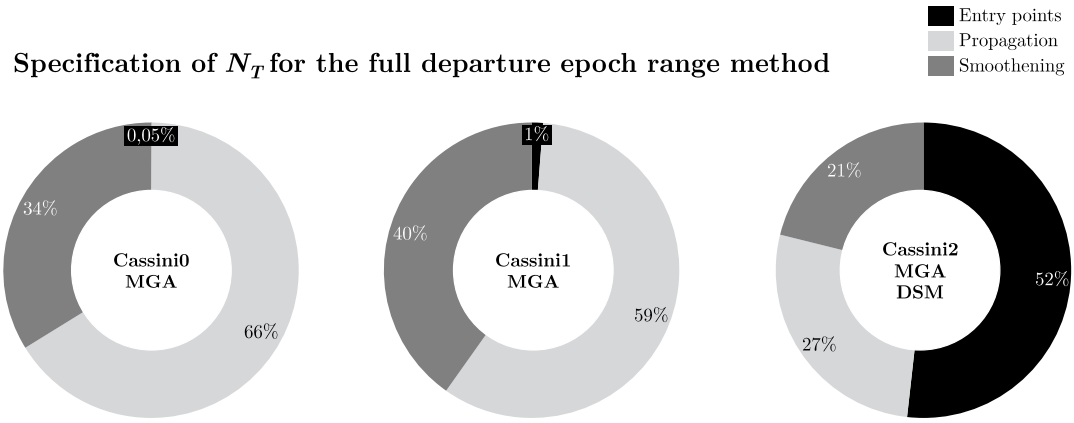


Figure 8.4: Relative contributions to the total number of function evaluations.

Table 8.2: Specification of the total number of function evaluations of the a posteriori full departure epoch range method. The last row shows the N_T values divided by N_{95} on the trajectory function. The subtotals do not precisely eqaul the products of the terms above them, since the numbers of function evaluations are distributed.

	Cassini0 MGA	**Cassini1 MGA**	**Cassini2 MGADSM**
Entry points			
Number of runs	$1 \cdot 10^1$	$3.2 \cdot 10^2$	$4.6 \cdot 10^3$
Number of function evaluations per run $\left(\overline{N_{fe}}\right)$	$1.14 \cdot 10^3$	$3.98 \cdot 10^3$	$1.21 \cdot 10^5$
Subtotal	$1.03 \cdot 10^4$	$1.27 \cdot 10^6$	$5.58 \cdot 10^8$
Propagation			
Number of departure epochs	$1.00 \cdot 10^4$	$1.00 \cdot 10^4$	$1.00 \cdot 10^4$
Number of function evaluations per run $\left(\overline{N_{fe}}\right)$	$8.88 \cdot 10^2$	$3.27 \cdot 10^3$	$2.12 \cdot 10^4$
Number of runs per departure epoch	1.52	1.60	1.32
Subtotal	$1.49 \cdot 10^7$	$6.01 \cdot 10^7$	$2.92 \cdot 10^8$
Smoothening			
Number of departure epochs	$1.00 \cdot 10^4$	$1.00 \cdot 10^4$	$1.00 \cdot 10^4$
Number of function evaluations per run $\left(\overline{N_{fe}}\right)$	$1.10 \cdot 10^3$	$3.65 \cdot 10^3$	$2.81 \cdot 10^4$
Number of runs per departure epoch	0.69	1.13	0.81
Subtotal	$7.60 \cdot 10^6$	$4.20 \cdot 10^7$	$2.41 \cdot 10^8$
Total (N_T)	$\mathbf{2.25 \cdot 10^7}$	$\mathbf{1.03 \cdot 10^8}$	$\mathbf{1.09 \cdot 10^9}$
$N_T/N95$	$449 \cdot 10^3$	$1.20 \cdot 10^2$	$2.86 \cdot 10^0$

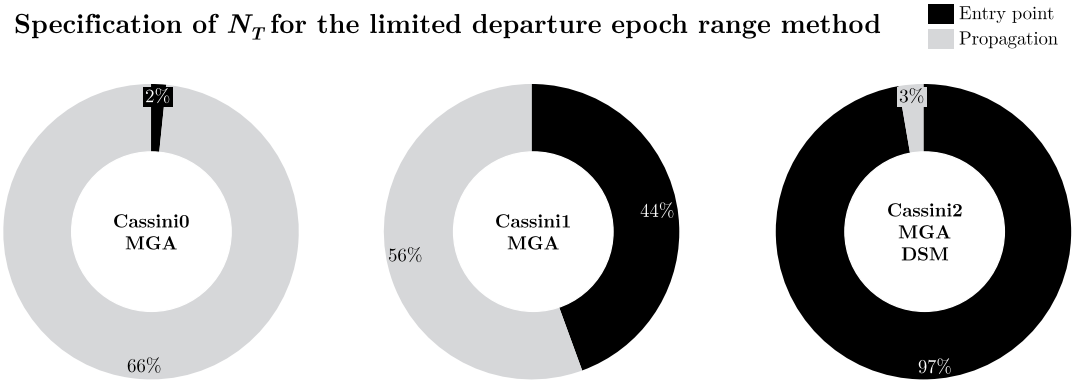


Figure 8.5: Relative contributions to the total number of function evaluations.

Table 8.3: Specification of the total number of function evaluations of the a posteriori limited departure epoch range method. The last row shows the N_T values divided by N_{95} on the trajectory function. The subtotals do not precisely eqaul the products of the terms above them, since the numbers of function evaluations are distributed.

	Cassini0 MGA	**Cassini1 MGA**	**Cassini2 MGADSM**
Entry point (global minimum)			
Number of function evaluations ($N95$ on the trajectory function)	$5.01 \cdot 10^3$	$8.64 \cdot 10^5$	$3.81 \cdot 10^8$
Propagation			
Number of departure epochs	$4.20 \cdot 10^2$	$4.20 \cdot 10^2$	$4.20 \cdot 10^2$
Number of function evaluations per run $\left(\overline{N_{fe}}\right)$	$6.90 \cdot 10^2$	$2.57 \cdot 10^3$	$2.52 \cdot 10^4$
Number of runs per departure epoch	1.03	1.00	1.00
Subtotal	$3.02 \cdot 10^5$	$1.08 \cdot 10^6$	$1.06 \cdot 10^7$
Total (N_T)	$\mathbf{3.07 \cdot 10^5}$	$\mathbf{1.94 \cdot 10^6}$	$\mathbf{3.92 \cdot 10^8}$
$N_T/N95$	61.3	2.25	1.03

Cassini2. An explanation for this greater reduction of $\overline{N_{fe}}$ on Cassini2 is proposed further below.

Both effects are illustrated in Figure 8.4, which shows that the relative contribution of propagation and smoothening becomes smaller from Cassini0 to Cassini2.

The number of function evaluations of the limited range method

The number of function evaluations of the limited range method is shown in Table 8.3, the relative contributions of the different routines are specified in Figure 8.5. The limited ranges are shaded gray in the zoomed-in plots of Figures 8.2 and 8.3.

Table 8.3 shows that relatively few function evaluations are required for this approach, compared to the full range method. The $N_T/N95$ ratio decreases from 61.3 to 2.25 to 1.03 from Cassini0 to Cassini2. It is remarkable that the limited range approach apparently requires only marginally

(3%) more function evaluations than trajectory function optimization on the Cassini2 problem. Table 8.3 also shows that the number of runs per departure epoch is different than on the full range. This is a result of the characteristics of the solution spaces around the global minima of the problems, it does not signify a performance difference. Furthermore, the average numbers of function evaluations are different from the full range method, but this difference stems from characteristics of the solution spaces near the minima as well, not from differences of the algorithms.

The limited range method has difficulty switching basins. This is due to its single entry point and it makes it unreliable when the minimum robust trajectory pair has trajectories in two different basins. In Figure 8.2 this is the case for the minimum robust pairs with $\Delta T0 \geq 21$ days and in Figure 8.3 for those with $\Delta T0 \geq 49$ days.

Relation between initial guess accuracy and dimensionality

The large decrease of $\overline{N_{fe}}$ when propagating the Cassini2 MGADSM trajectory (minus 83% compared to $\overline{N_{fe}}$ for the entry points) is remarkable. It is hypothesized that initial guesses become increasingly accurate when the dimensionality increases. The following explanation is proposed.

In this analysis, accuracy is defined as the volume of the part of the (normalized) solution space that is at least as close to the global minimum, as the initial guess. This volume is defined by a hypersphere around the global minimum with the initial guess at its surface. Furthermore, it is supposed that the initial guess has distance d to the global minimum, in each dimension. The analysis below calculates for which d it holds that the ratio between volumes V_D and V_{D-2} of two hyperspheres that have a dimensionality difference of two, is always smaller than one.

For a fixed distance d in every dimension, radius r of a hypersphere increases with dimensionality D, as shown below.

$$r = \sqrt{D} \cdot d \tag{8.1}$$

Furthermore, the following recursive formula holds for volume V_D, as function of volume V^*_{D-2} (de Costa Campos 2014). This last hypersphere has dimensionality $D-2$, and the same radius r_D as the D-dimensional hypersphere.

$$V_D = V^*_{D-2} \cdot \frac{2\pi r^2}{D} \tag{8.2}$$

However, we are interested in the relation with V_{D-2}, which has radius r_{D-2} instead of r_D. This leads to a simple equation of two hypersphere volumes of the same dimensionality $(D-2)$, but two different radii.

$$V^*_{D-2} = V_{D-2} \cdot \left(\frac{r_D}{r_{D-2}}\right)^{D-2} \tag{8.3}$$

Substituting Equation 8.3 in 8.2 yields Equation 8.4.

$$V_D = V_{D-2} \cdot \frac{2\pi r_D^2}{D} \cdot \left(\frac{r_D}{r_{D-2}}\right)^{D-2} \tag{8.4}$$

Equation 8.1 is then substituted in Equation 8.4.

$$V_D = V_{D-2} \cdot \frac{2\pi D d^2}{D} \cdot \left(\frac{\sqrt{D} \cdot d}{\sqrt{D-2} \cdot d}\right)^{D-2} \tag{8.5}$$

This can be simplified to the following expression. Note that D must be larger than 2.

$$V_D = V_{D-2} \cdot 2\pi d^2 \cdot \left(\frac{D}{D-2}\right)^{\frac{D}{2}-1} \tag{8.6}$$

We are interested in the situation in which $V_D < V_{D-2}$. This is true when the following inequality holds.

$$2\pi d^2 \cdot \left(\frac{D}{D-2}\right)^{\frac{D}{2}-1} < 1 \tag{8.7}$$

To find the d for which this inequality is guaranteed to hold, the following part of Equation 8.7 is investigated.

$$\left(\frac{D}{D-2}\right)^{\frac{D}{2}-1} \tag{8.8}$$

The derivative of this expression with respect to D is continuously positive for $D > 2$. Therefore, its limit for D to infinity yields its maximum value. Analysis reveals that this limit is e.

$$\lim_{D\to\infty} \left(\frac{D}{D-2}\right)^{\frac{D}{2}-1} = e \tag{8.9}$$

Now substituting Equation 8.9 in 8.7 leads to the value of d.

$$2\pi \cdot d^2 \cdot e < 1 \tag{8.10a}$$

$$d < \sqrt{\frac{1}{2\pi e}} \tag{8.10b}$$

$$\sqrt{\frac{1}{2\pi e}} \approx 0.2420 \tag{8.10c}$$

Therefore, it has been proved that when $d < \sqrt{1/(2\pi e)}$, the volume of hypersphere that contains all points that are at least as close to the global minimum as the initial guess, decreases when the dimensionality increases (for $D > 2$). It can be expected that $x < \sqrt{1/(2\pi e)}$ holds nearly always; it represents a situation in which the initial guess approaches the global minimum with an accuracy of 24% of the range of a variable.

8.1.4 Algorithm review

The algorithm review of both a posteriori variants is focused on complexity, the influence of design parameters, and versatility.

Complexity

The architecture of the a posteriori approach is relatively complex. Several subroutines are required for various tasks. These include discretizing the departure epoch, generating initial guesses and detecting solutions that should be retried. Furthermore, smoothening the optimal ΔV budget curve has not been fully automated because identifying potential areas for smoothening can be difficult to automate. A solution could be to smoothen the range of departure epochs from end to end, back and forth multiple times. However, this would be computationally inefficient.

Influence of design parameters

The a posteriori approach includes various design parameters that affect the computational efficiency of the algorithm. They are discussed below.

1. The number of steps N_{dps} into which the departure epoch is discretized has a large impact on the number of function evaluations. Indeed, for every departure epoch at least one optimization problem needs to be solved. On the other hand, smaller step sizes are likely to result in more accurate initial guesses, thus faster convergence and fewer runs per departure epoch. As discussed in Section 7.1.1, an optimal resolution is likely to exist, but it depends on the local characteristics of the solution space and is difficult to estimate. Therefore, N_{dps} is taken conservatively, resulting in a high number of function evaluations. Furthermore, it should be noted that for a fixed resolution on the full range approach, N_{dps} is proportional to the departure epoch range.

2. The limit on the numerical derivative, the limit on the numerical second derivative and the maximum number of runs per departure epoch affect the average number of function evaluations per departure epoch. Their impact is large in sensitive areas and is amplified by the fact that the number of function evaluations per run is higher in these areas.

Furthermore, the number of sections N_s in which entry points are generated determines the ability of the full range method to identify different basins. Also, the design parameters of differential evolution apply to the a posteriori (as well as the a priori) method, their impact has been discussed in Section 6.1.

Versatility

A very clear advantage of the a posteriori approach is the versatility of the result. The optimal ΔV budget curve can be used for any departure epoch interval. Also, a Pareto-optimal front of the minimum ΔV_{rb} and $\Delta T0$ can be generated using the results. Figure 8.2 and 8.3 illustrate the fact that $\Delta T0$ can be chosen freely. Were one interested in a robust pair with a $\Delta T0$ of 35 days on the Cassini1 and Cassini2 trajectories, it may be valuable to know that this can be expanded to 49 days at the cost of relatively little extra ΔV. Another advantage of the a posteriori approach is that it also shows the behavior of the optimal ΔV budget curve between the two departure epochs. Although there are no requirements on this behavior due to the definition formulated in Chapter 3, it may be of interest for specific applications. It can be concluded that the results are versatile and provide more information that required by the problem formulation.

Lastly, the a posteriori approach may be applicable to isomorphic problems beyond the field of interplanetary trajectories as well. This is likely to demand new studies on the optimal initial guess strategy and design parameters (such as the number of sections N_s of which the value depends on the typical width of basins), since these have been tuned to the characteristics of the MGA and MGADSM solution spaces.

8.1.5 Conclusion a posteriori results

The a posteriori approach is accurate to approximately 0.0005 km/s of the best attained solution. The full range method requires approximately $2.25 \cdot 10^7$, $1.03 \cdot 10^8$ and $1.09 \cdot 10^9$ function evaluations for Cassini0 to Cassini2, the limited range method requires respectively $3.07 \cdot 10^5$, $1.94 \cdot 10^6$ and $3.92 \cdot 10^8$ function evaluations. Furthermore, comparing the different methods and problems, the following conclusions are drawn.

1. The a posteriori method seems to scale well with the dimensionality of the trajectory function due to two reasons.
   (a) The number of runs per departure epoch does not increase with dimensionality.
   (b) Initial guesses become relatively more accurate for a higher dimensionality.
2. The limited range approach requires far fewer function evaluations than the full range method. On the difficult Cassini2 problem, this meant that finding a minimum robust trajectory pair requires only marginally (3%) more function evaluations than optimizing a single trajectory. However, on Cassini0, the global minimum robust trajectory pair has not been found by the limited range method.

Disadvantages of the a posteriori approach are its complexity and reliance on design parameters for efficiency. The main advantage is that the optimal ΔV budget curve can be used for any departure epoch interval $\Delta T0$, which is inherent to the a posteriori approach.

8.2 Results of the a priori method

In this section, the results of the four a priori methods are presented and discussed. First, the performance is assessed (accuracy and computational efficiency), followed by an algorithm review.

Table 8.4: Best attained values of each the a priori variants, compared to the overall best attained solution on each trajectory.

Problem		**Standard**		**Narrow**		**Symmetric**		**Mirror**	
	ΔV_{best}	ΔV_{rb}	**Error**	ΔV_{rb}	**Error**	ΔV_{rb}	**Error**	ΔV_{rb}	**Error**
	km/s	km/s	km/s	km/s	km/s	km/s	km/s	km/s	km/s
Cassini0	9.51671	9.51675	0.00004	9.51671	0.00000	9.51676	0.00005	9.51682	0.00012
Cassini1	5.03702	5.03817	0.00115	5.03735	0.00033	5.03702	0.00000	5.03719	0.00017
Cassini2	8.45513	9.86032	1.40520	8.45513	0.00000	8.70154	0.24641	8.63605	0.18093

8.2.1 Accuracy

Table 8.5 shows the lowest ΔV_{rb} values per method. On the Cassini0 and Cassini1 problems, all methods reach very similar values near the minimum robust trajectory pair. The best values on the Cassini2 MGADSM trajectory vary more. Only the narrow relative constraints algorithm is found capable of reaching a fit value of 8.455 km/s; Figure 8.3 shows that it is the minimum robust pair. The standard a priori method is not capable of reaching the threshold for success of 8.8 km/s by a wide margin.

8.2.2 Computational efficiency

The value of performance indicator $N95$ is provided in Table 8.5 and displayed in Figure 8.6 for each of the four a priori variants. Several observations on the reported performances are made.

General observations

A conclusion that can be drawn from Figure 8.6 is that the biasing and pruning methods have a beneficial effect on all problems, with the exception of the narrow relative constraints algorithm on Cassini0. Furthermore, as has been observed in Chapter 6, the standard approach does not converge to a solution below the VTR of 8.8 km/s on Cassini2, thus no performance is displayed

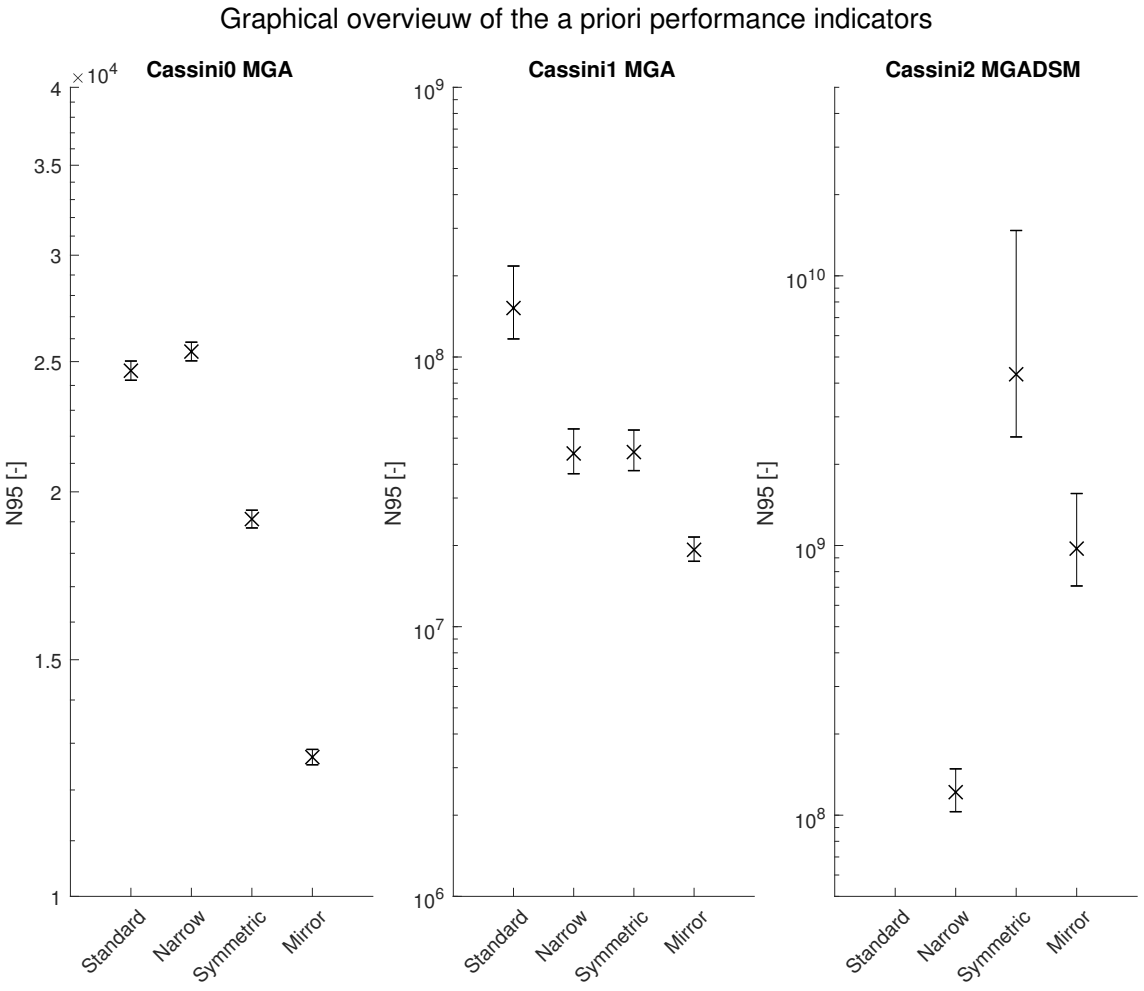


Figure 8.6: The performance indicator $N95$ of the a priori methods. The error bars correspond to the $N95$ values calculated using $p \pm \sigma_p$. The standard a priori method did not find a solution on the Cassini2 MGADSM trajectory.

in Figure 8.6.

The reader should note that the VTR of 9.6 km/s on Cassini0 allows the robust pair to be located in two different regions: either at opposing sides of the second local minimum or at opposing sides of the global minimum (see Figure 8.2). In the former case, both trajectories lie in different basins, while in the latter case both are in the same basin. Consequently, the biasing mechanisms (variable mirror and symmetric initialization) find more solutions near the global minimum, since the two trajectories are more similar there.

Comparison of pruning and biasing methods

Inspection of Figure 8.6 reveals that the magnitudes of the improvements induced by pruning and biasing vary between the methods as well as between the problems. Below, a series of observations and suggested explanations is stated.

1. The performance of the narrow relative constraints algorithm compared to the others, improves from Cassini0 to Cassini2. A likely explanation can be found in shrinking relative solution space; its volume decreases from 0.5 to $2.47 \cdot 10^{-5}$ to $5.03 \cdot 10^{-18}$, compared to a nor-

Table 8.5: Overview of the results of the priori methods. The VTR values are respectively 9.6 km/s, 5.1 km/s, and 8.8 km/s.

Cassini0 MGA

Quantity	**Symbol**	**Standard**	**Narrow**	**Symmetric**	**Mirror**
Sample size	N_s	10000	10000	10000	10000
Average fun. evals.	N_{fe}	3875	4337	3763	4134
Success rate	p	37.6%	40.0%	44.6%	62.3%
Standard deviation	σ_p	0.34%	0.35%	0.35%	0.48%
Performance indicator	$N95$	3.09E+04	3.25E+04	2.53E+04	1.99E+04

Cassini1 MGA

Quantity	**Symbol**	**Standard**	**Narrow**	**Symmetric**	**Mirror**
Sample size	N_s	20000	20000	20000	10000
Average fun. evals.	N_{fe}	27894	20252	25223	30323
Success rate	p	0.06%	0.14%	0.17%	0.47%
Standard deviation	σ_p	0.02%	0.03%	0.03%	0.07%
Performance indicator	$N95$	1.52E+08	4.33E+07	4.44E+07	1.93E+07

Cassini2 MGADSM

Quantity	**Symbol**	**Standard**	**Narrow**	**Symmetric**	**Mirror**
Sample size	N_s	10000	10000	10000	10000
Average fun. evals.	N_{fe}	684515	373698	564667	907268
Success rate	p	0.00%	0.92%	0.05%	0.28%
Standard deviation	σ_p	0.00%	0.10%	0.02%	0.05%
Performance indicator	$N95$	-	1.22E+08	3.42E+09	9.75E+08

malized absolute solution space of the respective problems. The fact that the narrow relative constraints algorithm performs worse than the standard a priori approach on Cassini0, the total volume of the total solution space has decreased, suggests that the relative objective function is a less effective formulation than the absolute objective function. This seems to be compensated on Cassini1 and Cassini2 by the sharp reduction of the solution space volume.

2. The variable mirror method works specifically well on the Cassini0 and Cassini1 trajectories. This may be related to the fact that MT has been optimized on the Cassini1 trajectory, the found optimal value of 1.06 has been verified to be (near) optimal for Cassini0 as well (1.05 is optimal for Cassini0). No MT optimization study has been done on Cassini2 due to the intractable computation times, thus there may exist a more optimal MT for this trajectory. If this would be confirmed, it would imply a defect of the variable mirror algorithm; having to tune the algorithm undermines its usability.

3. Symmetric initialization has a moderately positive effect on all three problems. A advantage of this method is that it seems best capable of dealing with a situation in which the minimum robust trajectory pair is dissimilar. Its initialization method gives it a bias, but the optimization loop is unperturbed.

It can be concluded that several different mechanisms contribute to the effectiveness of the pruning and biasing methods.

Differences between the average number of function evaluations

It has been observed that the average numbers of function evaluations are affected by each of the pruning and biasing methods. The following explanations for differences with respect to the standard methods are proposed.

1. The variable mirror operator increases the number of function evaluations, which has already been described in Section 7.2.2. The increase can be attributed to extra function evaluations that determine the fitness of mirrored members, and to postponed convergence. Despite the higher number of function evaluations, a better $N95$ (compared to the standard approach) is attained because of the higher success rates.

2. It seems that the number of function evaluations of the narrow relative constraints algorithm is correlated with the reduction of the solution space. Indeed, its $\overline{N_{fe}}$ compared to the that of the standard a priori approach reduces from Cassini0 to Cassini2, while the relative solution space volume also decreases (see Table 8.5).

3. The symmetric initialization method leads also to a reduction of the number of function evaluations. This seems to be a result of the fact that a symmetric initial population is likely to find a (near) symmetric solution earlier.

These remarks conclude the analysis of the performances of the a priori methods.

8.2.3 Algorithm review

Below, the characteristics of the a priori approach are determined by assessing the complexity, influence of design parameters, and versatility.

Algorithm complexity

The a priori methods are conceptually simple. Indeed, all methods consist of optimization runs using regular differential evolution, except for the variable mirror method which includes an additional operator. However, this operator is of limited complexity and it is modularly implemented in differential evolution.

Influence of design parameters

The standard and symmetric initialization a priori methods have the advantage that they do not require any extra design parameters besides the settings of differential evolution (all variants require these). However, the two methods that show the best performances on the problems, variable mirroring and the narrow relative constraints algorithm, do require values to be set by the user.

It has been demonstrated that the value of MT has a significant impact on the performance of the variable mirror algorithm. The value has been determined by optimizing its performance on Cassini1. However, were this algorithm to be applied on very different objective functions, it may be found that MT is not the ideal metric. It relates two ΔV values directly to each other, thus the ideal value of MT depends on the shape of the solution space. Furthermore, it is likely that the magnitude of MT depends on departure epoch interval $\Delta T0$, which as remained fixed on 21 days in this study. An interesting further development would be including a feedback loop that makes MT self-adjusting, for example dependent on mirror ratio MR. This hinges on the hypothesis that an MR value exists that is (near) optimal in the general case. This hypothesis remains to be confirmed.

For the narrow relative boundaries algorithm to work well, the values of these boundaries need to be set.[1] On the one hand, too narrow values will prohibit the minimum robust trajectory pair to be found. On the other hand, to wide boundaries lead to an unnecessarily high number of function evaluations. Another note is that it may actually be desired to prune minimum robust trajectory pairs that are dissimilar. A good example is the minimum robust pair of Cassini0; the two trajectories have times of flight that differ by more than five years. It is possible that such solutions are tried to be avoided; the narrow relative constraints method enables one to prune these directly.

Versatility of the results and methods

Since the a priori method requires $\Delta T0$ to be set beforehand, the results only apply to the $\Delta T0$ value in question; the general behavior of the optimal ΔV budget curve cannot be derived from the results. This property is a direct result of the definitions that have been formulated in Chapter 3.

The general formulations of the symmetric and narrow relative constraints algorithms makes one suspect that they can be easily applied to problems outside the field of spacecraft trajectory optimization as well. The variable mirror algorithm's MT parameter makes it less generally applicable.

8.2.4 Conclusion a priori results

On the Cassini0 and Cassini1 trajectories, all four variants reached an accuracy of 1 m/s or less. The variable mirror method is most computationally efficient on these problems, with respectively

[1] It can be argued that these are strictly not design parameters since they do not affect the optimization mechanism, but they are addressed in this section because they also need to be set by the user and have a major impact on performance.

two and ten times fewer function evaluations than the standard a priori approach. On the Cassini2 problem, the narrow relative constraints algorithm is the only variant that reaches high accuracy, it yields the best solution attained in this research (8.455 km/s). Its number of function evaluations is an order of magnitude lower than that of the variable mirror algorithm (second best performance). In general, it seems that the high dimensionality of this problem (43) has a large negative effect on the performance of the a priori approach, through low success rates.

The standard and symmetric forms do not require design parameters, but the narrow relative constraints and variable mirror algorithms do require them. This is an unfavorable characteristic because they need to be tuned. Advantages of the a priori method are its simplicity and potential general applicability, but inherent to the approach, each $\Delta T0$ requires a new optimization campaign.

Chapter 9

Conclusion

Based on the results that have been presented in the previous chapter, a final comparison between the a posteriori and a priori approach is made. This includes a conclusion on the hypothesis that an a priori approach requires fewer function evaluations than an a posteriori approach Also, a wider conclusion on the relative performances and the effects of several problem characteristics is included.

9.1 Comparison between both methods

The comparison of this section is split in a quantitative analysis of the performance and a qualitative algorithm review.

9.1.1 Performance

The quantitative performance is arguably the most important aspect of any optimization method; it is often the main criterion in optimizer selection. In the following section, accuracy and computational efficiency are addressed respectively.

Accuracy

The accuracy of the a priori approach is limited by the stopping criteria of differential evolution only, the spread in best values obtained by the different variants is 0.1 m/s on Cassini0 and 1 m/s on Cassini1. On the Cassini2 only the narrow relative constraints algorithm reached the minimum robust trajectory pair accurately. The a posteriori approach has an additional limitation to its accuracy; discretization of the departure epoch (the step size is 0.1 day) limits the fitness on all problems at 0.5 m/s from the best solutions of the a priori approach. Errors due to the linked-conics approximation are likely to be larger.

Computational efficiency

A comparison between the computational efficiency of the various methods is presented in Figure 9.1. The a priori approach performs significantly better on Cassini0, while on the other problems the a posteriori limited range method shows the best performance. It must be remarked that the latter prunes drastically, leading to a suboptimal pair on Cassini0. Also, the rightmost figure includes two limited range N_T values; one for $VTR = 8.5$ km/s (left) and one for $VTR = 8.8$ km/s (right). The latter allows for the best comparison with the a priori method, which also has a VTR

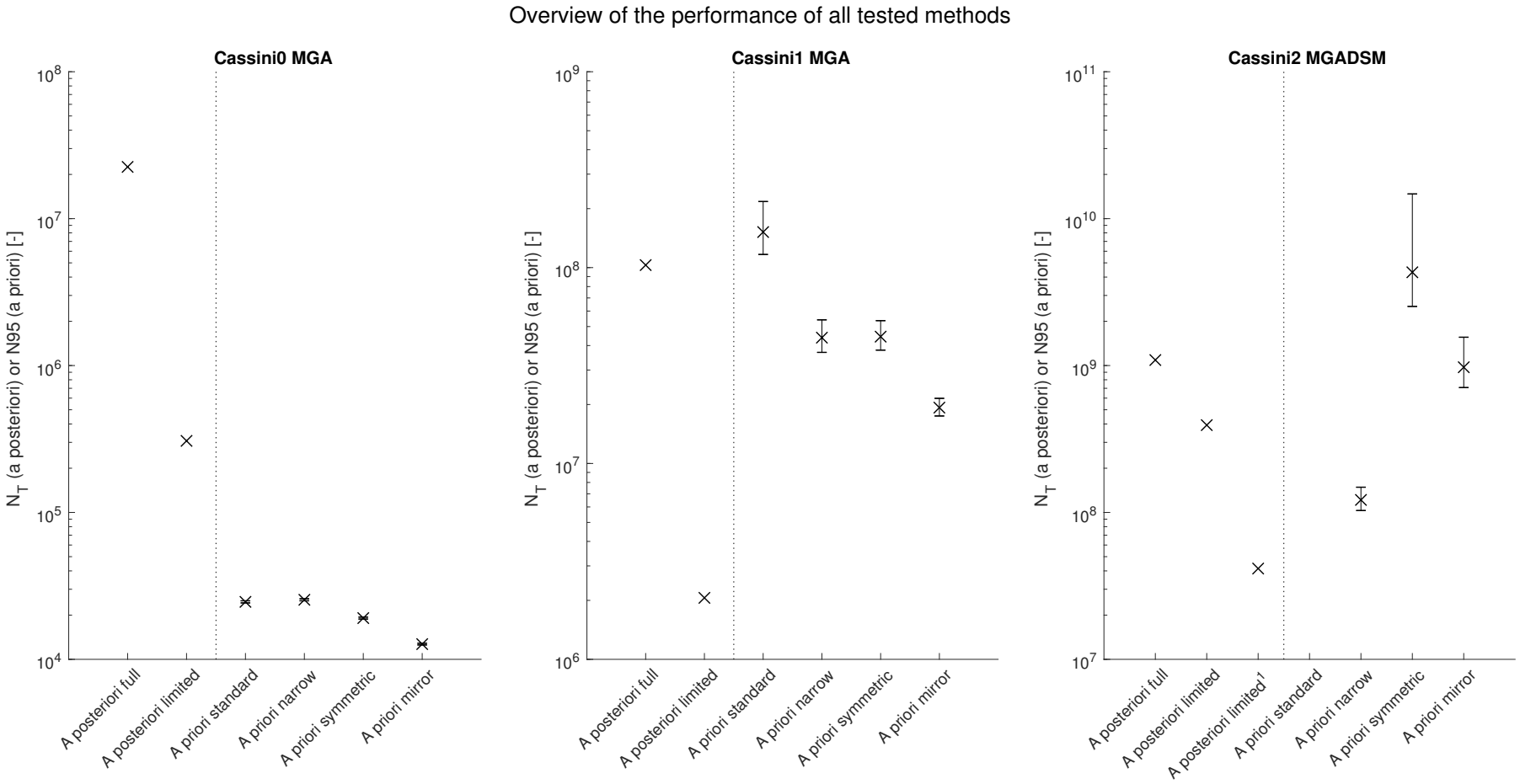


Figure 9.1: Comparison of the $N95$ parameter of all methods. The error bars correspond to $N95$ computed using $p \pm \sigma_p$. [1]) Calculated using a $N95$ for a VTR of 8.8 km/s, instead of 8.5 km/s.

of 8.8 km/s. On the full range, relaxing the value-to-reach does not lead to a significantly lower N_T. Between the different variants of the a priori approach, the most notable performances are attained by variable mirroring (biasing) and the narrow relative constraints algorithm (pruning).

The reader should notice that the performances of the a posteriori and a priori methods are measured in the total number of function evaluations N_T and performance indicator $N95$ respectively. Because differential evolution does not always converge to the minimum robust trajectory pair, the computational efficiency of both methods has to be measured slightly differently. Still, Figure 9.1 provides a good indication of the relative performances of both approaches.

Scaling of computational efficiency with trajectory dimensionality

The trajectory optimization problems that are considered are of dimension $D = 3$, $D = 6$ and $D = 22$. This wide range makes it possible to relate the performances of the methods to the dimensionalities of the trajectory problems. The following conclusions have been drawn.

1. The computational efficiency of the a posteriori approach scales relatively well with an increasing dimensionality, because of the following two reasons.
   (a) The success rate of propagating and smoothening does not decrease when the dimensionality increases.
   (b) The accuracies of the initial guesses of the a posteriori method become relatively better, when the dimensionality increases.
2. The performance of the a priori approach deteriorates significantly with an increasing dimensionality. The number of optimization variables nearly doubles, compared to the trajectory function. This leads to lower success rates, and a higher numbers of function evaluations per run.

It is concluded that the computational efficiency of the a posteriori approach scales better with trajectory dimensionality D, than the computational efficiency of the a priori approach. This is supported by the results presented in Figure 9.1.

Scaling of computational efficiency with departure epoch range

A difference between the full range a posteriori method, and the a priori methods is that the number of function evaluations of the former increases approximately proportionally with the range of $T0$ (for a fixed step size of $T0$), while the latter does not (a property of evolutionary algorithms). Therefore, the relative performance of the a priori approach is likely to improve for larger ranges of $T0$, compared to the full range a posteriori method.

The influence of the departure epoch interval

In this is study, departure epoch interval $\Delta T0$ has remained fixed at 21 days. Analyzing the effect of the value of this parameter, the following can be stated. Regarding the full range a posteriori approach, N_T is independent of $\Delta T0$. On the limited range method, the number of function evaluations required for propagating and smoothening scales approximately linear with $\Delta T0$. For the a priori approach, the pruning and biasing mechanisms are likely to become less effective for larger $\Delta T0$ values, since similarity decreases with a larger $\Delta T0$.

9.1.2 Algorithm review

Based on reviews of the developed algorithms, conclusions are drawn on the relative complexity, the influence of design parameters and versatility of both the methods and results.

Complexity

The a posteriori approach requires several mechanisms to work efficiently, making the algorithm complex. On the other hand, the a priori methods have a much simpler architecture. They are basically regular differential evolution optimization efforts, with the exception of the variable mirroring scheme which adds an additional operator. In general, it is concluded that the a posteriori method involves a more complex algorithm than the a priori method.

Influence of design parameters

The aforementioned mechanisms in the a posteriori approach require several parameter values to be set by the user. The largest influence is exerted by the number N_{dps} of departure epochs in which $T0$ is discretized. The fact that this parameter needs to be set as a first step, without any knowledge of the solution space, is a considerable challenge.

In case of the a priori method, only the narrow relative constraints and mirror methods are influenced by user-set values. For setting the narrow relative constraints, one can rely to a certain extent on intuition, but one always risks pruning optimal pairs. For setting the non-physical quantity of the mirror threshold MT, one cannot rely on intuition. The performance of the (near) optimal MT of Cassini0 and Cassin1, is not as good on Cassini2. It suggests that MT could vary per problem, which would undermine the applicability of the algorithm. Furthermore, it is also likely that MT depends on the magnitude of the departure epoch interval.

It can be concluded that the a posteriori method and two best performing a priori methods (variable mirroring and narrow relative constraints) are both sensitive to design parameters.

Versatility of the results and methods

The results produced by the a posteriori method are more versatile than those of the a priori methods; the optimal ΔV budget curve can be used for any departure epoch interval. Contrary to this, the a priori method only yields information for a single departure epoch interval per optimization effort. These properties are inherent to both approaches, it depends on the demands of the mission designer what the merit of the versatility of the results of the a posteriori approach is.

Looking at the versatility of the methodology, it can be argued that the a priori algorithms are more versatile. They may be readily applied to similar problems that have a '$T0$-dimension' with a very different shape (of course, this would be a different quantity), with the exception of the variable mirror algorithm. This last variant, as well as the initial guess algorithm, would have to be tuned to the characteristics of the solution space.

9.1.3 Final word on the comparison

In this study, it has been found that the pruning and biasing techniques are decisive for the relative performances of both the a posteriori and the a priori approach. The a posteriori approach involves many consecutive optimization problems, but mitigates the computational effort with a powerful initial guess generator. The formulation of the a priori approach allows all control variables to be

optimized at once, thereby avoiding the sub optimization problems. The price for this is an almost double number of optimization variables. This doubling is an issue that becomes increasingly hampering for higher dimensional solution spaces, while the initial guesses appear to become increasingly effective. It is therefore concluded that the dimensionality is a key determinant of the relative performances of the methods. The hypothesis that the a priori approach requires fewer function evaluations than the a posteriori approach is confirmed on the simple Cassini0 problem, but refuted on the more difficult Cassini1 and Cassini2 problems.

9.2 Recommendations for further research

Several fields of improvement have been identified. Below, a list of four suggestions for future studies is included.

1. The developed algorithms can be incrementally improved. Severals suggestions for such improvements have been made in the text of this report. Examples are a self-adapting mirror threshold and a comparison between different initial guess initialization methods. Furthermore, a study on the optimal stopping criteria of the differential evolution optimizer is recommended. This is likely to increase the success rates on the Cassini2 MGADSM problem.

2. A hybrid a posteriori method that evaluates fewer departure epochs than the full range method, but still scans the entire solution space, is proposed. It generates entry points analogously to the full range method and then propagates outward from the fittest entry point, until a robust trajectory pair can be evaluated. This repeats itself for the second fittest entry point, and so on, until it encounters an entry point that is less fit than the fittest robust pair. Increasing the departure epoch step size within basins is likely to further enhance the performance. Because this approach generates entry points similarly to the full rang method, but propagates only a limited section, it is named hybrid.

3. Applying the a posteriori and a priori methods to the multiple-shooting model (Section 2.2) is likely to increase the computational efficiency. This would also involve replacing differential evolution with monotonic basin hopping, which is expected to further enhance the performance on the higher dimensional problems.

4. To avoid costly objective function evaluations, the trajectory models may be supported by a surrogate model (for example established using an artificial neural network). Evaluating the surrogate model is likely to be less expensive than the trajectory functions (Ampatzis and Izzo 2009; Izzo, Sprague, and Tailor 2018).

Over the course of this thesis, some clarity has been produced on the relative performance of the a posteriori and a posteriori approaches to finding pairs of spacecraft trajectories of which the highest ΔV budget is minimized. The list above makes clear that, as seems to be inherent to research in an interesting field, answering the research questions has made various new ones arise.

Bibliography

Abilleira, F. et al. (2014). "Mars InSight mission design and navigation". In: *24th AAS/AIAA Space Flight Mechanics Meeting*. URL: https://trs.jpl.nasa.gov/bitstream/handle/2014/.../14-0444_A1b.pdf.

Ampatzis, C. and D. Izzo (2009). "Machine learning techniques for approximation of objective functions in trajectory optimisation". In: *Proceedings of the IJCAI-09 Workshop on Artificial Intelligence in Space*, pp. 1–6. URL: https://www.esa.int/gsp/ACT/ai/projects/ijcai09/papers/s5_2ampat.pdf.

Bate, R. R., D. D. Mueller, and J. E. White (1971). *Fundamentals of astrodynamics*. Courier Corporation. URL: http://books.google.com/books?vid=ISBN0486600610.

Bedini, P. D. (2017). "Messenger mission overview". In: *Space Science Reviews* 34.1, pp. 5–13. DOI: 10.1007/s11214-007-9247-6.

Biesbroek, R. (2016). *Lunar and interplanetary trajectories*. Springer International Publishing. DOI: 10.1007/978-3-319-26983-2.

Brennan, M. (2015). "Preliminary interplanetary trajectory design tools using ballistic and powered gravity assists". PhD thesis. The University of Texas at Austin. URL: https://repositories.lib.utexas.edu/handle/2152/31342.

Byrd, H, J. C. Gilbert, and J Nocedal (2000). "A trust region method based on interior point techniques for nonlinear programming". In: *Mathematical Programming* 89.1, pp. 149–185. DOI: 10.1007/PL00011391.

Ceriotti, M. (2010). "Global optimisation of multiple gravity assist trajectories". PhD thesis. University of Glasgow. URL: http://theses.gla.ac.uk/2003/.

Coates, A. (2017). "Cassini – Huygens : Saturn, rings and moons". In: *Astronomy and Geophysics* 58.4, pp. 420–425. DOI: 10.1093/astrogeo/atx137.

Conway, B. (2010). "The problem of spacecraft trajectory optimization". In: *Spacecraft trajectory optimization*. Ed. by B. Conway. Cambridge University Press. Chap. 1, pp. 1–15. DOI: 10.1017/CBO9780511778025.

Curtis, H. D. (2013). *Orbital mechanics for engineering students*. Butterworth-Heinemann. DOI: 10.1016/B978-0-08-097747-8.01001-X.

de Costa Campos, L. M. B. (2014). *Generalized calculus with applications to matter and forces*. CRC Press. URL: http://books.google.com/books?vid=ISBN9781420071153.

Dekking, F. M. et al. (2005). *A modern introduction to probability and statistics: Understanding why and how*. Springer Science and Business Media. DOI: 10.1007/1-84628-168-7.

Du, D.-Z. and P. M. Pardalos (2013). *Minimax and applications*. Springer Science and Business Media. DOI: 10.1007/978-1-4613-3557-3.

Eichorn, H. et al. (2016). "A comparative study of programming languages for next-generation astrodynamics systems". In: *Proceedings of the 6th International Conference on Astrodynamics Tools and Techniques*, pp. 115–123. URL: 10.1007/s12567-017-0170-8.

Englander, J. A. (2013). "Automated interplanetary trajectory planning". PhD thesis. University of Illinois at Urbana-Champaign. URL: http://hdl.handle.net/2142/44367.

Englander, J. A. and A. C. Englander (2014). "Tuning Monotonic Basin Hopping". In: *International Symposium on Space Flight Dynamics 2014*. Vol. 24. URL: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140007521.pdf.

Fletcher, R. et al. (2002). "Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming". In: *SIAM Journal on Optimization* 13.3, pp. 635–659. DOI: 10.1137/S1052623499357258.

Gauss, C. F. (1857). *Theory of the motion of the heavenly bodies moving about the sun in conic sections: a translation of Carl Frdr. Gauss" Theoria motus": With an appendix. By Ch. H. Davis*. Little, Brown and Comp. URL: http://books.google.com/books?vid=ISBN1173799184.

Grosan, C. and A. Abraham (2007). "Hybrid evolutionary algorithms: Methodologies, architectures, and reviews". In: *Studies in Computational Intelligence* 75, pp. 1–17. DOI: 10.1007/978-3-540-73297-6_1.

Hansen, N. and A. Ostermeier (2001). "Completely derandomized self-adaptation in evolution strategies". In: *Evolutionary Computation* 9.2, pp. 159–195. DOI: 10.1162/106365601750190398.

Holland, J. H. (1975). "Adaptation in natural and artificial systems". In: *University of Michigan Press*. DOI: doi.org/10.1137/1018105.

Hollenbeck, G. R. (1975). "New flight techniques for outer planet missions". In: *AIAA Conference on the Exploration of the Outer Planets*. URL: http://adsabs.harvard.edu/abs/1975aiaa.confY....H.

Ishimatsu, T., J. Hoffman, and O. D. E. Weck (2011). "Method for Rapid Interplanetary Trajectory Analysis using ΔV Maps with Flyby Options". In: *Journal of the British Interplanetary Society* 64, pp. 204–213. URL: http://hdl.handle.net/1721.1/99890.

Izzo, D. (2010). "Global optimization and space pruning for spacecraft trajectory design". In: *Spacecraft Trajectory Optimization*. Ed. by B. Conway, pp. 178–201. DOI: 10.1017/CBO9780511778025.008.

— (2014). "Revisiting Lambert's problem". In: *Celestial Mechanics and Dynamical Astronomy* 121.1, pp. 1–15. DOI: 10.1007/s10569-014-9587-y.

Izzo, D., D. Hennes, and A. Riccardi (2015). "Constraint handling and multi-objective methods for the evolution of interplanetary trajectories". In: *Journal of Guidance, Control, and Dynamics* 38.4, pp. 792–800. DOI: 10.2514/1.G000619.

Izzo, D., C. Sprague, and D. Tailor (2018). "Machine learning and evolutionary techniques in interplanetary trajectory design". In: URL: https://pdfs.semanticscholar.org/5f7d/d8fcc8c996f92899bf92c99230ddb72e9a21.pdf.

Izzo, D. et al. (2007). "Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories". In: *Journal of Global Optimization* 38.2, pp. 283–296. DOI: 10.1007/s10898-006-9106-0.

Izzo, D. et al. (2013). "Search for a grand tour of the jupiter Galilean moons". In: *Proceedings of the 15th annual conference on genetic and evolutionary computation*, pp. 1301–1308. DOI: 10.1145/2463372.2463524.

Janin, G. and M. A. Gòmez-Tierno (1985). "The genetic algorithms for trajectory optimization". In: *Stockholm International Astronautical Congress*. URL: http://adsabs.harvard.edu/abs/1985stoc.iafcS....J.

Jones, D. R., C. D. Perttunen, and B. E. Stuckman (1993). "Lipschitzian optimization without the Lipschitz constant". In: *Journal of Optimization Theory and Applications* 79.1, pp. 157–181. DOI: 10.1007/BF00941892.

Jones, D. (2016). *Trajectories for flyby sample return at Saturn's moons*. DOI: 10.2514/6.2016-5266.

Karaboga, D. and B. Basturk (2007). "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm". In: *Journal of Gobal Optimization* 39.3, pp. 459–471. DOI: 10.1007/s10898-007-9149-x.

Kazimipour, B., X. Li, and A. K. Qin (2014a). "A review of population initialization techniques for evolutionary algorithms". In: *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*, pp. 2585–2592. DOI: 10.1109/CEC.2014.6900618.

Kazimipour, B., X. Li, and a. K. Qin (2014b). "Effects of population initialization on differential evolution for large scale optimization". In: *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 2404–2411. DOI: 10.1109/CEC.2014.6900624.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). "Optimization by simulated annealing". In: *Science* 220.4598, pp. 671–680. DOI: 10.1126/science.220.4598.671.

Knittel, J et al. (2017). "Automated sensitivity analysis of interplanetary trajectories for optimal mission design". In: *Proceedings International Symposium on Space Flight Dynamics* 14, pp. 1–8. URL: http://hdl.handle.net/2060/20170003691.

Lauretta, D. S. (2012). "An overview of the OSIRIS-REx asteroid sample return mission". In: *Lunar and Planetary Science Conference*. Vol. 43. URL: https://www.lpi.usra.edu/meetings/lpsc2012/pdf/2491.pdf.

Locatelli, M. and F. Schoen (2003). "Efficient algorithms for large scale global optimization: Lennard-Jones clusters". In: *Computational Optimization and Applications* 26.2, pp. 173–190. DOI: 10.1023/A:1025798414605.

Longuski, J. M. and S. N. Williams (1990). "Automated design of multiple encounter gravity-assist trajectories". In: *AIAA/AAS Astrodynamics Conference*, pp. 985–994. URL: http://adsabs.harvard.edu/abs/1990asdy.conf..985L.

McAdams, J. V. et al. (2006). "Trajectory design and maneuver strategy for the MESSENGER mission to Mercury". In: *Journal of Spacecraft and Rockets* 43.5, pp. 1054–1064. DOI: 10.2514/1.18178.

McCarthy, D. D. and M. A. Hoskin (1998). *The julian and modified julian dates*. Vol. 29. Journal for the history of astronomy 4. SAGE Publications, pp. 327–330. URL: http://adsabs.harvard.edu/full/1998JHA....29..327M.

Meltzer, M. (2013). *Mission to Jupiter: A history of the Galileo project*. The NASA History Series. CreateSpace Independent Publishing Platform. URL: https://books.google.nl/books?id=r34nnwEACAAJ.

Mezura-Montes, E., J. Velázquez-Reyes, and C. A. Coello Coello (2006). "A comparative study of differential evolution variants for global optimization". In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. May, pp. 485–492. DOI: 10.1145/1143997.1144086.

Minovitch, M. A. (1963). *The determination and characteristics of ballistic interplanetary trajectories under the influence of multiple planetary attractions*. Tech. rep. California Institute of Technology, Jet Propulsion Laboratory. URL: http://www.gravityassist.com/IAF3-2/Ref.3-160.pdf.

Musegaas, P. (2012). "Optimization of space trajectories including multiple gravity assists and deep space maneuvers". Master thesis. Delft University of Technology. URL: https://repository.tudelft.nl/.

Oberth, H (1929). *Wege zur Raumschiffahrt*. R. Oldenbourg. DOI: https://doi.org/10.1007/BF01700815.

Olds, A. D., C. A. Kluever, and M. L. Cupples (2007). "Interplanetary mission design using differential evolution". In: *Journal of Spacecraft and Rockets* 44.5, pp. 1060–1070. DOI: 10.2514/1.27242.

Price, K. V., R. M. Storn, and J. A. Lampinen (2005). *Differential evolution - a practical approach to global optimization*. Springer-Verlag Berlin Heidelberg. DOI: 10.1007/3-540-31306-0.

Raphson, J. (1702). *Analysis aequationum universalis seu ad aequationes algebraicas resolvendas methodus generalis, & expedita, ex nova infinitarum serierum methodo, deducta ac demonstrata*. typis Tho. Braddyll, prostant venales apud Johannem Taylor. URL: https://archive.org/details/bub_gb_4nlbAAAAQAAJ.

Schlueter, M. (2014). "MIDACO software performance on interplanetary trajectory benchmarks". In: *Advances in Space Research* 54.4, pp. 744–754. DOI: 10.1016/j.asr.2014.05.002.

Schlueter, M., J. A. Egea, and J. R. Banga (2009). *Extended ant colony optimization for non-convex mixed integer nonlinear programming*. DOI: 10.1016/j.cor.2008.08.015.

Storn, R and K Price (1997). "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of Global Optimization* 11.4, pp. 341–359. DOI: 10.1023/A:1008202821328.

Stracquadanio, G. et al. (2011). "Design of robust spacecraft trajectories". In: *Research and Development in Intelligent Systems XXVIII*, pp. 341–354. DOI: 10.1007/978-1-4471-2318-7_26.

Strange, N. J. and J. M. Longuski (2002). "Graphical method for gravity-assist trajectory design". In: *Journal of Spacecraft and Rockets* 39.1, pp. 9–16. DOI: 10.2514/2.3800.

Strange, N. J. and J Sims (2002). "Methods for the design of V-infinity leveraging maneuvers". In: *Advances in the Astronautical Sciences* 109, pp. 1959–1976. DOI: 10.2514/2.3800.

Tsiolkovsky, K. E. (1903). "The exploration of cosmic space by means of reaction devices". In: *Scientific Review* 5.

Vallado, D. A. (2006). "Perturbed motion". In: *Modern Astrodynamics*. Ed. by P. Gurfill. Elsevier. Chap. 1, pp. 1–22. DOI: 10.1016/S1874-9305(07)80003-3.

Vasile, M and P. D. Pascale (2006). "Preliminary design of multiple gravity-assist trajectories". In: *Journal of Spacecraft and Rockets* 43.4, pp. 794–805.

Vasile, M. and M. Ceriotti (2010). "Incremental techniques for global space trajectory design". In: *Spacecraft Trajectory Optimization*. Ed. by B. Conway, pp. 202–237. DOI: https://doi.org/10.1017/CBO9780511778025.

Vasile, M., E. Minisci, and M. Locatelli (2010). "Analysis of some global optimization algorithms for space trajectory design". In: *Journal of Spacecraft and Rockets* 47.2, pp. 334–344. DOI: 10.2514/1.45742.

— (2011). "An inflationary differential evolution algorithm for space trajectory optimization". In: *IEEE Transactions on Evolutionary Computation* 15.2, pp. 267–281. DOI: 10.1109/TEVC.2010.2087026.

Vasile, M., E. A. Minisci, and M. Locatelli (2008). "On testing global optimization algorithms for space trajectory design". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Guidance, Navigation, and Control*, pp. 18–21. DOI: https://doi.org/10.2514/6.2008-6277.

Vavrina, M, J Englander, and D Ellison (2016). "Global optimization of N-maneuver, high-thrust trajectories using direct multiple shooting". In: *26th AAS/AIAA Spaceflight Mechanics Meeting*. URL: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160001644.pdf.

Vinko, T and D Izzo (2008). *Global optimisation heuristics and test problems for preliminary spacecraft trajectory design*. Tech. rep. URL: http://www.esa.int/gsp/ACT/doc/INF/pub/ACT-TNT-INF-2008-GOHTPPSTD.pdf.

Waltz, R. A. et al. (2006). "An interior algorithm for nonlinear optimization that combines line search and trust region steps". In: *Mathematical Programming* 107.3, pp. 391–408. DOI: 10.1007/s10107-004-0560-5.

Wolpert, D. H., M. Field, and W. G. Macready (2005). "Coevolutionary free lunches". In: *IEEE Transactions on Evolutionary Computation* 9.6, pp. 721–735. DOI: 10.1109/TEVC.2005.856205.

Wolpert, D. H. and W. G. Macready (1997). "No free lunch theorems for optimisation". In: *IEEE Trans. on Evolutionary Computation* 1.1, pp. 67–82. DOI: 10.1023/A:1021251113462.

Woolley, R. C. and C. W. Whetsel (2013). "On the nature of Earth-Mars pork chop plots". In: *23rd AAS/AIAA Spaceflight Mechanics Meeting*. URL: https://trs.jpl.nasa.gov/bitstream/handle/2014/44336/13-0679_A1b.pdf?sequence=1.

Zaharie, D. (2009). "Influence of crossover on the behavior of Differential Evolution Algorithms". In: *Applied Soft Computing Journal* 9.3, pp. 1126–1138. DOI: 10.1016/j.asoc.2009.02.012.

Zuo, M., G. Dai, and L. Peng (2017). "EPDEII: A significant algorithm to search the optimal solution for global optimization of multi-gravity assist trajectory". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 51, p. 095441001771400. DOI: 10.1177/0954410017714009.

Appendices

Appendix A

Solution space analysis

This appendix contains several figures that provide qualitative insight into the shape of the solution spaces of the trajectory function and the a priori objective function.

Table A.1: Comparison of the widths of the global minimum basins and the number of local minima in Figure A.1 and A.2.

	Cassini1		Cassini2	
Variable	**Minima**	**Basin widths**	**Minima**	**Basin widths**
T0	14	19%	31	3%
T1	7	43%	13	14%
T2	12	12%	10	32%
T3	6	33%	5	16%
T4	2	70%	3	29%
T5	1	100%	1	100%

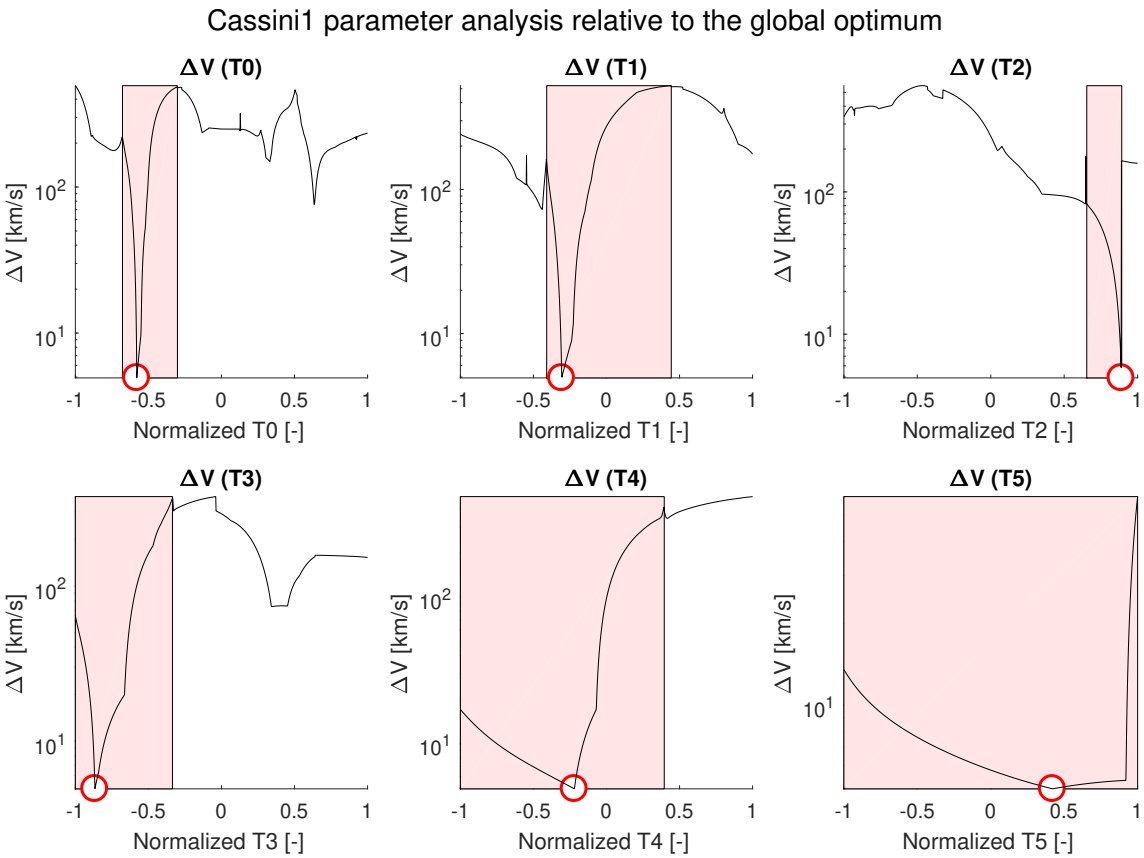


Figure A.1: Cassini1 ΔV as function of the departure epoch and times of flight of the legs, while all other variables are fixed in the global minimum. The control variables have been normalized on the interval [-1, 1] (hence the names of the x-axes). The red transparent rectangle corresponds to the global minimum basin in the respective cross section. The widths of the global minimum basins are provided in Table A.1 together with the numbers of minima in the cross sections.

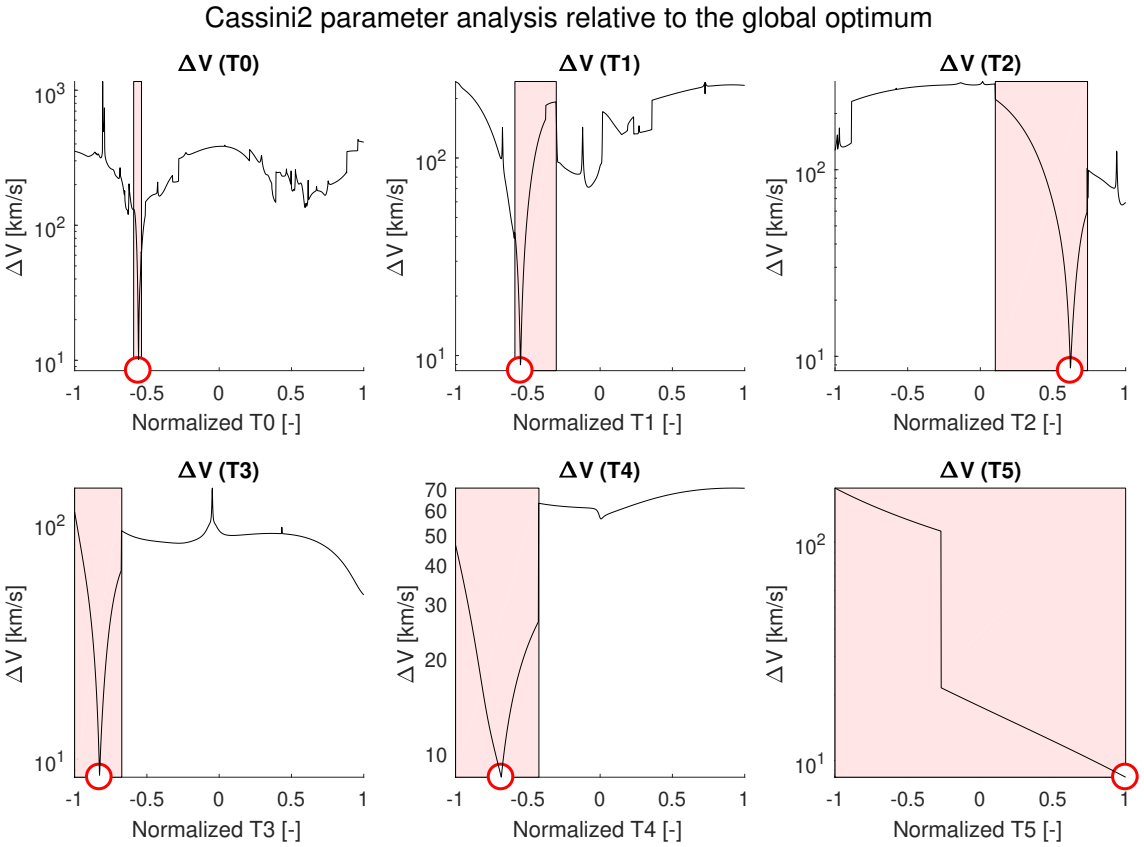


Figure A.2: Cassini2 ΔV as function of the departure epoch and times of flight of the legs, while all other variables are fixed in the global minimum. The control variables have been normalized on the interval [-1, 1] (hence the names of the x-axes). The red transparent rectangle corresponds to the global minimum basin in the respective cross sections. The widths of the global minimum basins are provided in Table A.1 together with the numbers of minima in the cross sections. The basins in this figure are narrower than those in Figure A.1.

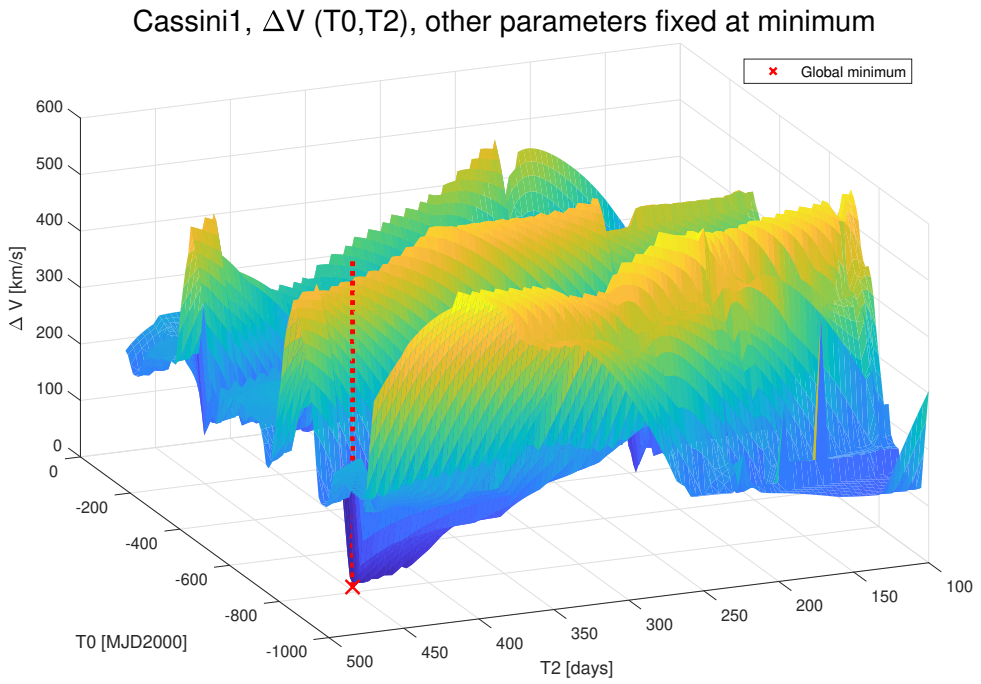


Figure A.3: A surface plot of ΔV of Cassini1 as function of the departure epoch $T0$ and time of flight of the second leg $T2$. It can be seen that the global minimum is very sensitive to increases of $T2$. Note the reversed direction of the $T2$ axis.

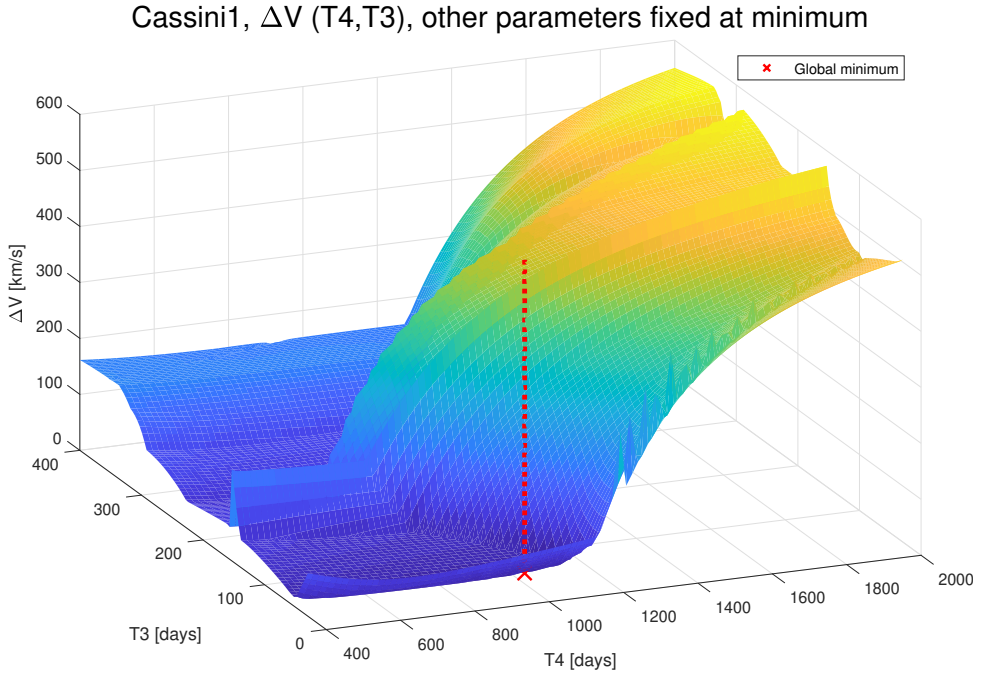


Figure A.4: A surface plot of ΔV of Cassini1 as function of the times of flight $T3$ and $T4$. The ΔV budget is much less sensitive to these variables than to $T2$ and $T0$ in the plot above.

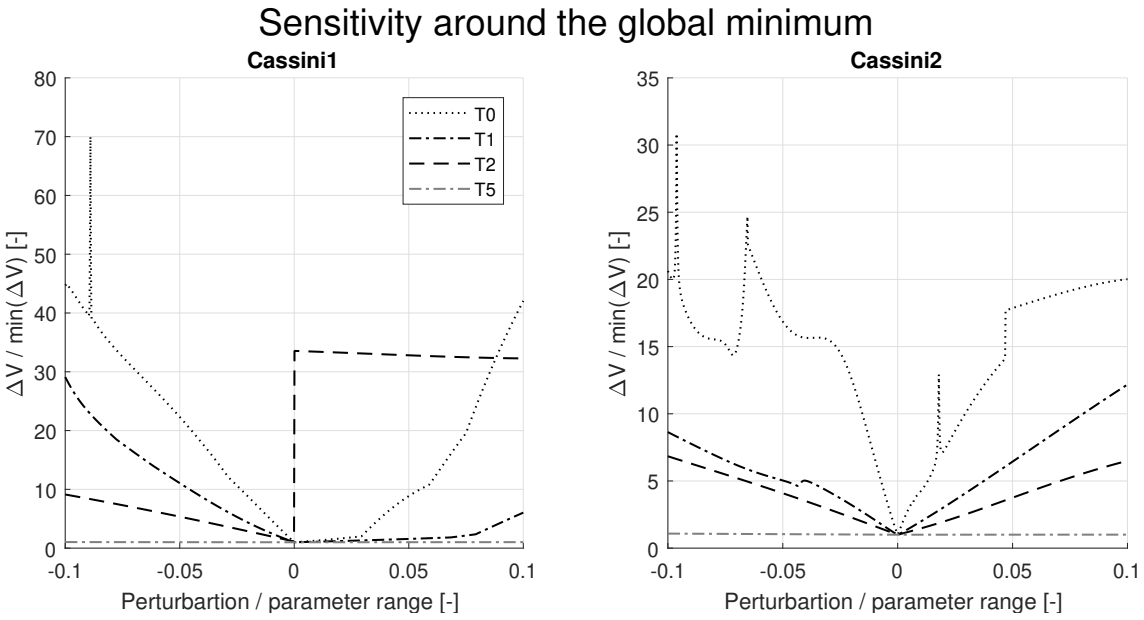


Figure A.5: Sensitivity of several control variables around the global minimum (0,1). Each control variable is varied between -10% and -10% of the variable ranges, while the others remain fixed at the value of the global minimum (0,1).

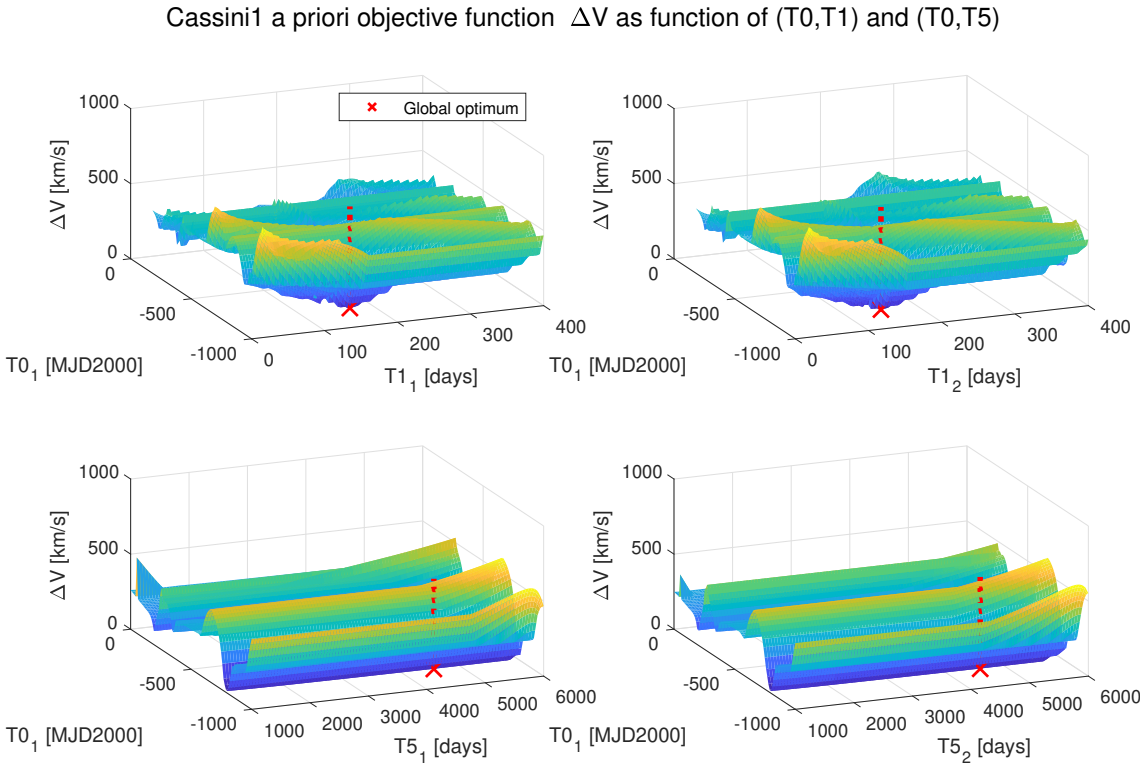


Figure A.6: Comparison between the surfaces of ΔV as function of first departure epoch $T0_1$ and two pairs of variables: ($T1_1$, $T1_2$) and ($T5_1$, $T5_2$). It can be observed that the solution surfaces are very similar for paired variables. Departure epoch interval $\Delta T0 = 21$ days.

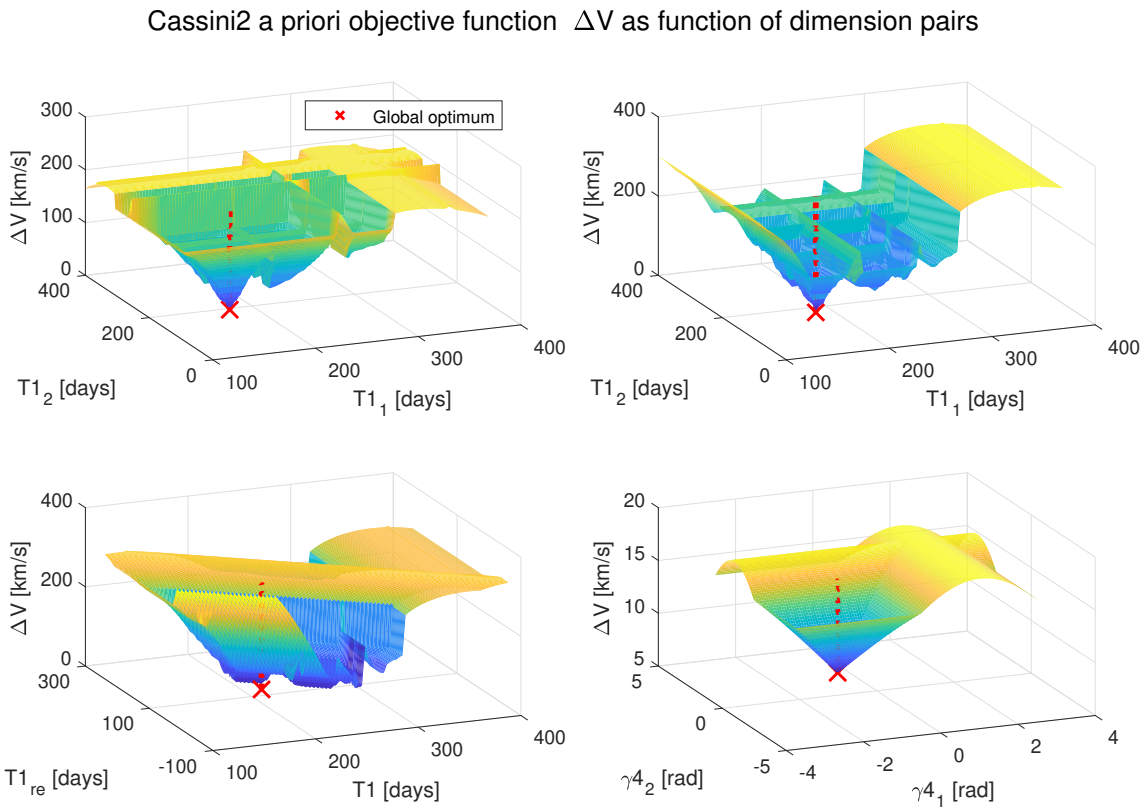


Figure A.7: Illustration of ΔV as function of control variables. The top left surface shows $\Delta V(T1_2, T1_1)$ for $\Delta T0 = 0$ days, which leads to symmetry over the $T1_1 = T2_2$ plane. The right figure shows $\Delta V(T1_2, T1_1)$ for $\Delta T0 = 21$ days. This figure is much less symmetrical, demonstrating the sensitivity of ΔV to $\Delta T0$. The bottom left surface shows the solution of the relative objective function formulation for $\Delta T = 21$ days and bounds of 10% of the $T1$ range on relative dimension $T1_{re}$. The global minimum lies on the $T1_{re}$ axis. The right bottom corner is taken by ΔV $(\gamma 4_1, \gamma 4_2)$ for $\Delta T0 = 21$ days, $\gamma 4$ is an example of a variable that is not sensitive to $\Delta T0$, judging from the diagonal symmetry.

Appendix B

Monotonic basin hopping

This appendix addresses the implementation of a monotonic basin hopping algorithm that has not been successful on the considered objective functions.

B.1 Algorithm description

Monotonic basin hopping is a very simple, yet powerful optimization algorithm. It combines a Monte-Carlo-type search method with local optimization. It starts by initializing a random starting point. Next, a local optimizer is run and the new optimal value is stored. Then, this solution is randomly perturbed in each dimension and from this perturbed location, the local optimizer is run again. If the new local optimum is better than the previous, the process will repeat itself from there. If it is not better, the process is repeated from the previous optimum. This scheme is looped until a user-specified criterion is satisfied. Figure B.1 illustrates the working principle of monotonic basin hopping.

Monotonic basin hopping works specifically well if the solution surface descends over long range towards the global minimum. In that case, an improving fitness function in a certain region is a good indicator that the global minimum may be found in that direction as well. Note that the fitness surface does not need to be strictly monotonically decreasing, the hopping plus optimizing allows for dealing with locally unfit regions by hopping over them (Locatelli and Schoen 2003). A pseudo code transcription has been included as Algorithm 11.

Algorithm 11 Pseudo code of the monotonic basin hopping algorithm

run NLP solver from a random location and save $\mathbf{x_{best}}$ and $\Delta V_{best} := f(\mathbf{x_{best}})$
while stop criteria are not satisfied **do**
  perturb $\mathbf{x_{best}}$ randomly to generate $\mathbf{x_{per}}$
  run NLP solver on $\mathbf{x_{per}}$ to generate new $\mathbf{x_{new}}$
  if $\Delta V(\mathbf{x_{per}}) < \Delta V(\mathbf{x_{best}})$ **then**
    $\mathbf{x_{best}} := \mathbf{x_{new}}$
  end if
end while

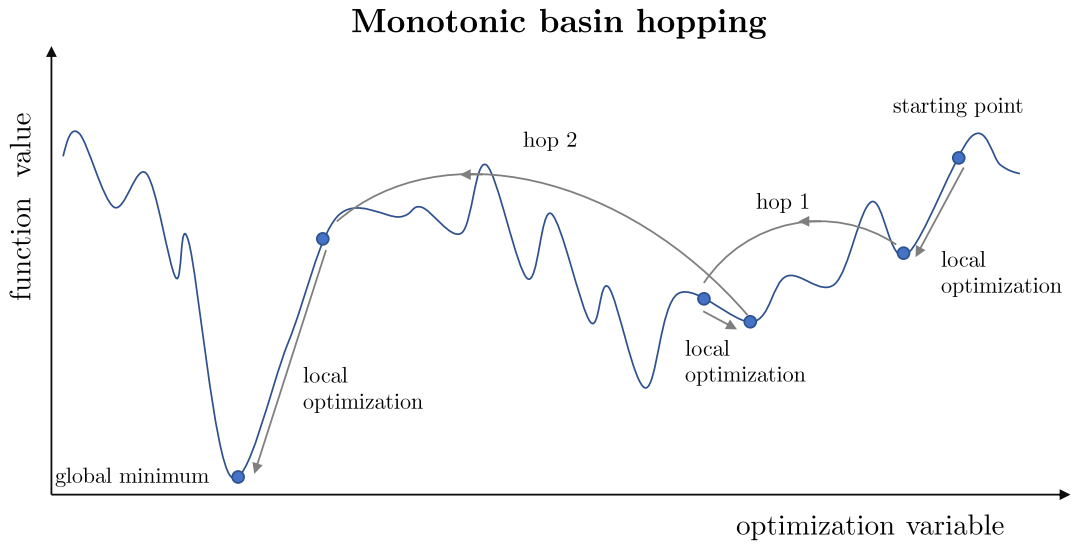


Figure B.1: Visual interpretation of the monotonic basin hopping algorithm.

B.1.1 Local optimization approach

The nonlinear programming algorithm that is used in this variant is MATLAB's built-in `fmincon` function. This is the same local optimizer as used by the monotonic basin hopping algorithm described by Vasile and Ceriotti (2010). It uses the *interior-point algorithm*, which numerically estimates the gradient of the objective function (Byrd, Gilbert, and Nocedal 2000; Fletcher et al. 2002; Waltz et al. 2006). The lack of lack of analytical derivatives is expected to lead to longer computation times; additional function evaluations are required to make numerical estimates of the derivatives. The effect of discontinuities due to penalty functions and the discrete minimax approach of the single objective function is expected to undermine the performance of the local optimizer further.

It must be stated as well that, besides the above mentioned limitations, more optimal nonlinear programming algorithms exist. For example, EMTG uses the (commercial) SNOPT library.

B.1.2 Perturbing operator

The goal of the perturbing operator is to hop to a new basin. Since the exact shape of the objective function is not known, the perturbation is determined randomly. The probability distribution of the perturbing operator has a major impact on the performance of the monotonic basin hopping algorithm. Two key considerations are the following.

1. An improving fitness function value may be an indicator that the optimizer is approaching a global minimum. This implies that perturbations should be small, in order to explore the current region extensively.

2. In solution spaces with multiple funnel-shaped regions, the optimizer might get stuck at the bottom of one funnel, while the global minimum is at the bottom of another funnel. This demands either a restart or a perturbing operator that allows the search to move to a different region.

Both considerations mentioned above seem to be conflicting. However, a study by Englander and

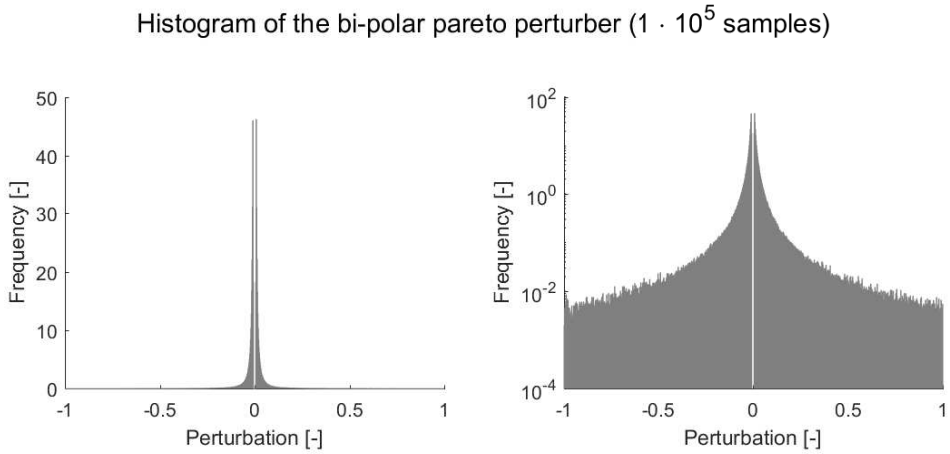


Figure B.2: Histogram of the bi-polar Pareto distribution. The area of the histogram is normalized to equal 1.

.

Englander (2014) was specifically aimed at finding a probability distribution for a perturbing operator that deals well with both issues on objective functions for interplanetary trajectories. It must be noted that this study focused on low-thrust trajectories, but the results seem very promising.

The best perturbing operator was found to use a bi-polar Pareto distribution. Its probability density function is shown in Figure B.2. It generates a random number according to Equation B.1, in which s is -1 or 1 with a 50% probability, r a random number between 0 and 1 (uniform) and it was found that $\epsilon = 1 \cdot 10^{-13}$ and $\alpha = 1.01$ yield the best results (Englander and Englander 2014).

$$\text{rand} = \frac{s}{\epsilon} \frac{(\alpha - 1)}{\left(\frac{\epsilon}{\epsilon + r}\right)^{-\alpha}} \tag{B.1}$$

The probability density distribution of this function is provided in Figure B.2. Two important properties of this distribution are the following.

1. The vast majority of the likely perturbations are small, with a fixed minimum hop distance determined by α and ϵ, in this case 0.741%. This means that the current region is explored very well.

2. The distribution has a long tail, which means that further regions are also visited, be it less frequently than region around the current minimum. This prevents the search from getting stuck in a single funnel. This tail only is visible on the logarithmic probability scale in the right graph of Figure B.2 since small perturbations remain much more likely.

An advantage of the long tail of this distribution is that monotonic basin hopping theoretically does not need to be restarted. Although it remains sensitive to the initial guess, periodic exploration of other regions would eventually lead to the algorithm finding the global minimum. Still, the decision between restarting or letting a single for effort run for a longer period depends on the problem. One argument against running multiple times, is the required introduction of new parameters that define stopping and restarting criteria. These may once again be required to be tuned per problem, increasing the need for human interference in each optimization problem.

B.2 Constraints

Boundary constraints are handled at two levels, in the perturbing operator and in the local optimizer. When a perturbation violates a boundary constraint in a certain dimension, the perturbing operation is reinitialized from the best known position in that dimension.

The interior point algorithm handles inequality constraints through a *barrier function* of which the value approaches infinity outside the feasible region of the optimization (Byrd, Gilbert, and Nocedal 2000; Fletcher et al. 2002; Waltz et al. 2006). This approach is also capable of handling nonlinear constraints, thus making it possible to avoid the penalty function that is included in the objective function.

Appendix C

Verification functions for differential evolution

C.1 Rosenbrock saddle

The following relation describes the Rosenbrock saddle, also known as the second De Jong function. Although it has only two dimensions, it is considered a difficult optimization problem. The minimum is located at (1,1) (Storn and Price 1997).

$$f(x_1, x_2) = 100 \cdot \left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 \tag{C.1}$$

C.2 Rastrigin function

The Rastrigin function is a highly modal function due to the cosine term. Its general N dimensional form is the following (Storn and Price 1997).

$$f(\mathbf{x}) = 10 \cdot N + \sum_{i=1}^{N} \left(x_i^2 - 10 \cdot \cos\left(2 \cdot \pi \cdot x_i\right)\right) \tag{C.2}$$

Appendix D

Mirror threshold research Cassini0

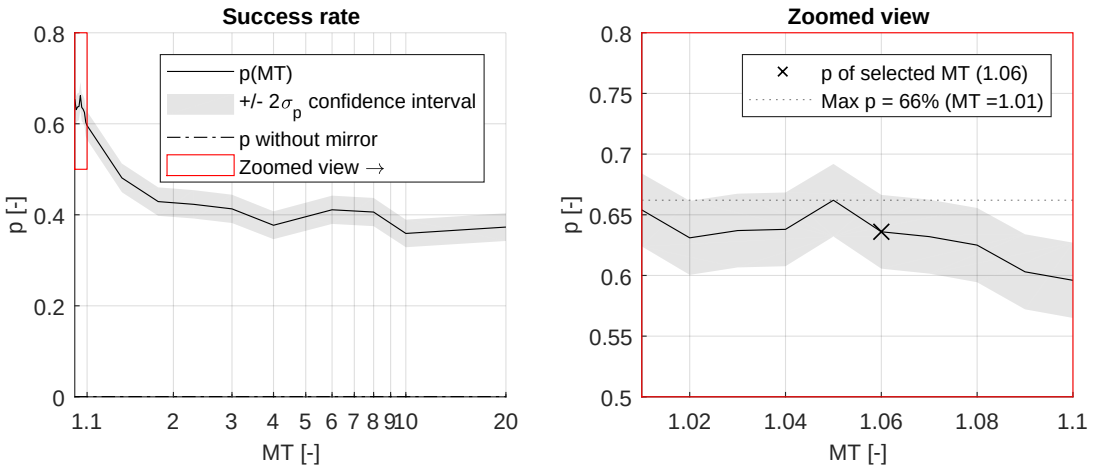


Figure D.1: The success rate for various mirror threshold values on the Cassini0 trajectory.

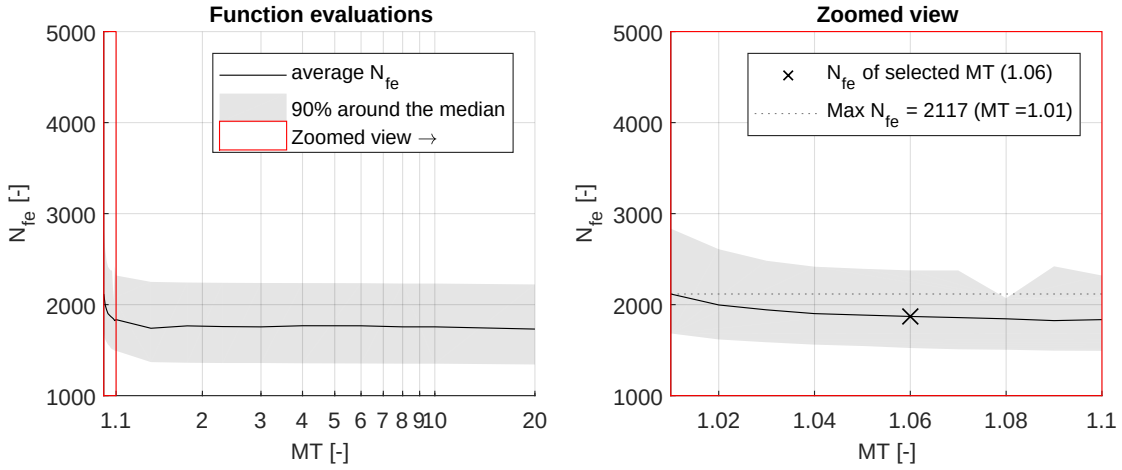


Figure D.2: The number of function evaluations for various mirror threshold values on the Cassini0 trajectory.

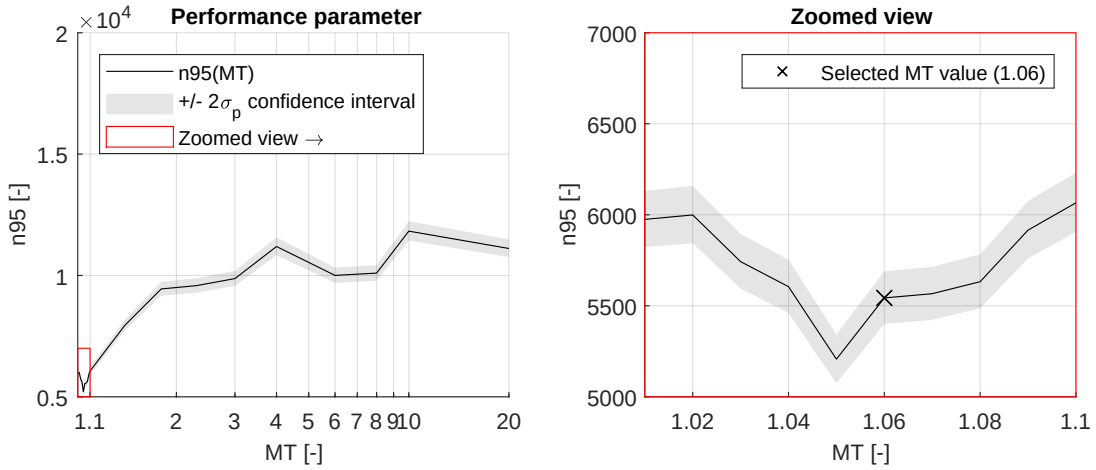


Figure D.3: The performance parameter for various mirror threshold values on the Cassini0 trajectory.

Appendix E

Contacted experts

Name	Institution	Subject
Jacob Englander	NASA Goddard Space Flight Center	General advice on various subjects
Martin Brennan	NASA Jet Propulsion Laboratory	Advice on TRACT (Brennan 2015)
Dario Izzo	ESA Advanced Concepts Team	Limited correspondence on PaGMO[1]
Stjepan Picek	Delft University of Technology, faculty of Electrical Engineering, Mathematics and Computer Science	Teleconference on evolutionary optimization algorithms
Jeannette Heiligers	Delft University of Technology, faculty of Aerospace Engineering	Teleconference on the a priori method
Matthijs Joosten	Delft University of Technology, faculty of Electrical Engineering, Mathematics and Computer Science	Statistics and confidence intervals