Cluster-based search space pruning in single leg low-thrust trajectory optimisation problems

Elmar G.F.B. Puts

Technische Universiteit Delft

TU Delft

Cluster-based search space pruning in single leg low-thrust trajectory optimisation problems

by

Elmar G.F.B. Puts

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday December 21, 2021 at 2:00 PM.

Student number:	4373456	
Thesis committee:	Prof. Dr. Ir. P. N. A. M. Visser,	TU Delft, chair
	Ir. K. J. Cowan, MBA,	TU Delft, supervisor
	Dr. A. Menicucci,	TU Delft

This thesis is confidential and cannot be made public until December 31, 2023.

Cover image (Saturn): https://www.peakpx.com/en/hd-wallpaper-desktop-kuyjn, retrieved on 31/10/2021

TU Delft

Preface

This report marks the end of my 7-year long TU Delft journey and is the product of the thesis that concludes the Aerospace Engineering Master. This document summarises the effort that went into exploring an unknown jungle, but does not include each and every path that was taken to reach the conclusions that are presented here. Nor should it, as it would have become a very lengthy, and most of all, utterly boring treatise. Still, many avenues were taken, and almost equally many were abandoned again. This thesis is thus to be treated as a first attempt at a road map in the jungle of clustering in the field of global trajectory optimisation.

Exploring new territory is of course seldom done on one's own. Indeed, this work would never have been completed without the support of many people. First of all, I would like to thank my thesis supervisor Kevin Cowan. Our weekly meetings were a joy and always gave me an impulsive boost[1], ready to try out the numerous (new) things we discussed. Unfortunately, we have never had the opportunity to have our meetings in person due to the Corona virus measures. Nevertheless, I will always fondly remember the vivid and interesting discussions we had during our Zoom sessions.

Next, I want to express my gratitude to my dear Stephan for always being there, especially when the roller coaster went down and motivation was in its zenith. Even when I did not see the light at the end of the tunnel, you managed to explain how I could find it, despite all the adverse circumstances. Mom, dad: thank you for listening to my frustrations, explanations (often in vain, I suppose...), and supporting me in the times I needed it most. The trips to Limburg were always a welcome distraction and felt a bit like going to a Kurort. Then there are, of course, those who made my life in Delft (especially during Covid-lockdown times) enjoyable while writing this thesis, and whom I would like to thank: the PlanSci-bazen (Bart, Viktor, Thijs, and Bob) for the numerous online Civilization sessions, drinks, and board games; Thijme, Marijn, Laura, Willem, Liv, Olivier, Bart, Daniël, Kyle, Zhouxin, Noortje, and Ilse, for the D&D sessions, dinners, and/or drinks; Corinna, for the proofreading and wonderful discussions on linguistics, language learning, and other random stuff, and everybody else who contributed to this thesis in one way or another.

"I wish there was a way to know you are in the good old days before you actually left them", a wise man once said,[2] and, looking back, I think this quote is very applicable to the last 7 wonderful years that I spent in Delft, Rome, the US, Canada, and Montpellier. I am sure that I will always cherish both the little and great things I got to experience, and think back to them as being 'the good old days'. Of course, the time has come to enter a new phase in life, but my time as a student will always have a special place in my heart.

Elmar Puts
Delft, December 2021

[1]Pun intended.
[2]Andy Bernard, The Office US

Abstract

Many contemporary interplanetary missions use efficient low-thrust engines to reach the far corners of our Solar System. Their trajectories, however, have proven to be complicated to optimise due to the non-impulsive manoeuvres involved in low-thrust spaceflight. Even though shaping methods have been used extensively to reduce the computational burden, multiple-gravity assists and the presence of constraints create significant computational hurdles. Reducing the number of fitness evaluations during optimisation is one way of speeding up the search and can be done by 'pruning away' regions of infeasible trajectories. In this research, we approach this by applying clustering, an unsupervised machine learning approach, to single leg trajectory optimisation problems based on a hodographic shaping trajectory model in combination with restricted two-body dynamics. Through clustering, groups of promising trajectories can be isolated so that unwanted regions can be discarded. Earth — Mars, Earth — Venus, and Earth — 9P/Tempel 1 trajectories are used as test cases and are shown to exhibit periodic behaviour (related to the synodic periods of the departure and target bodies) that enables clustering on grid search-generated datasets. Different clustering algorithms were compared using the Silhouette, Davies-Bouldin, and Calinski-Harabasz internal validation indices. However, traditional clustering algorithms such as (H)DBSCAN, OPTICS, KMeans, and Gaussian Mixture Models, failed to robustly provide clusterings that can be used for pruning because of the oblong shape of the clusters and the absence of data/noise density differences due to the artificial nature of the problem. Instead, a multimodality-based clustering model called SkinnyDip was found to be much more promising for this task. This algorithm comes with the additional advantage of having very few hyperparameters, eliminating the need for extensive parameter tuning.

List of Acronyms & Symbols

Acronyms

ABC	Artificial Bee Colony Optimisation
ACO	Ant Colony Optimisation
API	Application Programming Interface
CPU	Central Processing Unit
DB	Database
EA	Evolutionary Algorithm
ECDF	Empirical Cumulative Distribution Function
EDF	Empirical Distribution Function
ESA	European Space Agency
GS	Grid Search
ICRF	International Celestial Reference Frame
KDO	Knowledge-Driven Optimisation
MC	Monte Carlo
MGA	Multiple Gravity Assist
ML	Machine Learning
NFLT	No Free Lunch Theorem
NLP	Nonlinear Programming
PSO	Particle Swarm Optimisation
SQL	Structured Query Language
SQP	Sequential Quadratic Programming
SSB	Solar System Barycentre

Constants

$\mu_\odot$	Gravitational parameter of the Sun	$1.33 \cdot 10^{20}$ m^{3}s^{-2}
c	Speed of light	$2.998 \cdot 10^{8}$ m s^{-1}
G	Gravitational constant	$6.67 \cdot 10^{-11}$ m^3 kg s^{-2}
g_0	Gravitational acceleration on Earth	9.81 m s^{-2}

Greek symbols

η	Propulsive efficiency	[-]
μ	Gravitational parameter	m^{3}s^{-2}
ϕ	Initial phase angle	[rad]
$\Phi(\cdot)$	Mayer term in cost functional	
θ	Angle	[rad]

Latin symbols

A	Area	[m^2]
b	Terminal constraint	
C_R	Reflection coefficient	[-]
i	Index	[-]
I_{sp}	Specific impulse	[s]
j	Index	[-]
$J[\cdot]$	Cost functional	
k	Constant	[-]
$L(\cdot)$	Lagrange term in cost functional	
m	Mass	[kg]
P	Power	[W] or [J s^{-1}]
p	Path constraint	
t	Time	[s]
V	Velocity function	
v	Velocity base function	
W	Solar radiation flux	[W m^{-2}]
f	Function	
$\mathbf{F}$	Force	[N]
$\mathbf{f}$	Specific force	[N kg^{-1}]
$\mathbf{r}$	Position vector	[m]
$\mathbf{u}$	Control vector	
$\mathbf{x}$	State vector	

Other symbols

$\dot{\square}$	First derivative with respect to t
$\ddot{\square}$	Second derivative with respect to t
$\square_i$	Initial value
$\square_f$	Final value
$\square_r$	Radial direction
$\square_\theta$	Tangential direction
$\square_z$	Out-of-plane direction
♈	Vernal equinox

List of Figures

List of Tables

Contents

1 Introduction

The advent of low-thrust, high specific impulse orbital engines and the increasing feasibility of solar sail-equipped spacecraft have proven a blessing and a curse for the space exploration community. Low-thrust spacecraft can now efficiently reach the far corners of our solar system, while the application of continuous thrust has greatly complicated finding optimal interplanetary trajectories. Low-thrust engines require quasi-impulsive manoeuvres to be replaced by continuous thrust trajectories, transforming the optimisation problem into an optimal control problem [12, 13].

Although these optimal control problems could in theory be solved analytically, this has proved to be a daunting task in most realistic situations. Constraints, complex dynamics, and path dependencies render it practically impossible to find the necessary expressions for these analytical methods, and numerically computing optimal control problems is computationally very expensive [12]. For these reasons, the focus has shifted towards other numerical methods that are quicker, allowing for a more thorough exploration of the trajectory design space. In the past decade or two, the so-called shape-based methods have become a popular choice as the tool of trade. They are fast and accurate enough for exploratory tasks, and thus have allowed mission designers to cover vast numbers of different trajectories in very complex interplanetary missions, such as JUICE and DAWN.

Although computational resources are becoming more and more abundant and powerful, the optimisation problems still present many obstacles to engineers and researchers. The design space is often multi-dimensional (e.g., when gravity assists are involved) and multiple objectives as well as (path) constraints are involved, forcing mission planners to resort to heuristics to solve the problem at hand. These algorithms then need to perform many computationally expensive fitness evaluations, causing the optimisation procedure to take infeasible amounts of time.

This sets the stage for *search space pruning*: a set of different approaches to making the search space less extensive. This means that in turn fewer fitness evaluations need to take place, drastically reducing optimisation run times and cost. One of these approaches is to perform knowledge extraction in the form of machine learning, which can help to prune the search space. Machine learning has gained much popularity in the past decade and is used in a broad spectrum of applications, ranging from marketing to disease treatment. Its use in trajectory optimisation is, however, still in its infancy.

In this thesis, we will explore the application of *clustering*, an unsupervised machine learning method, to interplanetary trajectory optimisation problems. Departing from the hodographic shaping method for computing single-leg trajectories, we will investigate how clustering methods can be used to extract knowledge from the problem and use that knowledge to prune the search space and improve the subsequent optimisation.

1.1. Research framework

In order to reach this objective, the following research question was formulated:

To what extent can clustering achieve search space pruning in single leg low-thrust trajectory optimisation problems?

To answer this question, a number of sub-questions were formulated that deconstruct the complex problem into smaller pieces:

1. *How is pruning used in conjunction with machine learning to improve heuristic optimisation algorithms in the preliminary design phase?*
2. *How are low-thrust trajectory design problems modelled in the preliminary design phase?*
3. *How can clusters be used in pruning interplanetary transfer problems?*
   (a) *How can we know if a certain data set, generated during optimisation, is clusterable?*
   (b) *If a transfer trajectory problem is clusterable, what patterns can be used by a clustering algorithm?*
   (c) *Do these patterns relate to physical properties of the astrodynamical problem?*
4. *What clustering algorithms are suitable to perform this task?*
   (a) *Are there properties of the problem that could give an indication of a match between a specific class of clustering algorithms and the trajectory optimisation problem?*
   (b) *Why are some classes of clustering algorithms more suitable in low-thrust trajectory optimisation than others?*
5. *In what way can clusters, found by a clustering algorithm, be validated in a situation where no ground truth data is available?*
6. *How should the design variables of the transfer problem be transformed to obtain an optimal clustering?*
7. *How can clustering-assisted pruning be used in a heuristic optimisation algorithm to increase global optimisation performance?*

1.2. Report structure

The goal of this report is to answer all of these sub-questions and the main research question by presenting the different steps that were taken and the results that were obtained during the thesis. It is structured as follows: the first part will present the scientific paper, which describes the context, methodology, results, conclusions, and recommendations of the research. Next, the second part contains relevant background: global optimisation, astrodynamics, shape-based methods, and clustering will be discussed. The last part will examine the details of the implementation of the optimisation, trajectory computation, and clustering, as well as the trajectory database that was used during this thesis.

I

Research

THE FIRST PART OF THIS REPORT contains the scientific article, which was written in parallel with this report, and will describe the methodology, results, discussion, and conclusions. It is included verbatim in Chapter 2, which is followed by a chapter containing the recommendations in a more elaborate fashion (Chapter 3).

2
Paper

Cluster-based search space pruning in single leg low-thrust trajectory optimisation problems

Elmar G.F.B. Puts[a], Kevin J. Cowan[a]

[a]*Delft University of Technology, Faculty of Aerospace Engineering, Kluyverweg 1, 2629 HS Delft, Netherlands*

ABSTRACT

Many contemporary interplanetary missions use efficient low-thrust engines to reach the far corners of our Solar System. Their trajectories, however, have proven to be complicated to optimise due to the non-impulsive manoeuvres involved in low-thrust spaceflight. Even though shaping methods have been used extensively to reduce the computational burden, multiple-gravity assists and the presence of constraints create significant computational hurdles. Reducing the number of fitness evaluations during optimisation is one way of speeding up the search and can be done by 'pruning away' regions of infeasible trajectories. In this research, we approach this by applying clustering, an unsupervised machine learning approach, to single leg trajectory optimisation problems based on a hodographic shaping trajectory model in combination with restricted two-body dynamics. Through clustering, groups of promising trajectories can be isolated so that unwanted regions can be discarded. Earth — Mars, Earth — Venus, and Earth — 9P/Tempel 1 trajectories are used as test cases and are shown to exhibit periodic behaviour (related to the synodic periods of the departure and target bodies), which enables clustering on grid search-generated datasets. Different clustering algorithms were compared using the Silhouette, Davies-Bouldin, and Calinski-Harabasz internal validation indices. However, traditional clustering algorithms such as (H)DBSCAN, OPTICS, KMeans, and Gaussian Mixture Models, failed to robustly provide clusterings that can be used for pruning, because of the oblong shape of the clusters and the absence of data/noise density differences due to the artificial nature of the problem. Instead, a multimodality-based clustering model called SkinnyDip was found to be much more promising for this task. This algorithm comes with the additional advantage of having very few hyperparameters, eliminating the need for extensive parameter tuning.

Nomenclature

Acronyms

CH	Calinski-Harabasz index
CVI	Clustering Validity Index
DB	Davies-Bouldin index
EA	Evolutionary Algorithm
EM	Earth – Mars transfer
ET	Earth – 9P/Tempel 1 transfer
EV	Earth – Venus transfer
GS	Grid Search
JD	Julian Day(s)
MC	Monte Carlo
NFLT	No Free Lunch Theorem
Sil	Silhouette index
V	Velocity function
v	Velocity base function

Symbols

$\ddot{\mathbf{x}}$	Second time derivative of $\mathbf{x}$	
$\dot{\mathbf{x}}$	First time derivative of $\mathbf{x}$	
Γ	Partitioning	
$\mathbf{f}$	Specific force	N kg^{-1}
$\mathbf{x}$	State vector	
$\prec$	Pareto dominance operator: first operand dominates the second	
d	Distance	
f	Clustering function	
i	Index	
j	Index	
$J[\cdot]$	Cost functional	
m	Number of objectives	
N	Number of clusters	
n	Number of decision variables	
O	Objective space	
S	Search/design space	
ΔV	Total velocity increase	m s^{-1}

e.g.f.b.puts@student.tudelft.nl (E.G.F.B. Puts); k.j.cowan@tudelft.nl (K.J. Cowan)

1. Introduction

The advent of low-thrust, high specific impulse orbital engines and the increasing feasibility of solar sail-equipped spacecraft have proven a blessing and a curse for the space exploration community. Low-thrust spacecraft can now efficiently reach the far corners of our solar system, while the application of continuous thrust has greatly complicated finding optimal interplanetary trajectories because of the arising optimal control problem [1, 2]. Analytical solutions exist only in very simplified cases and hence numerical methods are often required, albeit at substantial computational cost [1].

In the past decade or two, the so-called shape-based methods have become a popular choice as the tool of the trade, allowing for a relatively quick exploration of the trajectory design space in simple cases. When complex trajectories using multiple gravity assists are considered, however, these methods still need substantial amounts of computational resources. This sets the stage for *search space pruning*: a set of different approaches to making the search space less complex. One of these approaches is to perform knowledge extraction in the form of machine learning. Machine learning has gained much popularity in the past decade and is used in a broad spectrum of applications, ranging from marketing to disease treatment. Its use in trajectory optimisation problems, however, is still in its infancy.

In this paper we will explore the application of *clustering*, an unsupervised machine learning method, to interplanetary trajectory optimisation problems. Departing from the hodographic shaping method for computing single-leg trajectories, we will investigate how clustering methods can be used to extract knowledge from the problem and use that knowledge to prune the search space and improve the subsequent optimisation.

To reach this goal, we will start by highlighting the fundamentals in the first three chapters. Pruning (Section 2), hodographic shaping (Section 3), and clustering (Section 4) will be reviewed, providing the reader with adequate knowledge to understand the context of this paper. Section 5 will discuss the technicalities of using the hodographic shaping method to generate the trajectories, as well as details about the implementation. We then proceed with an endeavour into the world of pork chop plots in Section 6 to discover what the search space looks like and how it relates to the physics of the problem. This will allow us to find out *how* clustering can be used, and what is still necessary to actually carry out this task. As we will see, data preprocessing is essential and therefore an entire section, Section 7, is devoted to this step. Following this, we will discuss the application of clustering itself in Section 8 and multimodalility-based clustering in Section 9. Lastly, the conclusions and recommendations will be presented in Section 10.

2. Search Space Pruning

This section addresses the concept of pruning, one of the core pillars of this research. We will elaborate on its goal as well as briefly describe which pruning methods exist and have been tried before.

A mathematical optimisation problem in its most general form with design/search space S and objective space O can be defined as follows [3]:

Definition 1. An optimisation problem with n decision variables and m objectives and objective function $\mathbf{F} : S \mapsto O$ amounts to finding $\min_{\mathbf{x}\in S} \mathbf{F}(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ \dots, f_m(\mathbf{x})\}$ where $S \subseteq \mathbb{R}^n$ and $O \subseteq \mathbb{R}^m$. $\mathbf{F}$, S, and O are not required to have much structure, i.e., the objective function does not need to be convex, smooth, or differentiable.

We thus observe that the problem concerns a mapping between two – possibly constrained – sets. S in Definition 1 is the *feasible* decision domain, i.e. the domain subject to constraints. As the number n of decision variables increases, the dimensionality of this domain usually also increases. The objective space is often constrained as well and therefore some solutions of the problem may be infeasible, yielding the feasible objective space which we will call O, with $O \subseteq \mathbb{R}^m$. The two spaces and the mapping $\mathbf{F}(\mathbf{x})$ between them are visualised in Figure 1.

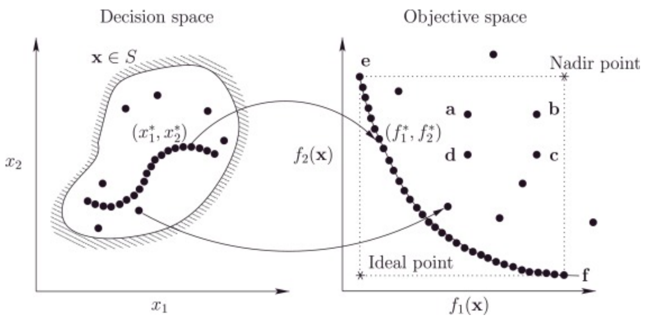

Figure 1: Mapping of points in the decision space S to the objective space O through $\mathbf{F}(\mathbf{x})$ [3].

When dealing with cases where $m > 1$, i.e., there is more than one objective, there is no notion of an absolute best solution, as the objectives are almost always conflicting: decreasing one objective will inevitably lead to increasing at least one of the others. In Figure 1 this is depicted as the curve of connected dots: the so-called *Pareto front* [4]. It represents the trade-off between the objectives and contains all solutions that are not Pareto dominated by other solutions, that is:

Definition 2. A solution $\mathbf{f}$ dominates $\mathbf{f}'$ if $f_i \le f_i' \,\forall\, f_i$ and if there is at least one f_i for which $f_i < f_i'$ is true. In this case we write $\mathbf{f_i} \prec \mathbf{f_i'}$. Thus, for a Pareto optimal solution $\mathbf{f_i}$ we must require that there be no $\mathbf{f_i'}$ such that $\mathbf{f_i'} \prec \mathbf{f_i}$.

Definition 3. The *Pareto front* is then the set of all non-dominated points $P \subset O$.

The difficulty of the optimisation problem depends on the properties of S and $\mathbf{F}(\mathbf{x})$, e.g., smoothness, continuity, differentiability, and convexity. Less structure usually means more difficulty, and lack of continuity and/or differentiability often implies that one has to resort to heuristics. Especially the complexity of the model that is embedded in $\mathbf{F}(\mathbf{x})$, which computes the objective values from the decision variables, can become cumbersome for complicated mathematical models and for functions that are not convex. Additionally, the size and complexity of S determine how many times the objective function needs to be called, directly affecting the run time of the optimisation algorithm.

Considering this fact, it would thus be fruitful to reduce the complexity and size of S. This process is called *search space pruning* or just *pruning*, as it is analogous to pruning away unwanted branches from a tree or shrub [5]. Three methods to achieve pruning using machine learning can be identified [6]:

- **Reducing the dimensionality of the search space:** the dimensionality n of the unconstrained search space $S \subset \mathbb{R}^n$ has a substantial impact on the search time because the algorithm has a time complexity of $\mathcal{O}(r^n)$ if each dimension is discretised into r points. This clarifies that reducing the number of control variables wherever possible is key in reducing resource usage, for example by transforming a high-dimensional problem ($n > 2$) into a cascade of multiple two-dimensional searches (see [7] and [8]) or by discarding variables that have little to no impact on the objective function.
- **Decreasing the size of variable domains:** apart from reducing the number of variables, we can also reduce the domain of each control variable by pruning out regions that do not yield feasible solutions, which happens if either search space constraints or objective space constraints are violated. This can be done for every variable, thus compressing the entire search space. After the evaluation of a few points distributed throughout the whole initial domain, an algorithm could possibly identify clusters of feasible and promising solutions. These can then be used as initial guesses for the actual optimisation heuristic, which is then expected to converge to an optimum more quickly.
- **Providing initial search points:** the results from the domain shrinking method, as described in the previous paragraph, might be used in a learning algorithm to come up with initial search points that will allow for the optimising heuristic to converge more quickly to fit solutions and potentially a global optimum.

Putting our discussion in the context of spacecraft trajectory design problems, some additional properties are of importance. Multiple gravity assist (MGA) trajectories give rise to the presence of many local optima due to the periodic nature of the celestial bodies' motion, which is reflected in the synodic periods of two bodies [7]. This translates to the existence of multiple launch windows, some of which are better than others. From an optimisation point of view, this situation is troublesome, as local optimisation methods will often converge to local minima instead of the global optimum, and some of the search space may be left unexplored, potentially ignoring good solutions. For this reason, heuristic algorithms are often used in this context, which are inherently based on systematic searching of the search space instead of using gradient information. Therefore, pruning of the decision space has substantial impact on the speed of the heuristic algorithm.

Having laid out the structure of the optimisation problem and pruning, we will now look at the model describing the mapping $\mathbf{F}(\mathbf{x})$ between the decision and objective spaces. In our case, the mapping is the result of the astrodynamical model that is used to describe the dynamics of the spacecraft and the celestial bodies. By its very nature, this model is complex and therefore requires some assumptions and/or simplifications.

3. Hodographic Trajectory Modelling

As mentioned in the introduction, preliminary low-thrust trajectory optimisation requires the evaluation of large numbers of different trajectories. Traditional methods, based on propagation of the governing equations of motion, are computationally intensive and thus render this exploration of the search space time consuming. Instead, research has led to the development of so-called shape-based methods, which assume a certain a priori analytical representation of the flown trajectory. This mathematical function is then fed into a dynamical model (usually a formulation of the restricted two-body problem), leading to an expression for the acceleration needed to stay on this hypothetical trajectory, i.e., the thrust vector as a function of time. This is in turn integrated w.r.t. time to obtain the total ΔV necessary to fly the trajectory.

Several shape-based methods have been developed over the years. The first type, called exponential sinusoids (or *exposins*), was developed by Petropoulos and Longuski in 2004 [9] and extended by Izzo [10]. Successive papers have introduced inverse polynomials [11], spherical shaping [12], Fourier series [13, 14], pseudo-equinoctial [15], and pseudo-spectral [16] shaping.

These methods are all based on shaping of the trajectory in position space, i.e., matching the departure and arrival positions and devising a trajectory in be-

Table 1
Base functions used

Base function	$v(t)$
Constant	1
Power	t^n
Sine	$\sin(nt)$
Cosine	$\cos(nt)$
Power times sine	$t^n \cdot \sin(nt)$
Power times cosine	$t^n \cdot \cos(nt)$

tween. This trajectory describes the position at all epochs given a few parameters [17, 18].

In contrast, the hodographic method shapes the trajectory in *velocity space*. This means that the departure and arrival velocities are matched to those of the initial and target orbits, and a mathematical description of the trajectory in between is formulated. This formula yields the velocity at every point along the trajectory and can be integrated to obtain the positions at all epochs. An example of this process is given in Figure 2: the right plot shows the hodographs of the initial (blue) and target (red) orbits along with the transfer trajectory (black). It can be seen that the transfer orbit intersects the hodographs of the two orbits, implying that the velocity boundary conditions have been met. Integrating this transfer trajectory over time, we obtain the shape of the transfer in canonical (i.e., spatial) coordinates (see the left figure). Note that a perfectly circular orbit with eccentricity $e = 0$ would be represented by an infinitesimally small point in the V_r vs V_t graph, with $V_r = 0$, as the tangential velocity is constant for this case. In the more general case of an elliptical orbit, its hodograph will be a circle in velocity space [19].

Figure 3 shows the velocity hodographs for three different types of Kepler orbits (elliptical, parabolic, and hyperbolic). Indeed, all shapes are circular, whereas the offset of the centre is determined by the type of orbit. In the case of a parabola, the null velocity at $r = \infty$ is shown as the circle touching the $\dot{r}$ axis. Hence, an increasing eccentricity relates to shifting the velocity hodograph down the $r\dot{\theta}$ axis.

After having established the trajectory as a linear combination of base functions, it is substituted into the equations of motion of the two-body problem. This then yields the thrust force in every direction, which is integrated to obtain the total ΔV. For the mathematical details, the reader is referred to Appendix B.

The advantage of the hodographic method is that the velocity (Von Neumann) boundary conditions are easily met, whereas the position (Dirichlet) boundary conditions are obtained without having to resort to iterative algorithms. Furthermore, this method lends itself well to low-thrust trajectories and allows both design with and without degrees of freedom. Gondelach proposes velocity functions that are linear combinations of power, sine, cosine, and power times (co)sine functions (see Table 1) [17, 18]. He also proposes two variants of his method, making a distinction between time and polar angle as the independent variable. Both methods have their advantages, as time-based shaping offers more intuitive insight into the dynamics, whereas polar angle-based shaping is more related to the (periodic) geometry of the transfer trajectory.

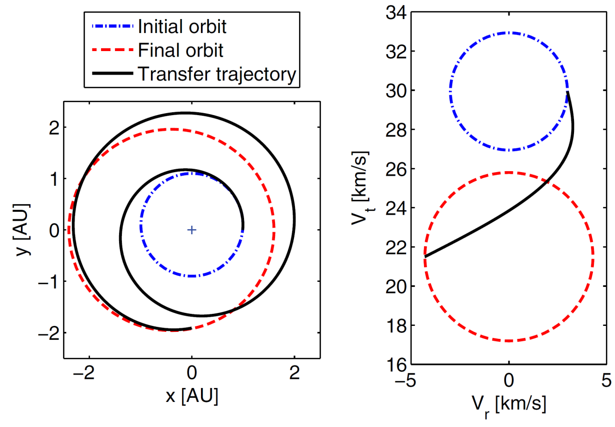

Figure 2: The figure on the left shows the trajectory in position space, whereas the figure on the right shows the same trajectory as a hodograph; taken from [18].

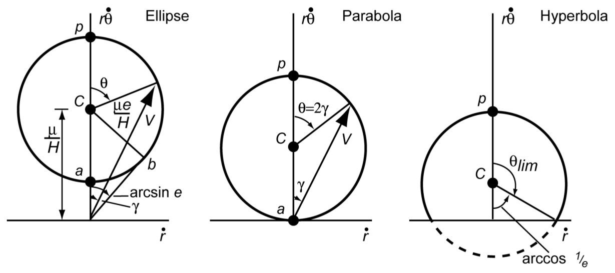

Figure 3: Illustration of the velocity hodographs for three types of Kepler orbits [19].

4. Clustering

This paper focuses on pruning by means of clustering: dividing the data set into disjoint, internally coherent groups. In our case, these clusters represent regions in the search space with feasible trajectories in terms of ΔV.

Clustering is however a very ill-defined concept and no consensus about its definition has been reached in the scientific community [22, 23]. Likewise, clustering is also problematic from a philosophical viewpoint, as the existence of 'true' or 'natural' partitionings is the topic of debate [24]. It is therefore crucial to specify the goal of the clustering, so that a 'good' clustering can be distinguished from a 'bad' one. In this paper, we want to stress the fact that there is no unique best clustering for any given case, but 'good'

Table 2
Summary of the four cluster validation techniques, their key strengths, and key weaknesses.

Validation Type	Key Strength	Key Weakness
External	Best performance in autonomous application	Requires ground truth
Internal	Requires no other clustering	Little absolute meaning
Relative	Often does not involve statistical tests [20]	Is inherently comparative in nature
Manual	Highest accuracy	Cannot be automated; only feasible when $n \leq 3$

Table 3
Comparison of the most common offline clustering types, along with examples for each category. See Table 22 in [21] for a comprehensive comparison.

Type	Examples	Advantages	Disadvantages
Centroid	**K-means**, PAM, CLARA	Efficient	Sensitive to outliers and K, drawn to local optimum; does not work well with non-flat geometry
Density	**DBSCAN**, **OPTICS**, Mean-shift	Non-flat geometry and high efficiency	Sensitive to parameters; does not work well with uneven density distributions
Distribution	DBCLASD, **GMM**		
Hierarchy	**HDBSCAN**, BIRCH, CURE, ROCK	Works with different kinds of shapes	K is a preset; high time complexity
Kernel	SVC, kernel K-means, MMC	Works well with high-dim. feature spaces	Sensitive to kernel type and parameters; high time complexity

clusterings will be regarded as such if they:

1. Match the existing periodic structure in the data (cf. synodic periods);
2. Help the optimisation algorithm that utilises the clustering stage in finding the optimal transfer trajectory.

The requirements stated above still fail to allow an objective assessment of clusterings. This inherent problem of clustering has led to the development of cluster validation, which aims to quantify the performance of a given clustering, in order to produce statements about its quality in an absolute or relative sense; it will be discussed in the next section.

4.1. Cluster Validation

Validating the clusters that arise from the clustering algorithm is arguably even more problematic than clustering itself. How do we know if the clustering is 'good' and how can we compare different clusterings?

Answering these questions is the domain of *cluster validation*. Four main validation methods are generally distinguished [20] (see also Table 2):

1. **Internal validation:** validation based on individual clusterings only. It usually deals with notions of cluster coherence (intracluster) and separation (intercluster) that are consolidated into one *metric* or *index*;
2. **External validation:** a clustering is compared to another clustering that is considered to be the ground truth partitioning. Again, a metric is used to indicate the degree of agreement between the validated and ground truth clusterings;
3. **Relative validation:** instead of comparing a clustering to a ground truth partitioning, a data set is clustered using two or more different clusterings, which are then numerically compared to provide a ranking for the different models. In this way, we can make a statement of a model *relative* to another, for that specific data set;
4. **Manual validation:** the last method is to let a human assess the clustering, relying on subjective judgement. This method is feasible when the dimensionality of the search space is low ($n \leq 3$), since visualisation of higher-dimensional spaces is very difficult.

It should be obvious that external validation is the preferable validation technique, as it compares a clustering to something we know to be 'good'. It depends, however, on a ground truth which is usually not available — we are performing unsupervised learning after all. In practice, only the three other methods remain, so we will now turn our attention to these validation techniques.

Internal validation uses information that is enclosed in the clustering to come up with a number that indicates the clustering performance, called *validation index*. Many of these indices exist, each with a different definition. Arbelaitz et al. compare 30 such indices in their paper [25], concluding that certain indices perform better in some instances than others, although no clear conclusion is drawn. It underlines the ubiquity of the 'no free lunch' principle, which states that there is no single index that outperforms all others in a statistically significant way; one can only hope to match an index to a specific problem (set).

Other than this, there is yet another problem. What does it mean when we obtain a value of say, 1.6, on a range from 0 to 3? Is it any good? The answer to this question is hard to find, because these indices carry little absolute meaning, if any. We can only make statements about the values with respect to either the extreme ends of the spectrum or by comparing the outcome for different clusterings, which leads us to relative validation.

Relative validation is based on the comparison of different clustering schemes, and the best one is selected according to some criterion [20]. This criterion can be more subjective in nature, e.g. inspection, or be one of the internal validation indices. It can be used to compare both different clustering algorithms and different sets of parameters for a given algorithm.

4.2. Clusterability

Related to the philosophical question of whether a true, unique clustering exists for a given data set, is the fact that the data generated by a grid search or other heuristic optimisation algorithm will typically not contain any obvious clusters, posing an additional challenge. When one then applies a clustering algorithm to the generated data set, it can and will find a certain number of clusters (when N_{clusters} is specified a priori), or it will just converge to a single cluster, spanning the entire data set. It should be obvious that both situations are meaningless and must thus be avoided.

We therefore use the notion of *clusterability* to describe to what degree an inherent clustering is present in the data set. Several methods of deriving a numerical representation for clusterability, i.e. clusterability metrics, have been proposed [26].

To solve the problem of a search space not being clusterable, we introduce a *thresholding* step that creates gaps between promising regions in the search space by effectively filtering out the data points that do not satisfy some kind of inequality constraint on the fitness values.

In the case of decision space pruning, clustering could identify clusters of trajectories which are often spread in a quasi-periodic way due to the repetitive motion of celestial bodies [7]. Obviously, we do not have any a priori labels for these clusters, as the clusters represent groups of trajectories that belong to a certain launch window. These regions can then be used as new bounds for subsequent optimisation runs, where the search domains can be constrained in a linear fashion (yielding rectangular boxes) or using nonlinear constraints (yielding arbitrary shapes).

More formally, we can define clustering as an operation that assigns each data point to a unique group (i.e. cluster) based on its relative proximity to other data points in the set. We can thus identify two requirements for a 'good' partitioning:

1. Data points belonging to the same cluster are in *close proximity* to each other;
2. Data points belonging to different clusters are *sufficiently far* apart.

Next to these requirements, Kleinberg identifies three desirable properties of clustering algorithms in his 2003 paper [23]. But before we can introduce them, we need to have a working definition for a clustering algorithm:

Definition 4. Given a set S with n data points, we define the distance function $d : S \times S \to \mathbb{R}$ such that $\forall i, j \in S$ the following holds: $d(i, j) \geq 0$ and $d(i, j) = 0 \iff i = j$. Furthermore, $d(i, j) = d(j, i)\ \forall (i, j)$. A partitioning Γ is then defined as $\Gamma = f(d)$, where $f(d)$ is the clustering function acting on S.

With Definition 4 in mind, Kleinberg defines the following properties:

1. *Scale-invariance:* for any distance function d and $\alpha > 0$, $f(d) = f(\alpha \cdot d)$, hence $\Gamma_d = \Gamma_{\alpha \cdot d}$ In words, we require that the partition Γ be invariant under a change of scale;
2. *Richness:* Range(f) is equal to the set of all partitions of S;
3. *Consistency:* Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transformation of d, then $f(d') = \Gamma$. See Definition 5 for the definition of a Γ-transformation.

Definition 5. A Γ-transformation d' of d is defined as $d'(i, j) \leq d(i, j) \quad \forall i, j \in C \wedge d'(i, j) \geq d(i, j) \quad \forall i \in C, j \notin C$, with $C \subset S \wedge C \subset \Gamma$.

Given these three properties, Kleinberg then postulates the following:

Theorem 1. *For each $n \geq 2$, there is no clustering function f that satisfies all three properties simultaneously. Here, n is the size of the point sets on which the clustering function f acts.*

The proof can be found in [23] and will not be repeated here. Which trade-off is made, depends on the class of the algorithm. For example, Kleinberg proves that centroid-based algorithms (such as k-means) do not satisfy the consistency property.

4.3. *Clustering Models*

There is a plethora of clustering methods, differing in performance and complexity, and, like optimisation algorithms, their efficacy depends on the distribution and nature of the data. There is no free lunch here either: a universally applicable algorithm does not exist.

Another taxonomy of clustering methods relates to the nature in which the data set is fed to the algorithm: *statically* or *dynamically*. The first refers to the more common case where all relevant data is clustered at the same time. Dynamic clustering, on the other hand, refers to the progressive clustering applied to a *data stream*, i.e., data is collected through time. With each time step, the clustering algorithm processes the new data and uses it to update the partitioning of the data set. Dynamic clustering comes with its own complications: the clustering should be stable but also flexible enough to accommodate new data points and, if necessary, increase or decrease the number of clusters.

Different (static) clustering types are compared in Table 3. This table is not exhaustive and the reader is referred to [21] for a comprehensive review of traditional and more modern clustering algorithms. Each type of clustering algorithm has its own advantages and disadvantages; therefore, the choice of which algorithm to deploy is heavily dependent on the nature of the data set and the number of data points. Some algorithms, like K-means, are highly efficient and simple in their use, but do not work well on data with a 'non-flat geometry'. The term *non-flat geometry* refers to manifolds that have a non-zero curvature[1], e.g. spirals or concentric circles. In those instances, Euclidean distances might not be the best option for clustering, and rather than looking at centroids, one should look at densities or linkage for correct clustering.

5. Software architecture

Having described the necessary background, we will now proceed to explaining the methodology for generating the trajectories, clustering, and data persistence. All simulations were run on the TU Delft Astrodynamics Toolbox (Tudat)[2] platform, a C++/Python library which has been developed by the Faculty of Aerospace Engineering at the Delft University of Technology [27]. It supports the numerical propagation of dynamical equations, variational equations, as well as optimisation. The latter is accomplished by means of an interface to the Pagmo/Pygmo 2 library.[3] It supports both global and local optimisation by providing numerous implementations of heuristic algorithms and local optimisation algorithms.

[1]Source: http://mathworld.wolfram.com/FlatManifold.html, accessed on 22/08/2019.
[2]https://tudat-space.readthedocs.io/en/latest/
[3]https://esa.github.io/pagmo2/index.html

Table 4
Used Python packages, including the version used during this thesis

Use	Package	Version
Simulation	tudat	2.9.0
...	tudatpy	0.5.16
Global optimisation	pygmo	2.16.1
Clustering & PCA	scikit-learn	0.23.2
HDBSCAN clustering	hdbscan	0.8.27
Database & ORM	sql-alchemy	1.3.23
Plotting	matplotlib	3.3.2
Data manipulation	numpy	1.19.1
Local optimisation	scipy	1.5.3

Figure 4 shows the structure of the simulation program that was developed for this research. The yellow part comprises the core modules that are responsible for clustering and pruning. These parts call the Python wrappers of the simulation and optimisation modules, here displayed in green. These libraries are implemented in C++for performance reasons, but can be accessed by the Python wrapper libraries that form the interface between the yellow and green parts. Lastly, the red part represents the cloud-run database and its access layer. The database was run on the Google Cloud Platform to ensure data persistence in case of local hardware failure. It contains the generated trajectories and the values of the parameters that describe them, the clusterings, and values for the different celestial bodies that were used in this thesis.

All simulations were carried out on an HP ZBook Studio G5 with an Intel® Core™ i7-8750H CPU with a clock frequency of 2.20 GHz and 12 logical cores, and 16 GB of memory. The resulting trajectories were stored in a cloud-run PostgreSQL database. The codebase can be found on the main author's GitHub page[4].

The Python modules also use various third-party packages (see Table 4), all available through Pip or Anaconda.

6. Exploiting synodic periodicity

Before one can apply clustering and use it to prune the search space, it is necessary to have a more thorough understanding of the structure of the latter. Does it present patterns that lend themselves to clustering in the first place? We will answer this question in this section.

We postulate here that single-leg trajectories between celestial bodies within the solar system exhibit a certain (repeating) pattern, which will form the basis for clustering. This statement is based on the observation that many transfers have multiple launch windows: corresponding groups of trajectories that are similar in propellant requirements (thus ΔV) and

[4]https://github.com/elmarputs/Thesis

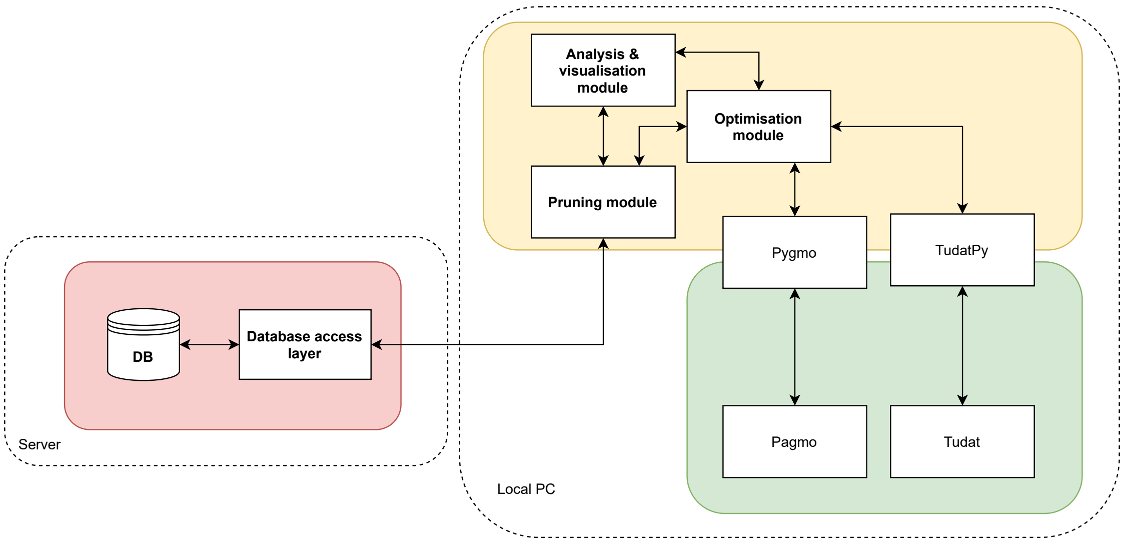

Figure 4: Overview of the software architecture. Yellow: machine learning & pruning modules; green: simulation & optimisation modules; red: cloud-run database with access layer module.

maximum acceleration. Furthermore, we will also show that these patterns arise from the synodic period, which is a function of both bodies' orbital period.

Obviously, it is impossible to test all different possible transfers between any two objects and exhaustively prove that this hypothesis holds true for all combinations. Instead, we rely on three different cases, which are discussed in the next section.

6.1. Test Cases

Three test cases were developed and implemented: two interplanetary transfers (Earth – Mars and Earth – Venus) and one comet rendezvous (Earth – 9P/Tempel 1). The Earth – Mars (EM) transfer was chosen as it is commonly used as a benchmark case in many previous studies in the field of low-thrust trajectory design [18, 28, 29]. The Earth – Venus (EV) case is considered to be representative of a transfer to an inner planet, possibly changing the dynamical nature of the problem. Both trajectories are also relevant study objects, as both Mars and Venus are commonly chosen as mission targets or fly-by candidates.

These two cases are not the most challenging, since they involve quasi-circular, i.e., low eccentricity, departure and arrival orbits,[5] possibly limiting the validity and/or applicability of any conclusions following from the results. Hence, we have decided to also include a more dynamically challenging trajectory. Comet Tempel 1 was chosen to this end, as a comet's trajectory greatly differs from the major planets' trajectories in terms of eccentricity (usually very elliptical) and inclination (often their orbits are not in the ecliptic). These two factors reduce the number of possible propellant-efficient transfers that are also feasible for low-thrust spacecraft. A reason that is more practical in nature is the availability of accurate ephemeris data in SPICE. In the remainder of this paper, we will refer to this case with the acronym ET.

All cases use the same window of 26/9/2022 — 25/9/2030, based on [17]. This window allows for the presence of four to five synodic periods for the EV and EM cases, and approximately two for the ET case. The time of flight bounds are 500 to 2000 Julian days, allowing for both relatively short and long transfers. Ephemeris data was taken from the NASA/NAIF SPICE library (Interpolated SPICE), based on the most recent orbital models (as of June 2021) of the bodies under investigation. For computational reasons, these ephemerides were sampled with an interval of 1 JD and interpolated using a 6th order Lagrange interpolator. The interpolation interval was extended by 200 Julian days on both boundaries to account for the unreliability of the interpolator at the edges of the domain. Lastly, all pork chop plots were produced using 50×50 and 25×25 grid searches of the design space.

6.2. Search space exploration results

Both the EM (Figure 5) and the EV (Figure 6) plots exhibit a strong pattern as far as short-lived transfers are concerned: 'flames' of dark, high ΔV regions are visible at a regular spacing. Similarly, separating these regions of adverse transfer conditions are the regions with relatively opportunistic ΔV values. This underlying regularity can indeed be explained by comparing the frequency of these phenomena to the synodic period of the two celestial bodies in question.

Using the simple formula $\tau = \frac{T_o}{T_i}$ to compute the

[5]In a heliocentric reference frame, that is.

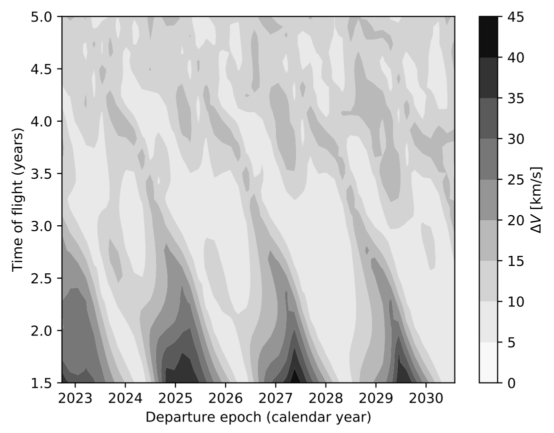

Figure 5: Porkchop plot for the Earth — Mars transfer.

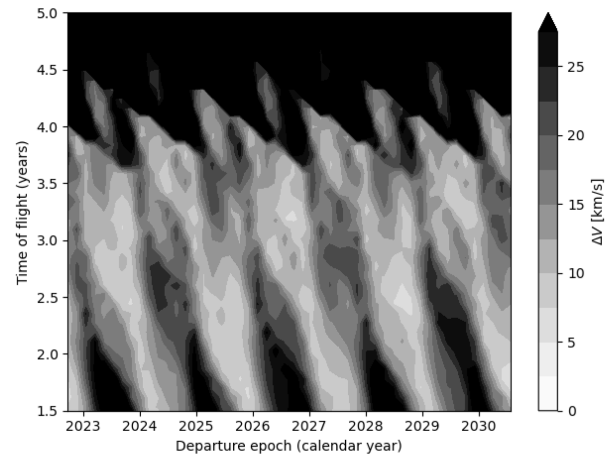

Figure 6: Porkchop plot for the Earth — Venus transfer.

synodic period τ using the inner body orbital period T_i and outer body orbital period T_o, we obtain $\tau = 1.88$ years in the EM case, whereas for EV this value equals approximately $\tau = 1.62$ years. This figure means that each τ years, both bodies return to their same initial relative position. Given the window length Δt, we can now calculate how often (f) we would expect both planets to line up:

$$f = \frac{\Delta t}{\tau} \tag{1}$$

With $\Delta t = 8$ years, we see that for the EM transfer we find $n = 4.25$ and in the EV case $n = 4.93$. We can clearly see that the frequency of the ridges and valleys in Figure 5 and Figure 6 correspond to the calculated frequencies f, backing up the statement made earlier that these regions of relatively advantageous trajectories correspond to the alignment of both bodies and thus to their synodic period.

As expected, the ET case (depicted in Figure 7) is

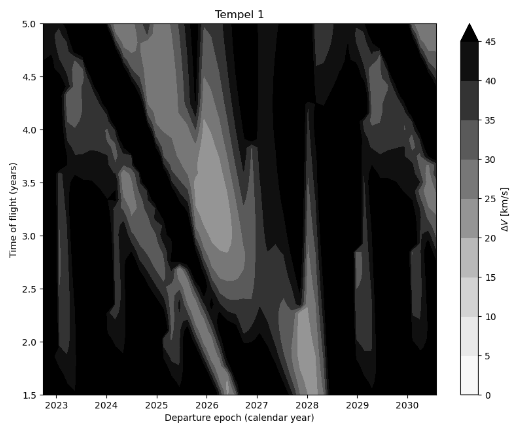

Figure 7: Porkchop plot for the Earth — 9P/Tempel 1 transfer.

different. Since Tempel 1's period is much larger than Earth's (approximately 5.58 years), we observe that only two synodic periods are captured in this specific window. Furthermore, two different patterns can be distinguished: a broad, lower ΔVregion, and a narrow region containing trajectories with higher ΔVvalues. Next to these oblique regions, that are similar to those in the EV and EM cases, we can also observe vertical 'spikes' emerging from the main bands.

7. Data preprocessing

Before clustering can be applied, it is crucial to format the dataset such that the clustering model can effectively partition the data into coherent, well-separated sets. This step is commonly known as *data preprocessing* and entails filtering out noise, scaling values, transforming to a different basis, etc.

7.1. Noise in clustering

In conventional data sets used in machine learning, data is not artificially generated but rather obtained from surveys, existing databases, and other sources. Usually, these include data points that do not necessarily belong to a group and are considered *noise*. Most clustering algorithms can, however, not properly deal with the presence of noise, which will result in distorted clusters, or even very poor clustering in general [30]. Only a few algorithms such as (H)DBSCAN and OPTICS have a notion of noise, where noisy data points are put in a separate 'cluster'.

7.2. Scaling

The second type of preprocessing is the use of scaling. By scaling, we mean operations such as normalising and standardising: transforming the range of values to another interval. In the case of normalising, we give all data points a unit norm, according to some norm (l_1, l_2, or l_∞). Standardising, instead, refers to nullifying the data set mean and enforcing unit variance.

7.3. Basis Transformation

The third important data transformation in this research is rotation, or, equivalently, changing coordinate basis. As we will see in Section 9.1, this is paramount for some clustering algorithms, which depend on projections of data onto an axis. As such, these models will perform poorly unless the data set is transformed to an appropriate basis.

7.4. Making Pork chops Clusterable

Lastly, we need to deal with a problem that is specific to our case: the datasets generated by the grid search contain points spaced at regular intervals, and every point in the mesh is covered. As a result, most, if not all, clustering algorithms will fail to come up with a clustering that meets our expectations. For example, density-based models will fail because of the lack of difference in density among the points. These models need a clear distinction between high-density clusters and low-density intercluster space. Other algorithms also produce poor results since equally-spaced points lack the separation between clusters; most algorithms will thus allocate all points to a single cluster or to clusters that do not make sense.

Given these poor results, we therefore propose the application of a *threshold* to the data set, thus forming clusters in otherwise unclusterable data. This threshold is implemented as a simple constraint, such as “include all data points for which $\Delta V \leq k$”, where k is some constant. In other words, we filter our data set by putting constraints on one or more features – in our case objectives. We also observe that this threshold is a double-edged sword: by throwing away poor performing trajectories with high ΔV values, we are already pruning the search space. We are left with the ‘valleys’ of reasonably efficient trajectories, that will be grouped by the subsequent clustering stage.

The introduction of a threshold does pose some additional problems, however. A question that arises naturally is one that also applies to other hyperparameters: which value should we choose? Of course, the answer is not easily found and one needs to treat the threshold value as another hyperparameter.

8. Cluster validation using indices

As mentioned in Section 4.1, cluster evaluation is an essential part of clustering and is performed using one of the four methods mentioned earlier. Several dozens of different validity indices exist to perform this job, rendering it impossible to thoroughly test all different combinations of indices, clustering algorithms, and data sets. As such, choices have to be made to confine the number of experiments that need to be run.

In this section, we will direct our focus to three different clustering validity indices (CVIs): the Silhouette, Davies-Bouldin, and Calinski-Harabasz indices, from here onwards referred to as Sil, DB, and CH, respectively. This choice was based on the results from the extensive comparative study of internal validation indices by Arbelaitz et al. [25]. The three CVIs used here consistently performed best across the many tests that were run. Factors such as clustering algorithm, number of clusters, dimensionality, data density, cluster overlap, and noise were all varied to compare the performance (in terms of success rate) of the 30 indices under investigation. This indicates that these three indices are most robust under changing circumstances and that they might prove to be best suitable to tackle internal validation.

However, one must remain wary of attaching too strong a conclusion to this result. As with many algorithms and statistics, there is usually no single one which outperforms the rest. This is generally known as the No Free Lunch Theorem (NFLT); see also [31] and [32] for applications in the field of (global) optimisation. With this theorem in mind, we will see that the CVIs selected will not be the holy grail. In the worst case scenario, none will be helpful in determining whether a certain clustering is any good.

8.1. Judging the performance of CVIs

Ideally, one would use a CVI to find some kind of ranking of clusterings. But to see if a certain CVI is suitable for this purpose, we need to evaluate the CVI itself, given some data sets that resemble the structure that will be encountered in future applications. To that end, we compare different clusterings of the same data set — generated by the grid search as described in Section 5 — using the three CVIs. As a benchmark, we will judge the outcome on the basis of manual inspection. The justification for this approach is given by the fact that the patterns observed in Section 6 are relatively obvious; as humans, we can quickly observe the ‘correct’ clustering that groups favourable trajectories. Of course, this method has a drawback: it relies on human, subjective judgement and not on quantitative assessment. However, in our case, we merely want to see if the CVIs actually correlate to our understanding of ‘good’ clusterings; i.e., whether they behave as some kind of *proxy* for our judgement.

8.2. CVI comparison results

The results for the CVI comparison regarding different grid resolutions are presented in Figure 8 and Figure 9, for Sil/DB and CH, respectively. Note that for the Sil and CH indices, higher values correspond to better clusterings, whereas the DB index works the opposite way, meaning lower values are preferable. With this information, we notice that in most cases, the grid resolution has only a minimal effect on the index-measured performance. Only for the HDBSCAN and OPTICS models, the difference is much more pronounced and we see that here, a higher grid resolution is beneficial. In the CH case, however, the situation is completely different. Grid resolution seems to be much more influential, where higher values in all but one case yield better index values. DBSCAN is the exception here, as little difference is observed between 25 and 50 grid points.

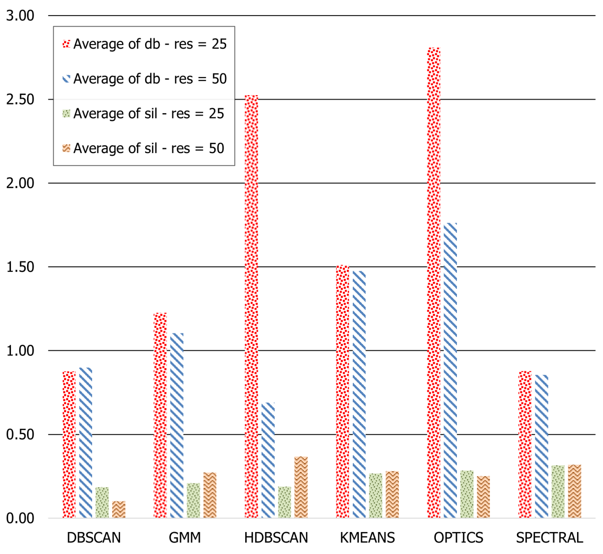

Figure 8: Comparison of six different clustering models for different grid resolutions using the Sil and DB indices. Note that higher values for Sil correspond to better clusterings; the opposite holds for DB.

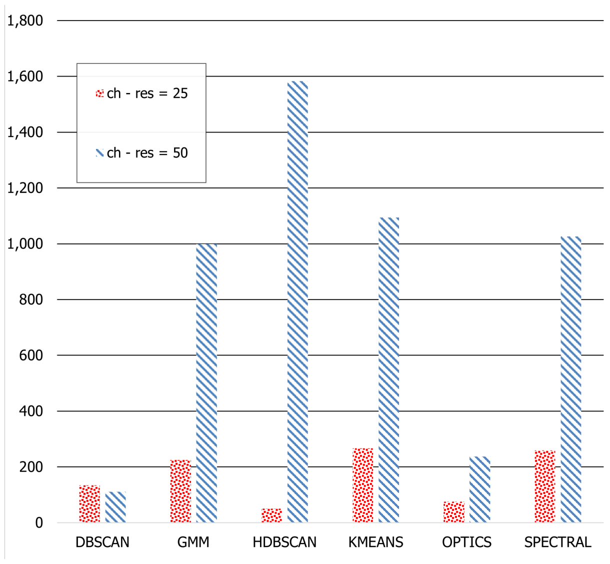

Figure 9: Comparison of six different clustering models for different target bodies using the CH index. Note that higher values for CH correspond to better clusterings.

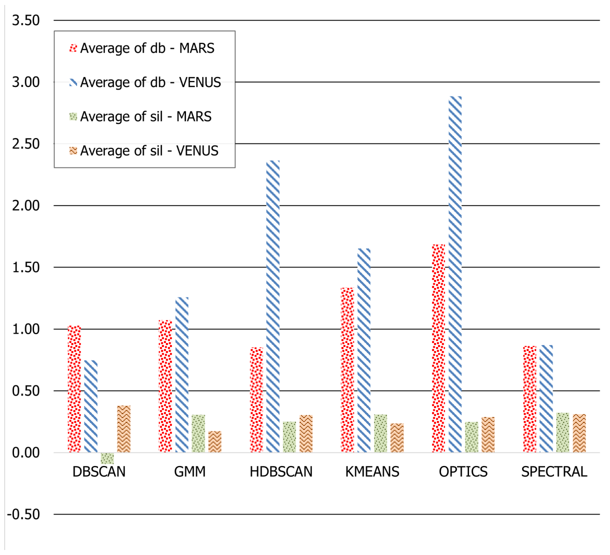

Figure 10: Comparison of six different clustering models for different target bodies using the Sil and DB indices. Note that higher values for Sil correspond to better clusterings; the opposite holds for DB.

OPTICS and HDBSCAN are also much more sensitive to a change in target body relative to the Sil index than the other algorithms, as can be seen in Figure 10. The clustering algorithms seem to perform better in the Mars transfer. Again, the difference with the CH index is striking (Figure 11). We see here that for all models, except DBSCAN, the Mars transfer yields much better clustering results than for the Venus case.

8.3. *CVI comparison discussion*

From the results shown in the previous paragraphs, an interesting observation can be made: DBSCAN seems to behave in an opposite manner with respect to the other clustering algorithms. It yields better scores when presented with a lower grid resolution and when trajectories to Venus are clustered.

Another important observation is that the CVIs are ambiguous in various situations. An illustrative example of this is seen when we look at Figures 8 and 10. We see that HDBSCAN is the best or second best algorithm for this data set when assessed with CH, but that the DB index yields a completely different story — here, HDBSCAN is among the worst. An example of such a clustering is shown in Figure 12, which is representative for all clusterings using this algorithm for EM transfers. When compared to the other algorithms (Figures 13 to 15), we see that it is closest to the desired clustering, together with DBSCAN. This is remarkable, since the CVI plots consistently rank DBSCAN much lower or higher than HBSCAN, even though the clusterings are very similar to those produced by HDBSCAN.

We also see that the Gaussian Mixture Model and

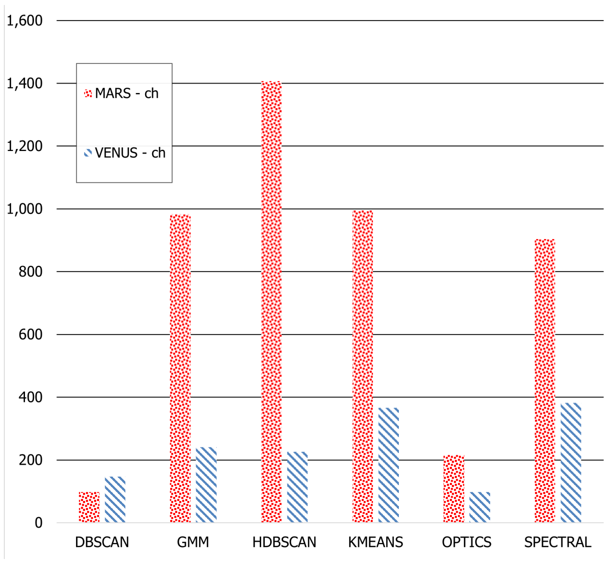

Figure 11: Comparison of six different clustering models for different target bodies using the Sil and DB indices. Note that higher values for Sil correspond to better clusterings; the opposite holds for DB.

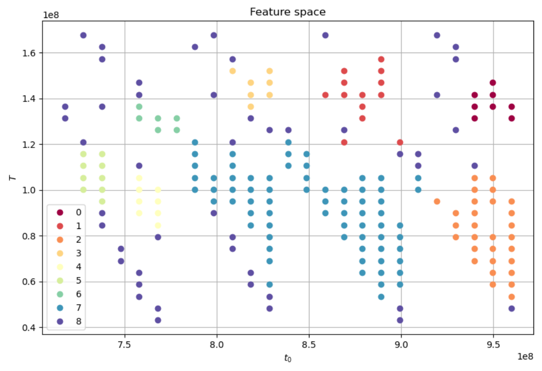

Figure 12: HDBSCAN, EM trajectory

KMeans are not able to produce any useful clusters, since they are based on centroids and thus group points that are within some range of these centroids. They are unable to deal with the elongated clusters that are encountered here.

From these results we have to conclude that using these CVIs to judge the performance of the tested clustering algorithms is not advisable. The rankings are inconsistent and very much dependent on which CVI is chosen; choosing the wrong index can lead to a ranking that is almost the opposite of the ranking done by a human. With wrong clusters, it is very unlikely that the optimisation algorithm will converge more quickly to the optimum, since the new search space identified by the clusters *still* include parts that are infeasible and/or uninteresting.

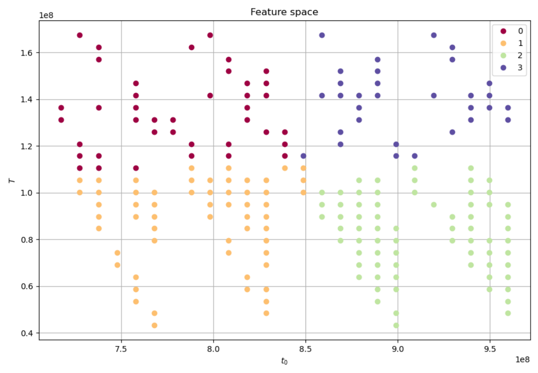

Figure 13: KMeans, EM trajectory

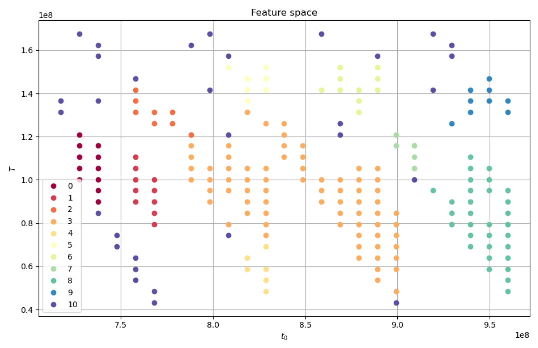

Figure 14: DBSCAN, EM trajectory

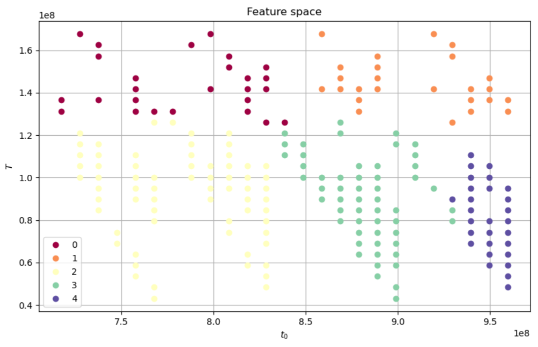

Figure 15: Gaussian Mixture Model, EM trajectory

9. Multimodality-based clustering

The clustering algorithms described in Section 4 and used in the previous steps are all based on notions of density (DBSCAN, OPTICS), centroids (KMeans), or statistical distributions (GMM, SPECTRAL). Additionally, CVIs were used to compare the models and tune the algorithms. As we have seen, though, this method does not prove fruitful for our case, as inconsistent and contradictory results were obtained. For this reason, we propose the use of modality-based clustering algorithms.

Multimodality is the existence of multiple 'peaks' in a univariate statistical distribution. A distribution having only a single peak, such as the normal distribution, is therefore called unimodal. Clusters can also be seen as 'peaks' of the distribution of some random variable. Since clusters can be defined on $\mathbb{R}$ or higher-dimensional spaces, we are mostly dealing with multivariate distributions. The clusters then correspond to modes of this distribution.

An example of this feature is shown in Figure 16. The figure depicts a typical trajectory optimisation data set that is obtained using a grid search and filtered using a threshold. We can observe the elongated clusters that form primarily in one direction, which is approximately aligned with the y-axis here. Both axes are accompanied by a 30-bin histogram, showing the frequency of data points for that axis. The x-axis histogram shows clear, well-separated peaks, which can be thought of as different modes of the underlying empirical distribution. The figure thus shows that the peaks correspond to the locations of the clusters and that we can extract the x-axis cluster intervals from these peaks.

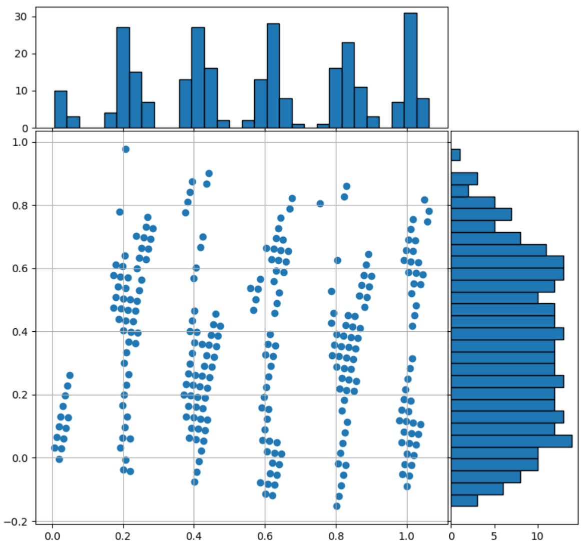

Figure 16: An example of a rotated trajectory data set with x-axis and y-axis histograms depicting the distribution of data points.

How can we use this observation to perform clustering? The answer lies in a statistical test, proposed by Hartigan and Hartigan in their 1985 paper, called the dip test [33]. This method tests the null hypothesis that the distribution is unimodal, by looking at the maximum difference between the empirical cumulative distribution function (ecdf) and the unimodal distribution that minimises this maximum difference. This maximum difference is aptly called the *dip*, hence the name of the statistical test.

The test merely rejects or accepts the null hypothesis, so it should be applied in a recursive manner to obtain all modes in a multimodal distribution. Also, Hartigan and Hartigan's work has one significant shortcoming: the test only works for univariate distributions, and is thus inherently unsuitable for clustering in its 1985 incarnation. However, this problem was solved by Maurus and Plant by employing the dip test in a recursive algorithm, called SkinnyDip, where the test is carried out in multiple dimensions to find clusters [30]. The algorithm uses recursion to find all peaks along one axis and to extract clusters in higher dimensions.

Figure 17 shows the different steps that are taken in the SkinnyDip algorithm. Because the dip-test relies on peaks in the empirical distribution function, it is first crucial to determine the appropriate basis vector that is perpendicular to the general direction of the clusters. If such a basis is found, the data is normalised and rotated, so that it is aligned with the chosen basis. Then, the data is projected onto the basis vector, producing a new, univariate distribution of the data along that axis. The data can now be processed by the dip-test, as it requires univariate data and cannot work in higher dimensions.

9.1. *Finding the Appropriate Basis*

Figure 16 also makes the importance of the choice of basis vectors clear: an inappropriate basis will yield histograms that show no distinct peaks, even though clusters are present in the data set. It is therefore essential to find a good basis that will maximise the likelihood of finding the clusters.

In this paper, we use an effective but simple method to find this angle (see Line 1). By rotating the data set between 0 and π radians and evaluating the dip at regular intervals, we can find the angle at which the dip is maximised. Since the clustering takes place in 2D in this work, this method is sufficient to quickly find the global optimum angle.

9.2. *SkinnyDip clustering results*

The results of the modality-based clustering for the different transfers can be seen in Figures 18 to 20. ΔV threshold values of 10, 15, and 35 km/s were used in the EM, EV, and ET cases, respectively, clipping out high ΔV trajectories and creating 'islands' of trajectories which can be clustered. We see that SkinnyDip consistently produces coherent clusters that each correspond to a distinct 'valley' of low ΔV trajectories. Note that the data sets in the figures are rotated back into their original orientation, although the dip-test takes place in the rotated frame as depicted in Figure 16, so that a single projection onto the x-axis creates an empirical distribution function that can be

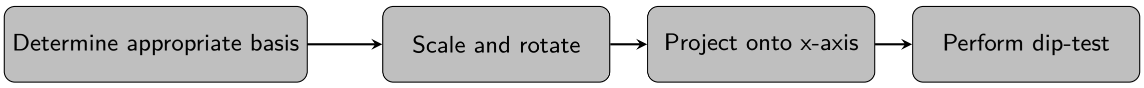

Figure 17: Flowchart of the Dip-based clustering

Algorithm 1: Finding the basis maximising the dip

```
Data: x
Result: Angle $\theta$ at which maximum dip is
        found and maximum dip
angles = range$(0, \pi)$;
dips := [];
for $\theta$ in angles do
    Construct basis vector: $\mathbf{e} := [\cos\theta, \sin\theta]$;
    Project data onto basis vector: $\mathbf{x} := \mathbf{x} \cdot \mathbf{e}$;
    Sort data: $\mathbf{x} :=$ sort($\mathbf{x}$);
    dip = diptest(x);
    dips.append(dip);
end
return max(dips);
return angles.at(max(dips))
```

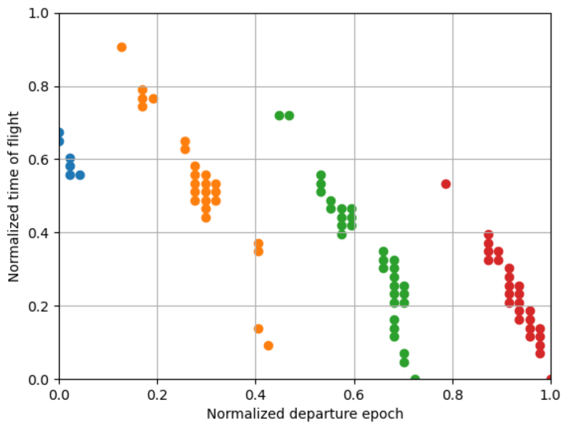

Figure 18: SkinnyDip clustering for the Earth – Mars transfer.

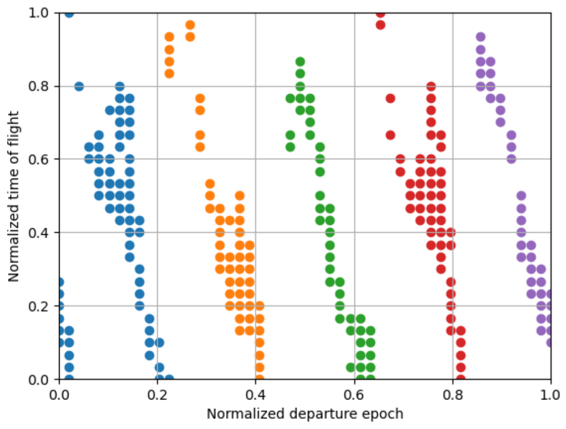

Figure 19: SkinnyDip clustering for the Earth – Venus transfer.

tested for unimodality.

It is clear that, with a proper threshold, SkinnyDip can efficiently group trajectories that form the elongated clusters, corresponding to the distinct 'valleys' encountered in the porkchop plots as seen in Section 6. In contrast, the more conventional clustering methods consistently failed to generate clusters that were internally coherent and well-separated.

Figure 20 also includes the objective values corresponding to transfers in each cluster that was found by SkinnyDip, along with the local Pareto fronts. This figure clearly illustrates that different clusters can produce similar Pareto fronts (as in the case of the blue and green clusters). Even though the departure epoch and time of flight values of these two groups differ greatly, the fitness of the best trajectories seems to overlap to some degree. This means that clustering can also be used to find trajectories in different launch windows with similar ΔV and maximum acceleration values, allowing for more robust mission planning.

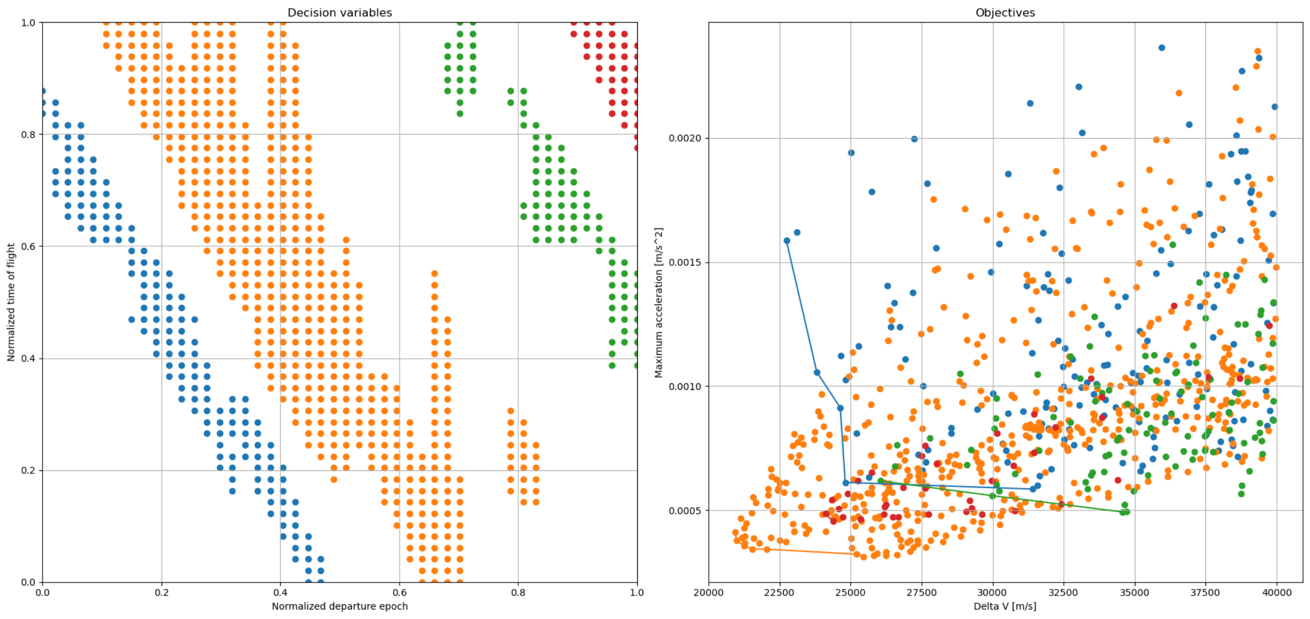

Figure 20: Left: SkinnyDip clustering for the Earth – 9P/Tempel 1 transfer. Right: Objective values (ΔV and maximum acceleration) and Pareto fronts for each cluster.

10. Conclusions

In this paper, we have explored the use of clustering for pruning the search space of single leg trajectory optimisation problems. All three transfer problems studied (EM, EV, and ET) exhibited a periodic structure that was coupled to the synodic period of the departure and target bodies. Since the data points were generated artificially, clustering can only be carried out after the data set has been filtered. To this end, we propose a simple but effective ΔVthreshold value. This method extracts the 'valleys' containing the favourable trajectories, which are then clustered. The clusters can provide knowledge in the form of new search region bounds, that in turn function as initialisation intervals for multi-start/parallel optimisation algorithms.

(H)DBSCAN, OPTICS, KMeans, and GMM failed to consistently and robustly provide clusterings that were satisfactory for pruning purposes, since the clusters are oblong and noise–cluster density differences are nonexistent due to the artificial nature of the data sets. A solution to this problem was found in the use of multimodality-based clustering, in the form of SkinnyDip. This clustering method uses the dip-test, which is a statistical test to assess whether some statistical distribution is unimodal or not. Due to its nature, it lends itself for long, narrow clusters that we encounter when optimising interplanetary transfers. SkinnyDip also has the advantage of requiring very little hyperparameter tuning. This stands in stark contrast to the other clustering algorithms, as they usually require extensive tuning before being able to produce useful results.

In every clustering application, it is crucial to evaluate the partition of the data. In this paper, we have seen that internal cluster validation indices are inadequate for our problem, since their values are inconsistent with human judgement of the quality of the clusterings. Unless some ad-hoc index is found that suits the clusters encountered in trajectory optimisation, the use of CVIs is discouraged, as they might lead to wrong conclusions about the quality of clusterings in the absence of human judgement.

References

[1] J. T. Betts, Survey of Numerical Methods for Trajectory Optimization, Journal of Guidance, Control, and Dynamics 21 (1998) 193–207. doi:10.2514/2.4231.

[2] A. V. Rao, A Survey of Numerical Methods for Optimal Control, Advances in the Astronautical Sciences 135 (2009) 497–528.

[3] S. Bandaru, A. H. C. Ng, K. Deb, Data Mining Methods for Knowledge Discovery in Multi-Objective Optimization: Part a - Survey, Expert Systems with Applications 70 (2017) 139–159. doi:https://doi.org/10.1016/j.eswa.2016.10.015.

[4] K. Deb, Multi-objective Optimization, Springer US, Boston, MA, 2014, pp. 403–449. doi:10.1007/978-1-4614-6940-7_15.

[5] D. Izzo, Global optimization and space pruning for spacecraft trajectory design, Cambridge University Press, 2010, pp. 178–201. doi:10.1017/CBO9780511778025.008.

[6] R. Liu, A. Agrawal, W. Liao, A. Choudhary, Search Space Preprocessing in Solving Complex Optimization Problems, in: 2014 IEEE International Conference on Big Data (Big Data), 2014, pp. 1–5. doi:10.1109/BigData.2014.7154118.

[7] D. R. Myatt, V. M. Becerra, S. J. Nasuto, J. M. Bishop, Advanced Global Optimisation Tools for Mission Analysis and Design, Report 03-4101a, European Space Agency, the Advanced Concepts Team, 2004.

[8] D. Izzo, V. M. Becerra, D. R. Myatt, S. J. Nasuto, J. M. Bishop, Search Space Pruning and Global Optimisation of Multiple Gravity Assist Spacecraft Trajectories, Journal of Global Optimization 38 (2007) 283–296. doi:10.1007/s10898-006-9106-0.

[9] A. E. Petropoulos, J. M. Longuski, Shape-Based Algorithm for Automated Design of Low-Thrust, Gravity-Assist Trajectories, Journal of Spacecraft and Rockets 41 (2004) 787–796. doi:10.2514/1.13095.

[10] D. Izzo, Lambert's Problem for Exponential Sinusoids, Journal of Guidance, Control, and Dynamics 29 (2006) 1242–1245. doi:10.2514/1.21796.

[11] B. J. Wall, B. A. Conway, Shape-Based Approach to Low-Thrust Rendezvous Trajectory Design, Journal of Guidance, Control, and Dynamics 32 (2009) 95–101. doi:10.2514/1.36848.

[12] D. M. Novak, M. Vasile, Improved Shaping Approach to the Preliminary Design of Low-Thrust Trajectories, Journal of Guidance, Control, and Dynamics 34 (2011) 128–147. doi:10.2514/1.50434.

[13] E. Taheri, O. Abdelkhalik, Shape-Based Approximation of Constrained Low-Thrust Space Trajectories Using Fourier Series, Journal of Spacecraft and Rockets 49 (2012) 535–545. doi:10.2514/1.A32099.

[14] E. Taheri, O. Abdelkhalik, Initial Three-Dimensional Low-Thrust Trajectory Design, Advances in Space Research 57 (2016) 889–903. doi:10.1016/j.asr.2015.11.034.

[15] P. De Pascale, M. Vasile, Preliminary Design of Low-Thrust Multiple Gravity-Assist Trajectories, Journal of Spacecraft and Rockets 43 (2006) 1065–1076. doi:10.2514/1.19646.

[16] B. De Vogeleer, Automatic and Fast Generation of Sub-Optimal and Feasible Low-Thrust Trajectories Using a Boundary-Value Pseudo-Spectral Method, Master's thesis, Delft University of Technology, 2008.

[17] D. Gondelach, A Hodographic Shaping Method for Low-Thrust Trajectory Design, Master's thesis, Delft University of Technology, 2012.

[18] D. J. Gondelach, R. Noomen, Hodographic-Shaping Method for Low-Thrust Interplanetary Trajectory Design, Journal of Spacecraft and Rockets 52 (2015) 728–738. doi:10.2514/1.A32991.

[19] K. F. Wakker, Fundamentals of Astrodynamics, Institutional Repository Library, Delft University of Technology, Delft, 2015.

[20] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On Clustering Validation Techniques, Journal of Intelligent Information Systems 17 (2001) 107–145. doi:10.1023/a:1012801612483.

[21] D. Xu, Y. Tian, A Comprehensive Survey of Clustering Algorithms, Annals of Data Science 2 (2015) 165–193. doi:10.1007/s40745-015-0040-1.

[22] M. Ackerman, S. Ben-David, Measures of clustering quality: Aworking set of axioms for clustering, Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference (2009) 121–128.

[23] J. Kleinberg, An impossibility theorem for clustering, in: Advances in Neural Information Processing Systems, Neural information processing systems foundation, 2003.

[24] C. Hennig, What are the true clusters?, Pattern Recognition Letters 64 (2015) 53–62. doi:10.1016/j.patrec.2015.04.009. arXiv:1502.02555.

[25] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, I. Perona, An extensive comparative study of cluster validity indices, Pattern Recognition 46 (2013) 243–256. doi:10.1016/j.patcog.2012.07.021.

[26] A. Adolfsson, M. Ackerman, N. C. Brownstein, To cluster, or not to cluster: An analysis of clusterability methods, Pattern Recognition 88 (2019) 13–26. doi:10.1016/j.patcog.2018.10.026. arXiv:1808.08317.

[27] K. Kumar, P. van Barneveld, D. Dirkx, J. Melman, E. Mooij, R. Noomen, Tudat: a modular and robust astrodynamics toolbox, in: A. Benoit (Ed.), Conference proceeding of the 5th ICATT Conference, ESA, 2012, pp. 1–8. 5th ICATT Conference ; Conference date: 29-05-2012 Through 01-06-2012.

[28] L. Bouwman, Gaussian Process Models for Preliminary Low-Thrust Trajectory Optimization, Master's thesis, Delft University of Technology, 2019.

[29] D. Izzo, Advances in Global Optimisation for Space Trajectory Design, in: Proceedings of the international symposium on space technology and science, volume 25, 2006, p. 563.

[30] S. Maurus, C. Plant, Skinny-dip: Clustering in a sea of noise, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 13-17-Augu (2016) 1055–1064. doi:10.1145/2939672.2939740.

[31] T. Joyce, J. M. Herrmann, A Review of No Free Lunch Theorems, and Their Implications for Metaheuristic Optimisation, Springer International Publishing, Cham, 2018, pp. 27–51. doi:10.1007/978-3-319-67669-2_2.

[32] D. H. Wolpert, W. G. Macready, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82. doi:10.1109/4235.585893.

[33] J. Hartigan, P. Hartigan, The dip test of unimodality, Annals of Statistics 13 (1985) 70–84.

[34] T. Calinski, J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics 3 (1974) 1–27.

[35] D. Davies, D. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1 (1979) 224–227. doi:10.1109/TPAMI.1979.4766909.

[36] P. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65. doi:10.1016/0377-0427(87)90125-7.

A. CVI definitions

- Calinski-Harabasz index (CH) [34]:

$$\mathrm{CH}(C) = \frac{N-K}{K-1} \frac{\sum_{c_k \in C} |c_k| \, \mathrm{d_e}\left(\overline{c_k}, \overline{X}\right)}{\sum_{c_k \in C} \sum_{x_i \in c_k} \mathrm{d_e}\left(x_i, \overline{c_k}\right)} \tag{2}$$

- Davies-Bouldin index (DB) [35]:

$$\mathrm{DB}(C) = \frac{1}{K} \sum_{c_k \in C} \max_{c_l \in C \setminus c_k} \left\{ \frac{\mathrm{S}(c_k) + \mathrm{S}(c_l)}{\mathrm{d_e}\left(\overline{c_k}, \overline{c_l}\right)} \right\} \tag{3}$$

- Silhouette index (Sil) [36]:

$$\mathrm{Sil}(C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{\mathrm{b}(x_i, c_k) - \mathrm{a}(x_i, c_k)}{\max\{\mathrm{a}(x_i, c_k), \mathrm{b}(x_i, c_k)\}} \tag{4}$$

where:

$$\mathrm{a}(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} \mathrm{d_e}(x_i, x_j) \tag{5}$$

and

$$\mathrm{b}(x_i, c_k) = \min_{c_l \in C \setminus c_k} \left\{ 1/|c_l| \sum_{x_j \in c_l} \mathrm{d_e}(x_i, x_j) \right\} \tag{6}$$

B. Hodographic Shaping Equations

B.1. *Base functions*

The hodographic shaping method fabricates a trajectory in velocity space. This implies that we have to come up with a function that describes the velocity in each direction as a function of the independent variable, being time t or polar angle θ. One could select some random function, but it is actually much more useful if we somehow select functions with more structure.

Finding this structure is not difficult, knowing that we must integrate the velocity functions in order to find the position as function of time or polar angle. It is hence useful to select functions that are analytically integrable, i.e., we can readily find their primitive. Secondly, the definite integral is a linear operator, so choosing a linear combination of integrable base functions seems useful. The general description of some velocity function $V(t)$ is shown in Equation (7).

$$V_j(t) = \sum_{i=1}^{n} c_i v_i(t) \quad \text{with} \quad j = \{r, \theta, z\}. \tag{7}$$

Here, $V_j(t)$ is the velocity function for one of the three base directions, n the number of base functions, c_i the i^th coefficient, and $v_i(t)$ the i^th velocity base function. The specific combination of base functions used in this thesis is shown in Equation (8) and was based upon the recommendations given by Gondelach himself [17].

$$\begin{aligned}
r(t) &= c_1 + c_2 \cdot t + c_3 \cdot t^2 + c_4 \cdot t \cos(f) \\
\theta(t) &= c_5 + c_6 \cdot t + c_7 \cdot t^2 + c_8 \cdot t \sin(f) \\
&\quad + c_9 \cdot t \cos(f) \\
z(t) &= c_{10} \cdot \cos(f(N+0.5)) + c_{11} \cdot t^3 \sin(f(N+0.5)) \\
&\quad + c_{12} \cdot t^3 \cos(f(N+0.5))
\end{aligned} \tag{8}$$

In these equations, $f = \frac{2\pi}{t_f - t_i}$ and N is the number of revolutions.

B.2. *Mathematical description of the hodographic shaping method*

To obtain the position as a function of the independent variable (here time t is considered), we integrate the three velocity functions with respect to τ, which is a dummy variable for t:

$$\begin{aligned}
r(t) &= r_0 + \int_0^t V_r \, \mathrm{d}\tau \\
\theta(t) &= \theta_0 + \int_0^t \frac{V_\theta}{r} \, \mathrm{d}\tau \\
z(t) &= z_0 + \int_0^t V_z \, \mathrm{d}\tau
\end{aligned} \tag{9}$$

Notice that the equation for the polar angle θ requires a factor $\frac{1}{r}$, which is the Jacobian for this set of coordinates. To find the values of the coefficients in front of the base functions, we need the boundary conditions. The trivial ones are:

$$V_r(0) = V_{r,0} \qquad V_r(t_f) = V_{r,t_f} \tag{10}$$

$$V_\theta(0) = V_{\theta,0} \qquad V_\theta(t_f) = V_{\theta,t_f} \tag{11}$$

$$V_z(0) = V_{z,0} \qquad V_z(t_f) = V_{z,t_f} \tag{12}$$

Those are not the only boundary conditions that we can find, however. We also happen to know the boundary conditions in position space, i.e., initial and final r, θ, and z, because the trajectory needs to coincide with the position of the departure and target body at $t = 0$ and $t = t_f$, respectively. Hence, we can rewrite Equation (9) as follows:

$$\begin{aligned}\int_0^{t_f} V_r \,\mathrm{d}\tau &= r_f - r_0 \\ \int_0^{t_f} \frac{V_\theta}{r} \,\mathrm{d}\tau &= \theta_f - \theta_0 \\ \int_0^{t_f} V_z \,\mathrm{d}\tau &= z_f - z_0\end{aligned} \tag{13}$$

Since this procedure allows determining three coefficients per direction, any extra base functions' coefficients need to be determined by an optimisation method. In general, it holds that having more base functions implies more accuracy, although this is not necessarily the case.

Having obtained the velocity functions with their coefficients, we can now proceed to calculating the thrust force that is necessary to fly this trajectory. To do this, we take the equations of motion in polar coordinates:

$$\begin{aligned}\ddot{r} - r\dot{\theta}^2 + \frac{\mu}{d^3} r &= f_r \\ r\ddot{\theta} + 2\dot{r}\dot{\theta} &= f_\theta \\ \ddot{z} + \frac{\mu}{d^3} z &= f_z\end{aligned} \tag{14}$$

where $d = \sqrt{r^2 + z^2}$. These equations hold under the following assumptions:

- Only the Sun exerts its gravitational force upon the spacecraft (other celestial bodies' gravitational pull is ignored);
- The only other force acting on the spacecraft is the thrust force $\mathbf{f}$ itself;
- The spacecraft has negligible mass w.r.t. the Sun's mass (restricted two body problem).

Since the specific thrust force is equal to the thrust acceleration, a simple integration of the thrust acceleration w.r.t. time yields the ΔV:

$$\Delta \mathrm{V} = \int_0^{t_f} \|\mathbf{f}\| \,\mathrm{d}t \tag{15}$$

where $\mathbf{f} = \begin{bmatrix} f_r \\ f_\theta \\ f_z \end{bmatrix}$.

3 Recommendations

Since this work is still a very exploratory endeavour into the application of clustering on global trajectory optimisation problems and given the conclusions that were discussed in the preceding section, we think there are still plenty of challenges that can be tackled in future studies. We have identified the following points that are likely to be the next steps in applying clustering to trajectory optimisation:

1. **Investigate how clustering affects the performance of heuristic optimisation algorithms and grid searches.** In this research, we have not looked into assessing whether significant improvements in optimisation performance occur after clustering. It is expected that pruning by means of clustering can improve the converge speed of these algorithms, since uninteresting areas are cut out and therefore do not impede the search for the global minimum.

2. **Implement a clustering-aided heuristic optimiser.** Instead of relying on clustering *before* optimisation, we expect greater merit from a dynamic clustering lodged inside the optimisation loop. In this case, clustering can be much more adaptive to new solutions generated by the heuristic algorithm, although this 'data stream' based clustering requires more tuning. Cluster stiffness (how eager clusters change) is an example of an extra hyperparameter that is introduced in this process, and needs to be tuned.

3. **Use clustering for a recursive 'zooming' optimisation algorithm.** Since clustering can be used to partition the population into distinct groups, consisting of potentially promising trajectories, these groups can each be processed in a separate optimisation loop. For every group, clustering is applied on a smaller, intra-group scale, such that 'zooming' functionality is obtained. As zoom depth is increased, it is expected that the algorithm comes nearer to the true optimum, while total computation time is increased. More research is necessary to quantify the effect of this functionality, so that a trade-off can be made between computation time and optimisation accuracy on a numerical basis;

4. **Further investigate the potential role and merit of CVIs in trajectory clustering.** In this research, we have concluded that CVIs are unreliable when it comes to tuning and comparing different clustering models, as the nature of the clusters encountered in trajectory optimisation seems to be unsuitable for these CVIs. As the NFLT dictates, measures that perform very well on some problems might perform poorly in other problems. As such, a different CVI — already in existence or still to be created — can be much more suitable to the nature of the clusters encountered in our problem, although this is rather ad-hoc and does not solve the underlying philosophical problem. In this regard, SkinnyDip seems to be much more consistent in its clusterings and might thus be a more promising path to take;

5. **Use clustering-aided pruning as a tool for multimodal optimisation.** Trajectory optimisation problems often present multiple regions with local optima, also called multimodal optimisation. A classical example of a function exhibiting multimodality is the Weierstrass function, which is a function contrived by Karl Weierstrass and often used to assess the performance of heuristic

global optimisation algorithms. These problems might give rise to multiple Pareto fronts in multi-objective optimisation problem, ideally corresponding to different 'valleys' of the fitness function. Under ideal circumstances, one is only interested in the best front, i.e. the one closest to the utopia point. However, real-life applications often demand a more robust approach that accounts for uncertainties possibly leading to violation of (some of) the constraints. For that purpose, lower-ranked Pareto fronts can be useful, as they contain good solutions that might still be feasible when uncertainties are taken into account. For an example of this application, see [**?**];

6. **Investigate effect of choice of coordinates on clustering.** In this paper, we have looked at the conventional decision domain, i.e., departure epoch and time of flight. Future research could also focus on mathematically rewriting the problem into a dimensionless version, using variables such as planetary phase angle difference. These coordinate transformations might reshape the data into very different clusters, possibly more advantageous for subsequent optimisation. Perhaps more general statements regarding synodic period and relative semi-major axis can be made as well, such that different classes of transfers can be linked to suitable clustering methods;

7. **Use as first stage in hybrid ML.** Since clustering can attach labels to unlabelled data, it could also function as a precursor for supervised algorithms like neural networks, support vector machines, etc. These machine learning models can — after proper training — estimate if a new individual (i.e., a trajectory design) is feasible or not, without having to calculate the fitness function. It could also be coupled with fitness function proxy models, which use neural networks or Gaussian Process Models to estimate the fitness values. An examples of this can be found in [14].

8. **Investigate the threshold value in greater detail.** The threshold value, introduced in this paper, is simple but effective in making an artificially generated data set clusterable, but also comes at a cost: an additional hyperparameter is introduced that needs to be tuned. Obviously, the ideal value for this parameter depends on the transfer problem (and thus on the ΔV values) but also on the type of clustering algorithm used.

II
Background

WE WILL NOW PROCEED to provide more background information on different topics underlying this thesis. Chapter 4 will comment and elaborate on the different aspects of trajectory optimisation, with special emphasis on global optimisation and its main challenges. We will continue with a discussion of the most important astrodynamical notions and shape-based methods that were put to use in this work in Chapter 5. Last but not least, we will examine machine learning in Chapter 6, in particular the application of clustering methods to trajectory optimisation problems.

4 Global optimisation

This chapter will discuss the necessary background of global optimisation. It is paramount to have a good understanding of this topic, because it provides the basic framework for finding optimal trajectories. Being the ultimate pursuit of the overall minimum (or maximum) of a function, global optimisation presents many challenges; especially when applied to astrodynamical problems.

We will start by discussing the mathematical principles behind optimisation in Section 4.1. The different approaches to and techniques for optimisation will then be treated in Section 4.2. Finally, we will turn our attention to global optimisation techniques, with an emphasis on heuristics, in Section 4.3.

4.1. Mathematical formulation of the low-thrust trajectory optimisation problem

Every dynamical optimisation problem deals with finding the minimum (or equivalently, the maximum) of the *cost functional* J [15]:

$$J = \Phi\left(\mathbf{x}(t_f),\, t_f\right) + \int_{t_0}^{t_f} L\left(\mathbf{x}(t),\, \mathbf{u}(t),\, t\right)\, \mathrm{d}t \tag{4.1.1}$$

in which $\mathbf{x} = \mathbf{x}(t)$ is the state vector and $\mathbf{u} = \mathbf{u}(t)$ is the control vector. The cost function therefore consists of two terms [1]:

1. **The Mayer term** $\Phi(\cdot)$: only dependent on the final time and state. Examples in spaceflight are the arrival epoch and arrival conditions.
2. **The Lagrange term** $L(\cdot)$: represents the cost depending on the integral over time of the state and control vectors. Examples include — but are not limited to — propellant mass, actuator overshoot, and velocity increase.

The problem in which no limitations are imposed on $\mathbf{x}$ and $\mathbf{u}$ is called an *unconstrained* optimisation problem. In general, however, there will be constraints that complicate the problem. In the case of spacecraft trajectory design, we cannot choose *any* $\mathbf{x}$, as we need to adhere to the physical laws governing the motion of the spacecraft through space. Additionally, there are control constraints like maximum thrust magnitude, thrust pointing limitations, etc.

The dynamical constrains are often written as a set of differential equations, as it is usually derived using Newton's second law:

$$\ddot{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{f}_i \tag{4.1.2}$$

where the acceleration $\ddot{\mathbf{x}}$, i.e., the second derivative of the position with respect to time, is given as the sum of all specific forces on the spacecraft. These include e.g. the gravitational forces exerted on the spacecraft by various bodies, thrust, and solar radiation pressure. Equation (4.1.2) is therefore the model that describes the dynamic environment that the spacecraft is subject to. It usually contains many terms which can be (indirectly) dependent on time and the spacecraft's position or velocity. For this reason, it can also be written in the more general form

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}(\mathbf{x}, \mathbf{u}, t). \tag{4.1.3}$$

Apart from the dynamic constraint, there are often also terminal constraints (in this case boundary conditions), which specify the value of the state and/or time at the initial and final epochs [2]:

$$b_L \leq b(\mathbf{x}_0, t_0, \mathbf{x}_f, t_f) \leq b_U \tag{4.1.4}$$

Lastly, the control and state vectors (or functions thereof) may also be subject to path constraints, specifying the allowable values at every epoch t:

$$p_L \leq p(\mathbf{x}(t), \mathbf{u}(t), t) \leq p_U \tag{4.1.5}$$

In summary, the goal of optimisation is to find both the optimal trajectory $\mathbf{x}(t)$ and optimal control $\mathbf{u}(t)$ which both satisfy the constraints *and* minimise/maximise the cost function J.

4.2. Optimal control strategy

This section provides a bird's eye view of the different approaches to an optimisation problem, placing this research in the wider context of the problem. A taxonomy of the different approaches and techniques will be given, as well as a brief discussion of the (dis)advantages of these methods, culminating in a rationale for the method that was used in this research.

Figure 4.1 presents one possible taxonomy of the different optimisation approaches that can be used to tackle the problem as discussed in Section 4.1. The first distinction is that between an analytical and numerical perspective. Ideally, one would like to have an exact representation of the solution, however, this goal is in practice often unattainable in trajectory design due to the already mentioned complexity of the problem, due to the nonlinear nature of the differential equations and the presence of path constraints. This complexity and the fast advances in computing power have shifted the trend towards numerical approaches, which discretise the continuous problem so that it can be solved by computers [12] [13].

In the numerical approach, most literature observes a dichotomy between two approaches: the direct and indirect method. The former converts the continuous optimal control problem into a discrete optimisation problem by parametrising the state and/or control vectors, so that nonlinear programming or heuristic techniques can be deployed to find a suitable solution. The other method is called indirect because the problem is first converted to a Hamiltonian boundary value problem and successively discretised for optimisation [2]. The main issue with the indirect methods, however, is that they rely on analytical techniques based on calculus of variations and Pontryagin's principle, and therefore yield complex derivations in complex problems such as (MGA) trajectory design [15]. For this reason, usually a direct method is chosen in spacecraft trajectory problems, which is also seen in the increased popularity of so-called shape-based methods (see Section 5.2), that try to represent the trajectory (i.e., the state) in a parametric way and add the control to the objective function [1]. Although direct methods are now mostly used, they do have a major disadvantage compared to their indirect counterparts: the obtained solution(s) are not guaranteed to be optimal, due to the lack of optimality conditions which are derived in the indirect method. An example of trajectory optimisation using an indirect method based on Pontryagin's minimum principle is given in [16], where the solutions are obtained through a heuristic algorithm. For more information on optimal control in trajectory optimisation, the reader is referred to [17].

Going down one level in Figure 4.1, we arrive at different *techniques* that can be applied in both direct and indirect methods. The two most common are the so-called shooting and collocation techniques. Others exist, such as differential inclusion (only applicable in a direct method), but they are not very common and therefore not treated here. For more information on this technique, the reader

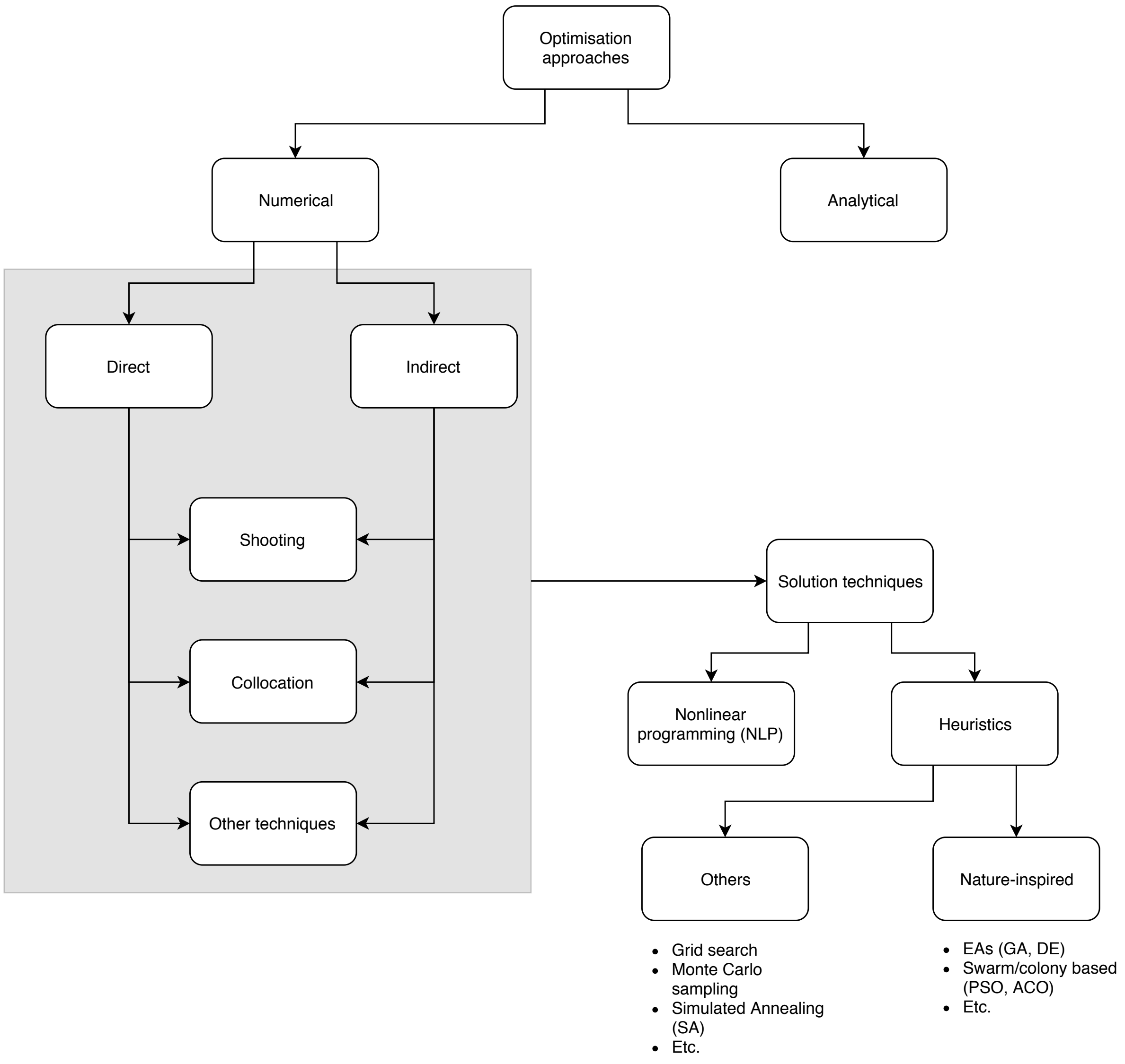

Figure 4.1: Taxonomy of optimisation approaches and solution techniques, adopted from [1] and [2].

is referred to [18]. Shooting is the procedure in which the initial state is fixed and the trajectory is propagated through time using an initial guess for the parameters. This procedure is repeated until convergence to the terminal conditions is reached. Collocation methods are different in the sense that they do not need explicit integration of the equations of motion, for the trajectory is now defined by values of the state vector at so-called *collocation points*. Between these points the state is approximated by an interpolant, and the derivative of this function is set equal to the state derivative as prescribed by the equation of motion in these collocation points [1] (see Figure 4.2[1]).

Up to this stage, the optimal control problem has been rewritten into a static parameter optimisation problem, which has to be solved numerically. Nowadays, there are two main branches of solution techniques: nonlinear programming (NLP) and heuristics. The former has been around for a few decades and is mainly based on gradient information. The second option is a method based on searching and evaluating the objective function, until the algorithm has converged to an optimum. One aspect of these methods that is often described as an advantage, is that they do not need gradient information. This does, however, mean that the algorithms become black boxes where little to no problem-specific knowledge is exploited. Despite this, heuristics, and in particular the nature-inspired methods, have proven their worth in very complex trajectory optimisation problems and have be-

[1]Retrieved from `https://mec560sbu.github.io/2016/09/30/direct_collocation/` on 25/08/2019.

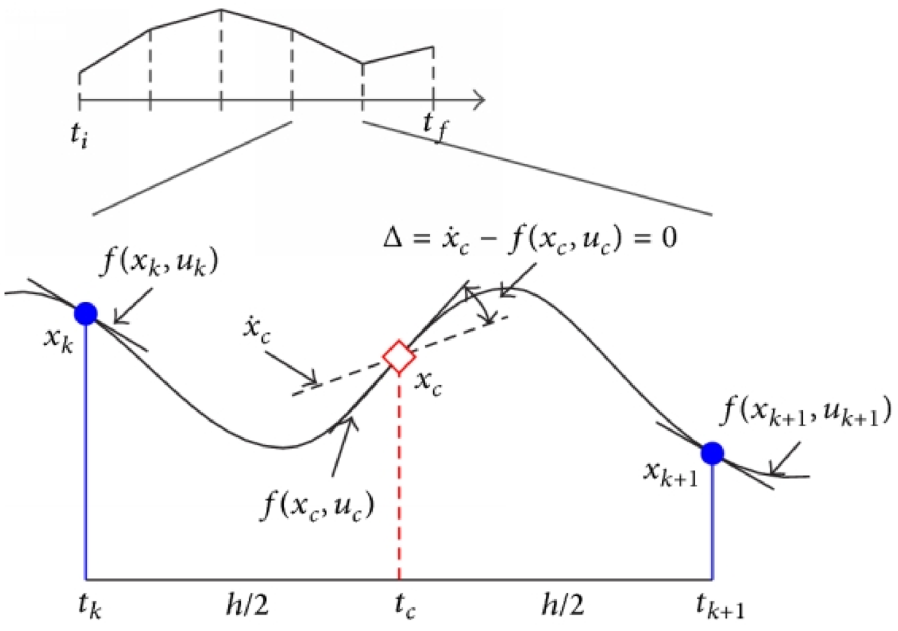

Figure 4.2: Illustration of the collocation method as described in the text.

come the method of preference. The reason for this trend can also be found in the lack of gradient information exploitation: heuristics are more likely to converge to a *global* optimum. NLP solvers are instead heavily sensitive to the initial guess and therefore often converge to a *local* optimum. The latter statement also explains the fact that heuristics are often called global optimisation algorithms.

Given their popularity, versatility and performance in complex optimisation problems, some global techniques will be explored further in the next section (Section 4.3).

4.3. Global optimisation techniques

As described in Section 4.2, heuristics have surpassed NLP techniques for solving complex trajectory optimisation problems. This subsection will briefly describe some of the most widely used algorithms as of today.

4.3.1. (Random) sampling methods

Except for the grid search method, all algorithms in this category are in some way dependent on realisations of random (also called stochastic) variables. For this reason, the reader might also encounter the term *stochastic methods* in literature, e.g. dealing with the topic of evolutionary algorithms and Monte Carlo search.

Grid search

Arguably, grid search (GS) is the most naive method for global optimisation. In GS, the domain of every variable is subdivided into a certain number of intervals, which can be different for every variable. In this way, a multi-dimensional grid is created, of which every grid point is sampled and its corresponding fitness value is computed. The overall lowest fitness is then considered to be the 'global' optimum, although no proof or warranty can be given that this point is actually the true global optimum.

Being a naive method, GS suffers from issues which make it unsuitable for complex problems, such as MGA trajectory optimisation. The first problem arises in problems with a high search space dimensionality, i.e., many input variables exist, each having their own domain. In case every variable is sampled at the same grid resolution n, the total number of grid points equals n^m, where m is the

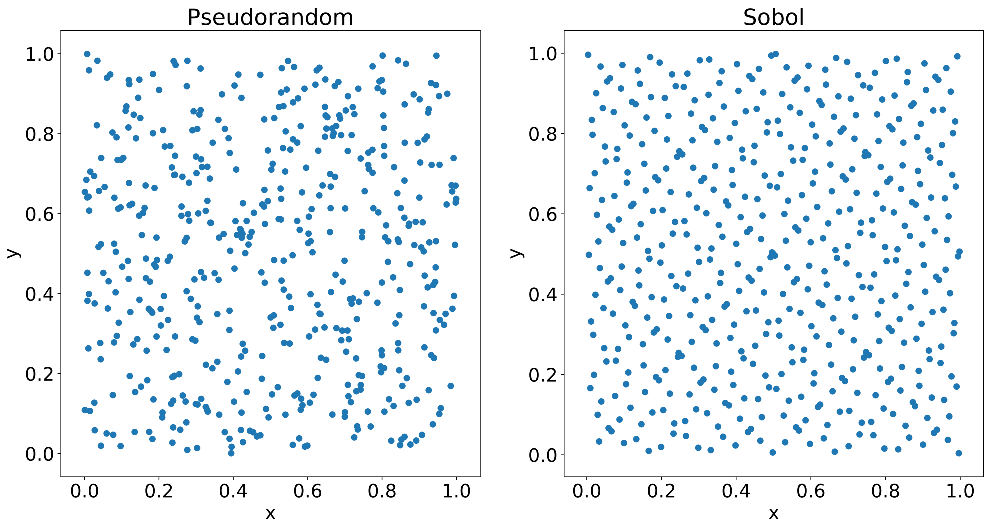

Figure 4.3: Comparison of a pseudorandom and a quasi-random (Sobol) generalisation for points in a two-dimensional domain.

dimensionality of the search space. Hence, the complexity of this algorithm is exponential with respect to the number of decision variables. Another drawback of GS is that a refinement of the grid does not necessarily lead to an improvement in the optimum value.

Monte Carlo search

Another sampling-based method is the so-called Monte Carlo search. Instead of sampling the search space in regular intervals, Monte Carlo search randomly samples the domain to evaluate the objective function at these points. Usually the points are samples from a uniform distribution, so that the entire search domain will (in theory) be covered more or less evenly, but this heavily depends on the exact realisation of the random distribution. A frequently mentioned flaw of Monte Carlo methods is, however, that the eventual coverage of the domain is far from uniform, even though a uniform distribution has been taken from which the points are drawn. The stochastic nature also implies that the found optimum can change from run to run, depending on how many points are sampled. These two complications challenge the convergence speed and the quality of the found optimum. To partially eliminate the existence of uncovered areas in the design space, a Monte Carlo optimisation is often run multiple times to assess whether a better optimum can be found.

Instead of relying on a completely (pseudo)random number sequence, one could also draw values from a quasi-random sequence, which offers a more uniform distribution of the points in the search domain. One such sequence is the Sobol sequence, developed by and named after the Russian mathematician Ilya M. Sobol. Figure 4.3 illustrates the difference between the Sobol sequence and a pseudorandom generalisation: it is clear that the pseudorandomly generated numbers do not evenly cover the domain and leave some areas more or less uncovered. In the context of optimisation this might mean that the algorithm potentially misses better optima. On the right side of the figure, it is visible that a Sobol sequence has the regularity of a grid sampling method but still has some randomness. The latter overcomes the problem with grid sampling as discussed in the previous section (grid refinement not yielding a better optimum).

4.3.2. Nature-inspired algorithms

A relatively new class of algorithms in optimisation is that of the nature-inspired heuristics. Although still based on the sample-and-evaluate method that is exploited in grid search and Monte Carlo search, the sampling phase is now more sophisticated and guided by some principle. Three main branches of nature-inspired heuristics are distinguished here: evolutionary, swarm, and colony-based. The following sections will discuss the principles, advantages, and downsides of each class.

No free lunch theorem

As opposed to the classical approaches in optimisation that use gradient information to converge to a (local) optimum, global methods are usually agnostic of the specifics of the problem at hand and one might therefore argue that there is no good reason to prefer one class of global algorithms over another. This statement would also be supported by the *No Free Lunch Theorems* as proposed by Wolpert and Macready in their 1997 paper [19], as they argue that no algorithm shows universal dominance over other algorithms when performance is averaged out over a large collection of problems. The crux is, however, in the nuance of 'averaged out': there might be a dominance (i.e., better performing algorithm) for a specific class of problems that share some common features. In that case, Wolpert and Macready use the terminology that the algorithm *matches* the problem to some extent. We might therefore identify one or two algorithms that may perform best on a certain interplanetary transfer problem.

In their 2019 paper, Joyce and Herrmann review the No Free Lunch theorems and discuss their implications on heuristics. They draw an important conclusion applying to the comparison of different heuristic algorithms: "When free lunches are possible, their prominence tends to depend crucially on the optimisation metric used." [20] This statement implies that in case we have a preferred algorithm, its number one position is heavily dependent on the metric. For instance, it might be the best performing algorithm if one considers the minimum function value to be the metric, but only rank e.g. third if one compares them with respect to convergence speed or number of function evaluations. One can therefore actually find one or two algorithms that perform better than the others, but comparison should be done carefully and its outcome is very problem-specific.

As we will see in Chapter 6, these statements also apply to clustering algorithms. Many such algorithms exist, based on different underlying principles. Again, no single clustering algorithm is universally dominant over the others when averaged out over a large number of different problems, so the problem of matching a problem to a suitable clustering algorithm arises here as well. Similarly, if we find an algorithm that seems to outperform the rest in some type of problems, we see that this heavily depends on the performance metric used, which is usually some kind of validation index.

Evolutionary algorithms

The class of evolution-based algorithms is the one that was first pioneered by scientists and researchers, with the introduction of the evolutionary algorithm (EA), which was inspired by Darwin's theory of evolution. In this metaphor, designs are represented by individuals in a population, and features by genes. Individuals can recombine to form offspring in the form of new individuals and mutation events can alter the genetics of the population. Additional genetic operators, such as genetic crossover (the exchange of some gene values between individuals) and migration, are also frequently applied. During every generation, all individuals are evaluated for their 'fitness' (i.e., objective function values). The fitness values are then used in a selection policy to determine which part of the population can reproduce and which is to be discarded. This process is then repeated until a convergence criterion is met.

Swarm-based algorithms

Another class of global optimisation algorithms is inspired by animal swarms. The best-known example is *Particle Swarm Optimization*, as first described by Kennedy and Eberhart [21]. Just like the EAs, PSO initialises a population of candidate solutions that continuously changes in order to locate the global optimum. It however lacks mutation, crossover and migration operators; instead, it mimics swarm behaviour as encountered in fish and bird populations. The individuals in the population will namely follow the fittest solution, while also being guided by their own notion of the best solution.

PSO is ubiquitous in engineering optimisation, and its popularity has also reached the realm of interplanetary trajectory design. See for example [16], [22], [23], [24].

Colony-based algorithms

Another class inspired by animal behaviour is that of the colony-based algorithms. Famous examples are *Ant Colony Optimisation* (ACO) and *Artificial Bee Colony* (ABC).

In contrast to the previously discussed algorithms, ACO is best suited for combinatorial problems. In astrodynamics, this is encountered in MGA trajectory design, as the flyby sequence is a discrete problem. See for example [25] and [26] for a case study of MGA planning using ACO.

5 Astrodynamics

The previous chapter introduced the optimisation problem that arises when one aims to find the optimal trajectory from one point in space to another, thereby minimising some cost function. We recall here that one of the constraints in the optimal control problem is defined by the dynamics of the problem; in the end, an optimal trajectory must of course adhere to the physical laws governing the motion of the spacecraft. This chapter will therefore discuss this part of the problem: how can one best represent the dynamics of the spacecraft?

Before one can start answering this question, it is first important to examine what constitutes the dynamic environment of the spacecraft, in terms of the forces acting on it and the motion of the celestial bodies. Having treated this, we can look into different representations of the motion and assess their advantages and shortcomings.

5.1. Dynamic environment: gravity and perturbing forces

In the case where analytic solutions are not available, because they are either hard to compute or even impossible to find, one must resort to numerical methods. One of them is the direct method, as has been discussed shortly in Chapter 4. In this method, we propagate the dynamics of the spacecraft and its environment numerically; i.e., the forces are computed at each epoch and integrated using a numerical scheme to obtain the change in velocity and position.

The framework of this approach is Newton's second law of motion, which in turn leads to the equations of motion of N point masses with constant mass m_N:

$$\frac{\mathrm{d}^2\mathbf{r}_N}{\mathrm{d}t^2} = \frac{1}{m_N}\sum_i \mathbf{F}_{i,N} \tag{5.1.1}$$

where $\mathbf{r}_N$ is the position of body N in some inertial reference frame (which makes the omission of pseudoforces possible) in metres, t the time in seconds, and $\mathbf{F}_{i,N}$ the force from source i acting upon body N, in $\frac{\mathrm{N}}{\mathrm{kg}}$. The fidelity of the model is hidden in the sum of the specific forces, on the right side of the equations, as a more complex and simultaneously more accurate model involves more forces, e.g. due to third-body effects and solar radiation pressure. The right-hand side also includes the control forces, which guide the spacecraft's motion through space.

The following subsections will treat the most important forces for an interplanetary transfer model, which is based on some assumptions. First of all, it is assumed that the velocity of the spacecraft is sufficiently low to be able to apply classical mechanics, such that relativity effects can be discarded. Furthermore, it is assumed that there are no atmospheres, so that aerodynamic forces can be eliminated from the equations of motion. A third assumption is that only gravitational forces of close, heavy bodies are taken into account. Obviously, this description is vague and the exact number of bodies taken into account depends on the requirements on propagation accuracy. The more bodies are included, the closer the model approximates the real, physical world.

5.1.1. Gravitational forces

In the case of interplanetary transfers, it can be defended that the spacecraft is at all times far enough from the attracting bodies such that it experiences a point-mass gravitational force. In the opposite case, one should model the gravity field with e.g. spherical harmonics to mimic the irregularity of the gravity field at a close distance. For interplanetary transfer purposes, it can be assumed that the spacecraft is — during its transit at least — far away enough from these bodies that the point-mass gravity model is accurate enough, so we can make use of Newton's law of gravity:

$$\mathbf{F}_{G,j} = -G\frac{m_i m_j}{r_{\text{ij}}^3}\mathbf{r}_{\text{ij}} \tag{5.1.2}$$

where $\mathbf{F}_{G,j}$ is the gravitational force exerted by body i on body j, G the gravitational constant (being equal to approximately $6.67 \cdot 10^{-11}$ m^3 kg s^{-2}, m the mass of the corresponding body in kg, and $\mathbf{r}_{ij}$ the vector of body i to body j (in m).

The equation above represents the gravitational force of one body acting upon another; hence, all contributions from the bodies that are included in the simulation must be computed.

5.1.2. Solar radiation pressure

Highly reflective spacecraft with a large frontal area are also prone to so-called solar radiation pressure. It is the force that results from photons, emitted by the Sun, bumping on the spacecraft's panels and either being reflected or (partially) absorbed. The magnitude of this force depends on the attitude of the spacecraft, its distance from the Sun, the exposed surface area, and the reflectance properties of the surface material. The importance of including this force is heavily dependent on the accuracy requirement imposed on the trajectory and on the physical properties of the modelled spacecraft.

The magnitude and direction of this force can be computed using the following equation:

$$\mathbf{f}_{SRP} = -C_R\frac{WA}{mc}\hat{\mathbf{e}}_S \tag{5.1.3}$$

with $\mathbf{f}_{SRP}$ being the specific solar radiation pressure force in N/kg, C_R the reflection coefficient, W the Solar radiation flux in W/m^2, A the frontal area of the spacecraft in m^2, m its mass in kg, c the speed of light, and $\hat{\mathbf{e}}_S$ the unit vector pointing from the spacecraft towards the Sun. Since the radiation pressure force always points *away from* the Sun, there is a minus sign in front of the right side of the equation.

5.1.3. Thrust force

The last force that is considered here is the thrust force, generated by the spacecraft's engine(s). The direction in which the thrust is applied, relative to the local frame, is called the thrust direction. It is usually taken as a control variable, similar to the thrust magnitude itself.

Since this research focuses on low-thrust transfers, it is wise to look at the different types of electric thrusters that are used to provide low-thrust propulsion to spacecraft. An overview of these types, along with their corresponding specific impulses and thrust levels is given in Table 5.1, taken from [10]. The table gives an indication of the attainable specific impulse and thrust magnitude of different engine types, and can therefore be used for setting proper thrust constraints during the optimisation.

The typical low-thrust acceleration range is put into perspective in Figure 5.1. We see that the thrust force is usually the second-most dominant force, after the Sun's gravitational attraction. Only in the neighbourhood of the planets, we see that their gravity dominates the Sun's and the engine's thrust force. Consequently, their attraction can be ignored during most of the interplanetary transfer. Lastly, we see that the solar radiation pressure is almost always an order of magnitude or more lower than the thrust, meaning that we can safely ignore it for preliminary trajectory optimisation.

For an electric thruster, its force output can be determined with the following formula if the electric power consumption as well as its efficiency is known [10]:

$$F_{\text{thrust}} = \frac{2\eta \cdot P_{\text{elec}}}{g_0 \cdot I_{\text{sp}}} \tag{5.1.4}$$

or, equivalently, its power consumption from the thrust force:

Table 5.1: Overview of electric thruster types and their properties [10]

	Type	Vacuum I_{sp} (s)	Thrust range (N)
Electrothermal	Resistojet	150-700	0.005-0.5
	Arcjet	450-1,500	0.05-5.0
Electrostatic	Ion	2,000-6,000	$5{\cdot}10^{-6}$-0.5
	Colloid	1,200	$5{\cdot}10^{-6}$-0.05
	Hall Effect	1,500-2,500	$5{\cdot}10^{-6}$-0.1
Electromagnetic	Magnetoplasma-dynamic	2,000	25-200
	Pulsed plasma	1,500	$5{\cdot}10^{-6}$-0.005
	Pulsed inductive	2,500-4,000	2-200

$$P_{\text{elec}} = \frac{F_{\text{thrust}} \cdot I_{\text{sp}} \cdot g_0}{2\eta} \tag{5.1.5}$$

where F_{thrust} is the thrust force in Newton, η the engine efficiency, P_{elec} the electric power consumption in Watt, g_0 the gravitational acceleration on the surface of the Earth (i.e., ≈ 9.81 m/s^2), and I_{sp} the specific impulse in seconds. Especially the power input is an important figure, as it states how much power is needed to provide the required thrust level. This number should be assessed for feasibility, especially in the case of solar electric propulsion (SEP), where the available power is heavily dependent on the distance from the Sun and thus also on the trajectory itself. For this reason, required engine power may be a better constraint in SEP-driven spacecraft and must be modelled as a function of distance from the Sun and spacecraft orientation.

5.2. Shape-based methods

Instead of representing the control function with discrete parameters, the trajectory itself, i.e. the state $\mathbf{x}(t)$, can also be parametrised. This is the approach of the so-called shape-based methods. The control ΔV is then included in the objective function [1]. Since Petropoulos [4], the number of proposed methods has increased considerably, and the most prominent ones will be treated in the following sections. Table 5.2 provides an overview of the methods considered below, and compares them with respect to their ability to produce 2D/3D trajectories, which boundary conditions can be imposed, and if thrust constraints can be included. For a detailed discussion of these shaping methods, see [3].

Table 5.2: Comparison of major shape-based methods, including the references to the original publications. P and V refer to position and velocity boundary conditions, respectively. Based on [3].

	2D/3D	Boundary conditions	Thrust constraint	Reference
Exposin	2D	P	No	[4], [27]
Inverse polynomial	2D + 3D	P + V	No	[28]
Spherical	3D	P + V	No	[29]
Fourier series	2D + 3D	P + V	Yes	[30], [31]
Hodographic	3D	P + V	No	[3] [5]
Pseudo-equinoctial	3D	P + V	Yes	[32]
Pseudo-spectral	3D	P + V	Yes[1]	[33]

[1]Included as a penalty in the cost function.

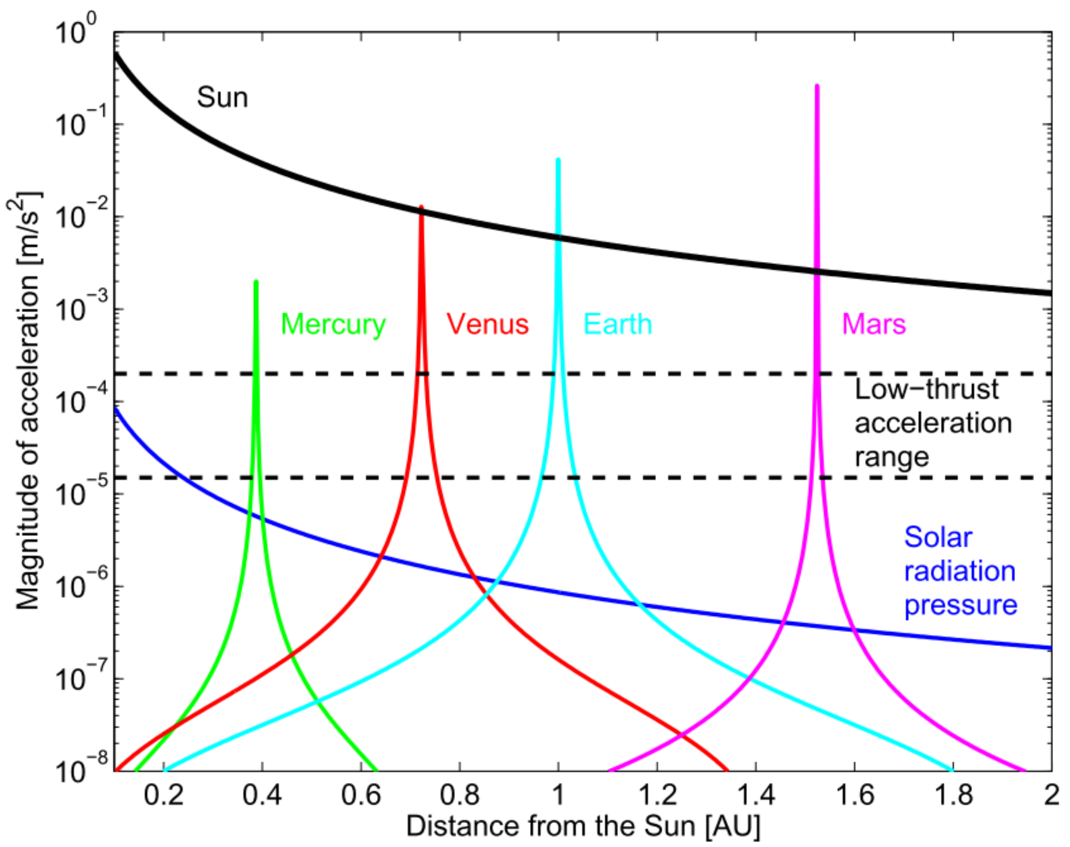

Figure 5.1: Comparison of force magnitudes within the inner solar system. [3]

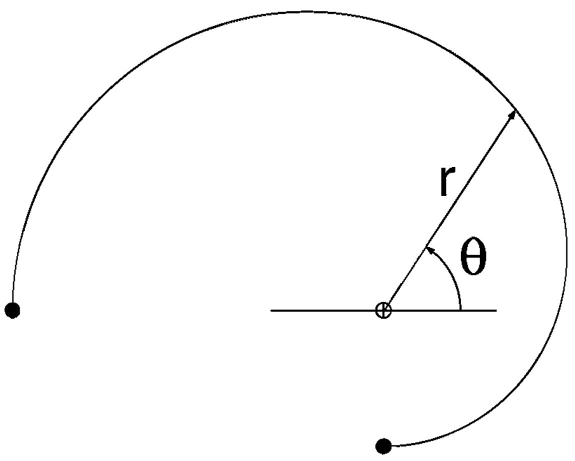

Figure 5.2: Example of an exponential sinusoid with the definition of the polar coordinates r and θ [4].

5.2.1. Exponential sinusoids

The first shape-based method was proposed by Petropoulos and Longuski in their 2004 paper [4]. Having studied various shapes, they concluded that the exponential sinusoid (hereafter exposin) was the "most promising of these for representing the powered portion of flight between gravity-assist bodies." [4, p. 788]

The trajectory described by an exposin is given by the following parametric equation in polar coordinates:

$$r = k_0 e^{k_1 \sin(k_2\theta+\phi)} \tag{5.2.1}$$

with r the radius, θ the angle between the radius vector and some reference direction, and k_0, k_1, k_2, ϕ constants that form the parameters describing the exposin. An example of an exposin along with the definition of the variables r and θ is given in Figure 5.2.

The significance of the constant parameter ϕ can be seen by setting θ equal to zero. In this case, we observe that ϕ is the 'phase angle' of the exposin; in other words, it controls the orientation of the trajectory in the 2D plane. k_0 is a scaling factor to match the size of the shape to the scale of the trajectory, and k_1

$$\tan\gamma = k_1 k_2 \cos\left(k_2\theta + \phi\right) \tag{5.2.2}$$

The normalised thrust acceleration a is defined as follows:

$$a \equiv \frac{F}{\mu/r^2} \tag{5.2.3}$$

and it can be written as a function of the shape parameters as follows:

$$a = \frac{(-1)^n \tan\gamma}{2\cos\gamma}\left[\frac{1}{\tan^2\gamma + k_1 k_2^2 s + 1} - \frac{k_2^2(1-2k_1 s)}{(\tan^2\gamma + k_1 k_2^2 s + 1)^2}\right] \tag{5.2.4}$$

with:

$$s \equiv \sin(k_2\theta + \phi) \tag{5.2.5}$$

For a detailed discussion of the method and derivation of these equations, the reader is encouraged to consult [4] and [27].

Although exposins allow for a quick and insightful analysis of the transfer options in between gravity assists, there are a few drawback to this method that limit its applicability and accuracy:

- **The imposed shape implies that thrust can only be defined a posteriori.** It is not possible to define the control profile beforehand, so a feasibility analysis has to be conducted after the determination of the trajectory to check whether the thrust constraints are still satisfied.
- **The trajectory is in general not optimal in terms of propellant usage or transfer time.** The fact that the trajectory is a priori molded into some analytical shape and fitting the thrust acceleration afterwards implies that the found trajectory is not optimal.
- **The thrust is constrained to the tangential direction only.** The use of exposins prohibits the use of thrust in another direction so that an analytical solution is possible. This also contributes to the lack of optimality of the solution.
- In a boundary value problem involving exposins, **a rendezvous solution is in general not found**. This is equivalent to the statement that the velocities at the departure and arrival points cannot be satisfied, which severely limits the applicability of this class of shaping functions.
- Lastly, **the trajectory is only defined in two dimensions.** Since they are described in polar coordinates, exposins are only defined in the plane by their very nature.

As we will see, some of these disadvantages are dealt with in the different shape-based methods below, although the a priori modelling is a feature that all methods share. Also, most methods suffer from sub-optimality of their generated shaped, but this is also inherent to the approach that underlies these methods.

5.2.2. Inverse polynomials

The problem of not being able to compute rendezvous trajectories using exposins was solved by Wall and Conway in 2009 [28]. Instead of expanding the formulation of exposins with two extra parameters, they proposed a new representation of the trajectory, based on an inverse polynomial:

$$r = \frac{1}{a + b\theta + c\theta^2 + d\theta^3 + e\theta^4 + f\theta^5} \tag{5.2.6}$$

Hence, this function allows the six scalar boundary conditions to be solved for, since there are exactly six free function parameters a to f. Obviously, this implies that the transfer time be left free; if one wants to specify the transfer time (e.g. in rendezvous problems), then an extra parameter g is added:

$$r = \frac{1}{a + b\theta + c\theta^2 + d\theta^3 + e\theta^4 + f\theta^5 + g\theta^6} \tag{5.2.7}$$

The equation for $T_a = f(\theta)$, which allows for the extraction of the thrust magnitude from the shape, is not restated here. The reader is referred to [28] for a complete overview of all relevant equations.

A three-dimensional extension of the inverse polynomial shaping function using cylindrical coordinates was proposed by Wall in his conference paper [34]. Hence, this method solves two of the issues related to Petropoulos's and Izzo's method, as now the full boundary value problem can be solved to obtain rendezvous transfers and 3D trajectories can now be modelled more accurately.

5.2.3. Spherical shaping

The next milestone in shaping methods was the proposal of the spherical shaping framework by Novak and Vasile [29]. This method was proven to be a further generalisation of the exposin and inverse polynomial methods, i.e., these are merely special cases of the spherical shaping method. Therefore, this shaping is carried out in three dimensions using the spherical coordinates r, θ, and ϕ. Like the previously described methods, the parameter is chosen to be θ and the geometry is completely described by the two shaping functions $r = R(\theta)$ and $\phi = \Phi(\theta)$. The third shaping function $t = T(\theta)$ then yields the evolution of time along the transfer trajectory.

The authors claim that "a relatively wide set of shaping functions for R and Φ can be used such that the boundary conditions can be satisfied analytically". In their paper, however, they use the following two functions for the geometrical shaping:

$$\begin{cases} R(\theta) = \dfrac{1}{a_0 + a_1\theta + a_2\theta^2 + (a_3 + a_4\theta)\cos\theta + (a_5 + a_6\theta)\sin\theta} & (5.2.8) \\ \Phi(\theta) = (b_0 + b_1\theta)\cos\theta + (b_2 + b_3\theta)\sin\theta & (5.2.9) \end{cases}$$

5.2.4. Fourier series shaping

In 2012, Taheri and Abdelkhalik outlined a novel shaping method, which does not assume a fixed shape, in contrast to the methods discussed previously [30]. Instead, they propose a finite Fourier series (FFS) expansion of the states, which are matched with the equations of motion at so-called discretisation points (DP). The DPs also allow setting thrust constraints, whereas imposing thrust constraints is not possible in the classical shape-based methods.

The authors propose the parametrisation of the polar coordinates in terms of the time t:

$$\begin{cases} r(t) = \dfrac{a_0}{2} + \displaystyle\sum_{n=1}^{n_r} \left\{ a_n \cos\left(\frac{n\pi}{T}t\right) + b_n \sin\left(\frac{n\pi}{T}t\right) \right\} & (5.2.10) \\ \theta(t) = \dfrac{c_0}{2} + \displaystyle\sum_{n=1}^{n_\theta} \left\{ c_n \cos\left(\frac{n\pi}{T}t\right) + d_n \sin\left(\frac{n\pi}{T}t\right) \right\} & (5.2.11) \end{cases}$$

where a, b, c, and d are coefficients (determined by imposing the boundary conditions, equations of motion, and thrust constraints), T the total trip time, and n_r, n_θ the number of Fourier terms for the radius and polar angle equations, respectively.

From Equations (5.2.10) and (5.2.11) it is clear that the trajectory is restricted to two dimensions. An extension to three dimensions is given in their 2016 paper; the reader is referred to [31] for more information.

5.2.5. Pseudo-equinoctial shaping

Instead of shaping the trajectory in terms of polar/spherical coordinates, one can also parametrise the so-called pseudo-equinoctial elements, which are more natural to the dynamic nature of the problem, as in a perfect Keplerian orbit only one of the variables changes through time. Perturbations can then often be modelled as secular and periodic variations of the other elements.

This property is exploited in the pseudo-equinoctial shaping method as described by De Pascale and Vasile in their 2006 paper [32]. The authors recognise that there are two distinct approaches, one most suitable for solar electric propulsion (SEP) and one for nuclear electric propulsion (NEP). In the first case, they propose the linear combination of a linear and trigonometric term, whereas NEP favours the use of an exponential of the longitude.

With this method, 3D trajectories can be found which in general tend to overestimate propellant mass consumption, but form a good first guess that can improve further search in more sophisticated approaches.

5.2.6. Pseudo-spectral shaping

The last shaping method discussed here is the one developed by [33], which is based on shaping functions that take the form of (truncated) power series for the cylindrical coordinates r, θ, and z:

$$\begin{cases} r(\theta) = \displaystyle\sum_{i=0}^{k} a_i\theta^i & (5.2.12) \\ \theta(t) = \displaystyle\sum_{i=0}^{l} b_i t^i & (5.2.13) \\ z(\theta) = \displaystyle\sum_{i=0}^{m} c_i\theta^i & (5.2.14) \end{cases}$$

The numbers k, l, and m represent the order of truncation for the series and their values should be chosen such that the accuracy requirements are met. It is also worth noting that the r and z functions depend on θ, which is in turn a function of the true independent variable t (time). The coefficients a_i, b_i, and c_i are to be found using optimisation; four of them can be found for each coordinate series by applying the problem-specific boundary conditions.

The accelerations in cylindrical components can be found with the following equations:

$$\begin{cases} a_{prop,r} = \ddot{r} - r\dot{\theta}^2 + \frac{\mu}{r^3}r & (5.2.15) \\ a_{prop,\theta} = 2\dot{r}\dot{\theta} + r\ddot{\theta} & (5.2.16) \\ a_{prop,z} = \ddot{z} + \frac{\mu}{r_{s/c}^3}z & (5.2.17) \end{cases}$$

For a detailed derivation, see pages 34-35 in [33].

5.2.7. Hodographic shaping

All previous methods are based on shaping of the trajectory in position space, i.e., matching the departure and arrival positions and devising a trajectory between them. This trajectory describes the position at all epochs given a few parameters.

The hodographic method, in contrast, shapes the trajectory in *velocity space*. This means that the departure and arrival velocities are matched to those of the initial and target orbits, and a mathematical description of the trajectory is formulated in between. This formula yields the velocity at every point along the trajectory and can be integrated to obtain the positions at all epochs. An example of this process is given in Figure 5.3: the right plot shows the hodographs of the initial (blue) and target (red) orbits along with the transfer trajectory (black). It can be seen that the transfer orbit intersects the hodographs of the two orbits, implying that the velocity boundary conditions have been met. After integrating this transfer trajectory over time, we obtain the shape of the transfer in 'normal' (i.e., space) coordinates (see the left figure). Note that a perfectly circular orbit with $e = 0$ would be represented by an infinitesimally small point in the V_r vs V_t graph, with $V_r = 0$, as the tangential velocity is constant for this case. In the more general case of an elliptical orbit, its hodograph will be a circle in velocity space [6].

Figure 5.4 shows the velocity hodographs for three different types of Kepler orbits (elliptical, parabolic, and hyperbolic). Indeed, all shapes are circular, whereas the offset of the centre is determined by the type of orbit. In the case of a parabola, the null velocity at $r = \infty$ is shown as the circle touching the x-axis. Hence, an increasing eccentricity relates to shifting the velocity hodograph down the y-axis.

The advantage of this method is that the velocity (Neumann) boundary conditions are easily met, whereas the position (Dirichlet) boundary conditions are obtained without having to resort to iterative algorithms. Also, this method lends itself well to low-thrust trajectories and allows design both with and without degrees of freedom (i.e., free parameters). Gondelach proposes velocity functions that are linear combinations of power, sine, cosine, and power times (co)sine functions [3]. He also proposes two variants of his method, making a distinction between time and polar angle as the independent variable. Both methods have their advantages, as time-based shaping offers more intuitive insight into the dynamics, whereas polar angle-based shaping is more related to the (periodic) geometry of the transfer trajectory.

Base functions

As we have already seen, the hodographic shaping method fabricates a trajectory in velocity space. This implies that we find a function that describes the velocity in each direction as a function of the independent variable, being time t or polar angle θ. One could select some random function, but it is actually much more useful if we select functions with more structure.

Finding this structure is not difficult, knowing that we must integrate the velocity functions in order to find the position as a function of time or polar angle. It is hence useful to select functions that are analytically integrable, i.e., we can readily find their primitive. Secondly, the definite integral is a linear operator, so choosing a linear combination of integrable base functions seems useful. The general description of some velocity function $V(t)$ is shown in Equation (5.2.18).

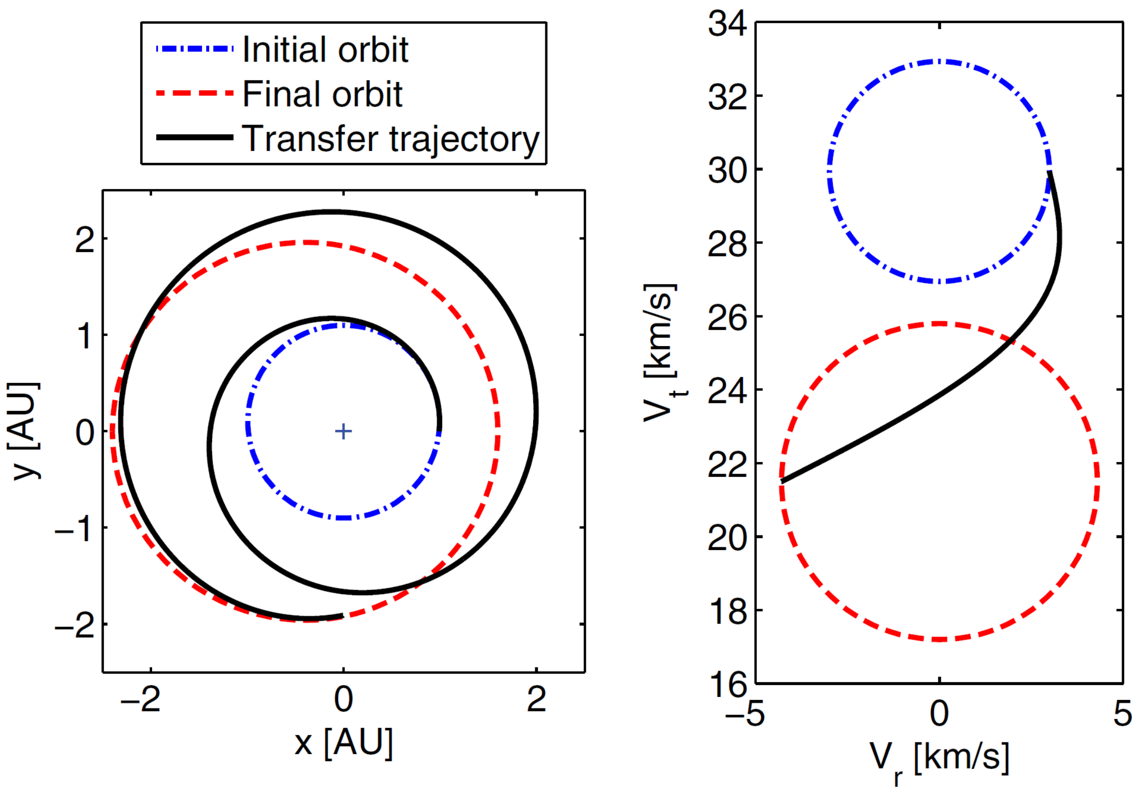

Figure 5.3: The figure on the left shows the trajectory in position space, whereas the figure on the right shows that trajectory as a hodograph; taken from [5].

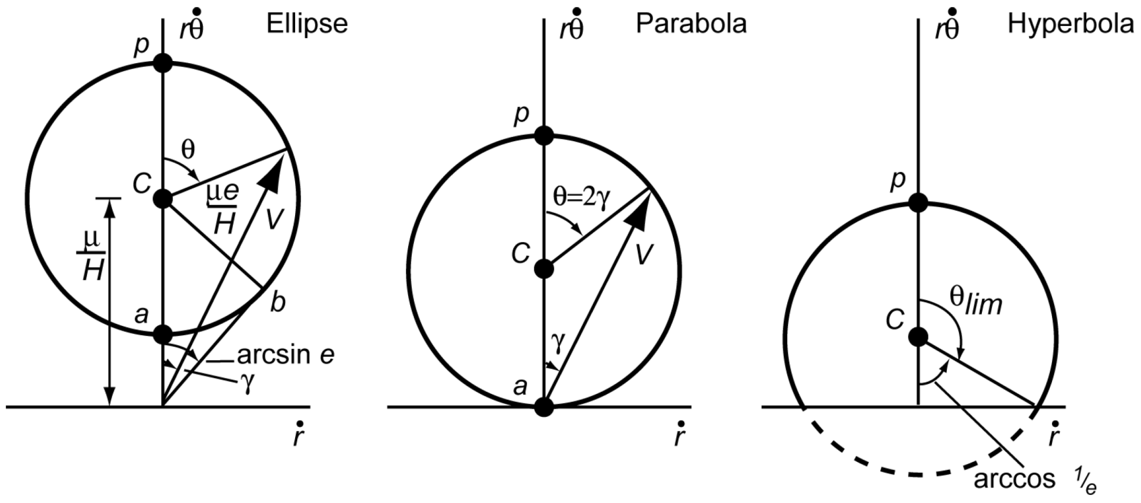

Figure 5.4: Illustration of the velocity hodographs for three types of Kepler orbits [6].

$$V(t) = \sum_{i=1}^{n} c_i v_i(t) \tag{5.2.18}$$

Here, $V(t)$ is the velocity function for one of the three base directions, n the number of base functions, c_i the i$^{\text{th}}$ coefficient, and $v_i(t)$ the i$^{\text{th}}$ velocity base function. The specific combination of base functions used in this thesis is shown in Equation (5.2.19) and was based upon the recommendations given by Gondelach himself [3].

$$\begin{cases} r(t) = c_1 + c_2 \cdot t + c_3 \cdot t^2 + c_4 \cdot t\cos(f) & (5.2.19) \\ \theta(t) = c_5 + c_6 \cdot t + c_7 \cdot t^2 + c_8 \cdot t\sin(f) + c_9 \cdot t\cos(f) & (5.2.20) \\ z(t) = c_{10} \cdot \cos(f(N+0.5)) + c_{11} \cdot t^3 \sin(f(N+0.5)) + c_{12} \cdot t^3 \cos(f(N+0.5)) & (5.2.21) \end{cases}$$

In these equations, $f = \frac{2\pi}{t_f - t_i}$ and N is the number of revolutions.

Mathematical description of the hodographic shaping method

To obtain the position as function of the independent variable (here time t is considered), we integrate the three velocity functions with respect to τ, which is a dummy variable for t:

$$\begin{cases} r(t) = r_0 + \int_0^t V_r \, \mathrm{d}\tau & (5.2.22) \\ \theta(t) = \theta_0 + \int_0^t \frac{V_\theta}{r} \, \mathrm{d}\tau & (5.2.23) \\ z(t) = z_0 + \int_0^t V_z \, \mathrm{d}\tau & (5.2.24) \end{cases}$$

Notice that the equation for the polar angle θ requires a factor $\frac{1}{r}$, which is the Jacobian for this set of coordinates. To find the values of the coefficients in front of the base functions, we need the boundary conditions. The trivial ones are:

$$V_r(0) = V_{r,0} \qquad V_r(t_f) = V_{r,t_f} \tag{5.2.25}$$
$$V_\theta(0) = V_{\theta,0} \qquad V_\theta(t_f) = V_{\theta,t_f} \tag{5.2.26}$$
$$V_z(0) = V_{z,0} \qquad V_z(t_f) = V_{z,t_f} \tag{5.2.27}$$

Those are not the only boundary conditions that we can find, however. We also happen to know the boundary conditions in position space, that is, initial and final r, θ, and z, because the trajectory needs to coincide with the position of the departure and target body at $t = 0$ and $t = t_f$, respectively. Hence, we can rewrite Equations (5.2.22) to (5.2.24) as follows:

$$\begin{cases} \int_0^{t_f} V_r \, \mathrm{d}\tau = r_f - r_0 & (5.2.28) \\ \int_0^{t_f} \frac{V_\theta}{r} \, \mathrm{d}\tau = \theta_f - \theta_0 & (5.2.29) \\ \int_0^{t_f} V_z \, \mathrm{d}\tau = z_f - z_0 & (5.2.30) \end{cases}$$

Since this procedure allows us to determine 3 coefficients per direction, any extra base functions' coefficients need to be determined by an optimisation method. In general, it holds true that having more base functions implies more accuracy, although it is no guarantee of significant improvement.

Having obtained the velocity functions with their coefficients, we can now proceed to calculate the thrust that is necessary to fly this trajectory. To do this, we express the equations of motion in cylindrical coordinates:

$$\begin{aligned} \ddot{r} - r\dot{\theta}^2 + \frac{\mu}{d^3}r &= f_r \\ r\ddot{\theta} + 2\dot{r}\dot{\theta} &= f_\theta \\ \ddot{z} + \frac{\mu}{d^3}z &= f_z \end{aligned} \tag{5.2.31}$$

where $d = \sqrt{r^2 + z^2}$. These equations hold under the following assumptions:

- Only the Sun exerts a gravitational force upon the spacecraft (other celestial bodies' gravitational pull is ignored);
- The only other force acting on the spacecraft is the thrust force **f**;
- The spacecraft has negligible mass w.r.t. the Sun's mass (restricted two body problem).

Since the specific thrust force is equal to the thrust acceleration, a simple integration of the thrust acceleration w.r.t. time yields the ΔV:

$$\Delta\mathrm{V} = \int_0^{t_f} ||\mathbf{f}||\ \mathrm{d}t \tag{5.2.32}$$

6 Machine learning

Machine learning (ML) is one of the core aspects of this research, as it constitutes the computational meta-algorithm that will optimise the trajectory design problem at hand. During optimisation, clustering methods were used to identify clusters of feasible solutions pertaining to interesting launch windows, and these clusters will be transformed into new (disjoint) search domains that can be explored by a global optimisation heuristic.

The aim of this chapter is to explain the rationale behind the choice of ML models and their working, as well as the aspect of software implementations. The chapter is structured as follows: the first section will deal with the basic notions in ML and the plethora of models and algorithms, providing more insight in the problem and highlighting the problem of choosing a model for this research. The most relevant models will be discussed further in Section 6.3. Finally, the last section will discuss a few more practical aspects, being training the algorithm and discussing assessment of its performance on a test set.

6.1. Machine learning tasks

Before we dive into different machine learning models, it is useful to define some key tasks that can be performed by machine learning models in general, so that we can appreciate its wide spectrum of applications. Not all tasks are relevant for this research, although they are included for the sake of completeness. In arbitrary order, we can identify (based on [35]):

- **Classification:** in a classification problem, the algorithm tries to assign a certain data point to a category. This can be in a binary sense (class I or II), also called dichotomous classification, but it may be the case that there are more than just two categories. As a dichotomous example in optimisation, we may be interested in determining whether a design (i.e., a collection of values for design variables) corresponds to a feasible solution (1) or not (0). A multiple-class problem is for instance the classification of solutions into different launch windows.

- **Regression:** in this case the problem is to attach a numerical value to a specified input. Examples are the cost of a car, the length of a person, ΔV and trip time. In contrast to classification, the model maps the input to a continuous output space, whereas classification relates to discrete outputs. In trajectory optimisation, a Gaussian process model for the ΔV and propellant mass was proposed in [14], where part of its task was to perform regression on the input variables.

- **Structure and pattern recognition:** the tasks described above are usually performed with *supervised learning* techniques, meaning that the labels are already available. In the contrary case, we might still want to find patterns in the data set without possessing explicit information in the form of labels. The algorithms of this *unsupervised* kind can then produce labels for the initially unlabelled data, allowing a follow-up supervised algorithm to further work on the data, thus paving the way for hybrid machine learning algorithms.

- **Optimal decision path:** in the case where there is an entity that performs actions in an environment, also known as an *agent*, one is often interested in which sequence of events is optimal in some way. This problem occurs in many fields and in astrodynamics one could for example think of the combinatorial problem that arises in MGA trajectories.

- **Feature extraction:** although an important task in general machine learning problems with large data sets and number of features, feature extraction does not play a very important role in optimisation of spacecraft trajectories, for the features that define the dimensionality of the search space are all considered important and contribute to the objective. This is in contrast to general optimisation, where it might be that a designer adds a certain feature to the optimisation process, which might not be influencing the objectives considerably (see for example [36]). For instance, the design of a re-entry vehicle shape is inherently an optimisation problem, but the parametrisation of the shape will involve parameters that contribute more and less to the heat load and downrange, so that they may be left out of the optimisation process.

6.2. Heritage of machine learning in trajectory optimisation

As discussed in the introductory chapter, low-thrust trajectory optimisation problems have a complex nature and often many computational resources are required to find a global optimum. Many techniques have been investigated and applied, with varying success. Especially problems with many parameters are hard to solve within a reasonable time frame, due to their high dimensionality and enormous search space. This section will describe the research that has been carried out on the topic of application of machine learning in the field of optimisation. Of course, this is a broad field and therefore the discussion below is not exhaustive. Where appropriate, the reader is referred to more detailed literature.

Before diving into the heritage, it is first useful to get a grip on the most common applications of ML in the context of (trajectory) optimisation:

- **Objective function estimation:** probably the most computationally intensive task in trajectory optimisation is the calculation of the objective function. Machine learning can help in estimating the objective functions, reducing calculation effort with limited loss of accuracy. By doing so, one creates a *surrogate* or *proxy model* of the actual fitness evaluation.

- **Search space pruning:** the initial search space in the optimisation algorithm usually contains regions where most or all combinations will result in poor objective values. 'Naive' algorithms ignore this and continue searching in these areas. It is more beneficial, however, if an algorithm could recognise these areas and 'cut' them out; this process is called search space pruning. This is often done manually, but large search spaces would actually benefit from an automated pruning technique to get rid of unpromising designs, e.g. in the form of an ML algorithm.

- **Neurocontrol:** especially low-thrust trajectories require continuous thrust to be applied for manoeuvring. This adds complexity to the problem, as the thrust times are not discrete anymore: the thrust becomes a (semi-)continuous function of time, which is to be optimised. This problem is called *optimal control* and neural networks have been applied to this field to speed up and improve optimisation results. This combination is called *neurocontrol*.

- **Multiple gravity assist trajectory combinatorics:** many interplanetary trajectories exploit the principle of momentum exchange to boost their heliocentric velocity without spending propellant. This is called a gravity assist (or swing-by) and heavily complicates trajectory optimisation problems. Machine learning can aid here by e.g. applying techniques from the field of reinforcement learning.

An in-depth discussion of so-called *knowledge discovery* in multi-objective optimisation is given in [37], which is the first paper in a series of two. This part treats the extraction of knowledge from the often large datasets that are generated in multi-objective optimisation problems in the most general way. The authors make a distinction between implicit and explicit knowledge representation, where the first is subject to interpretation by humans (giving rise to subjectivity) and thus explicit

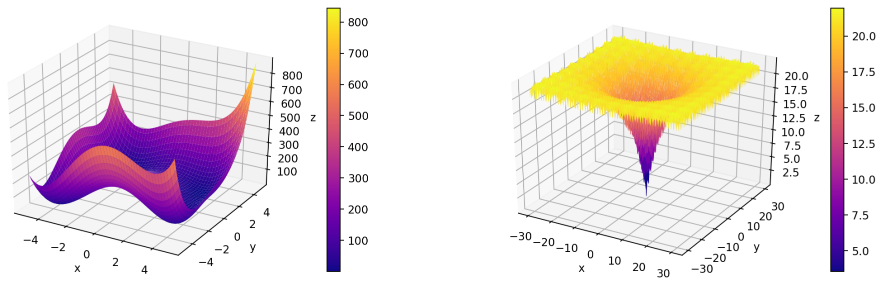

Figure 6.1: Functions used in [7] to test their method.

representation is preferred, especially for the concept of *Knowledge-Driven Optimisation* (KDO). The latter of course relates to this research, as the goal is to actively use machine learning techniques in optimisation procedures, with a focus on 'online' machine learning (i.e. the extracted knowledge is used *during* the optimisation). In this case, it is claimed that this actually demands an explicit representation of features; implicit data is not sufficient but can of course still aid in the process. For instance, implicit methods like neural networks or support vector machines can assist in building a surrogate model of the objective function, speeding up the function evaluation.

Part B of the paper is a follow-up study and implements four methods of knowledge extraction in multi-objective optimisation problems [38]. These are then applied to three case studies on production systems in industry, allowing for a comparison of the four proposed methods. Two of those are unsupervised models (sequential + flexible pattern mining), one is supervised (classification trees), and the fourth method is a hybrid approach, joining the forces of an unsupervised method (clustering) and classification trees. Although the researchers extensively compare the different methods, it remains unclear how the actual optimisation benefits from the introduction of machine learning techniques in terms of convergence and quality of the found optimum.

The use of classification for the creation of rules is a trend that is also found in two papers of Liu et al. [36, 39]. These two papers are on the cutting plane between pruning and ML applied to optimisation, because rule-based classifiers are deployed to prune the variable-specific domain, therefore cutting down the number of necessary fitness function evaluations. However, as discussed previously, this method was only benchmarked in artificial optimisation problems, leaving a gap for applications of this method to real-life engineering problems such as ours.

A similar but more extensive research on the application of data mining techniques on optimisation was carried out by Chen and Huang [7]. Similar to Liu et al., the authors propose the use of classification and association rules to narrow down the search space. Also, a third method based on clustering is investigated. The three different methods are then each applied to two different optimisation problems, being the unconstrained and constrained versions of the Bumpy problem. In both cases, the optimisation is carried out using Sequential Quadratic Programming (SQP), a gradient-based method, both with and without data mining techniques. In both cases, the optimisation with data mining assisted pruning yielded a higher likelihood that the global optimum is found, if the search is carried out multiple times. Even better results are obtained when this method is combined with a heuristic (evolutionary) algorithm. The authors claim that they were able to find the global optimum in every run, yielding a likelihood of 100% of finding the true optimum.

Multiple papers treat the potential advantages of applying machine learning techniques to *multi-objective* optimisation problems, as the existence of a Pareto front implies that no single optimal solution can be found, but that the decision is left to humans. Here, the Pareto front is the mathematical representation of the conflict between two (or more) objectives, such that the preference of the decision maker herself (themselves) underlies the final decision. Sato et al. therefore argue that knowledge discovery (i.e. ML) can assist in identifying patterns that are usually obfuscated by the complexity and sheer size of the optimisation problem [40]. To this extent, they use a hybrid of clustering and association rule learning to obtain largely implicit information hidden in the solutions. This paper is relevant for it discusses clustering in both the design and objective spaces and successively makes use of the clusters to come up with association rules. These rules are then expected to accommodate

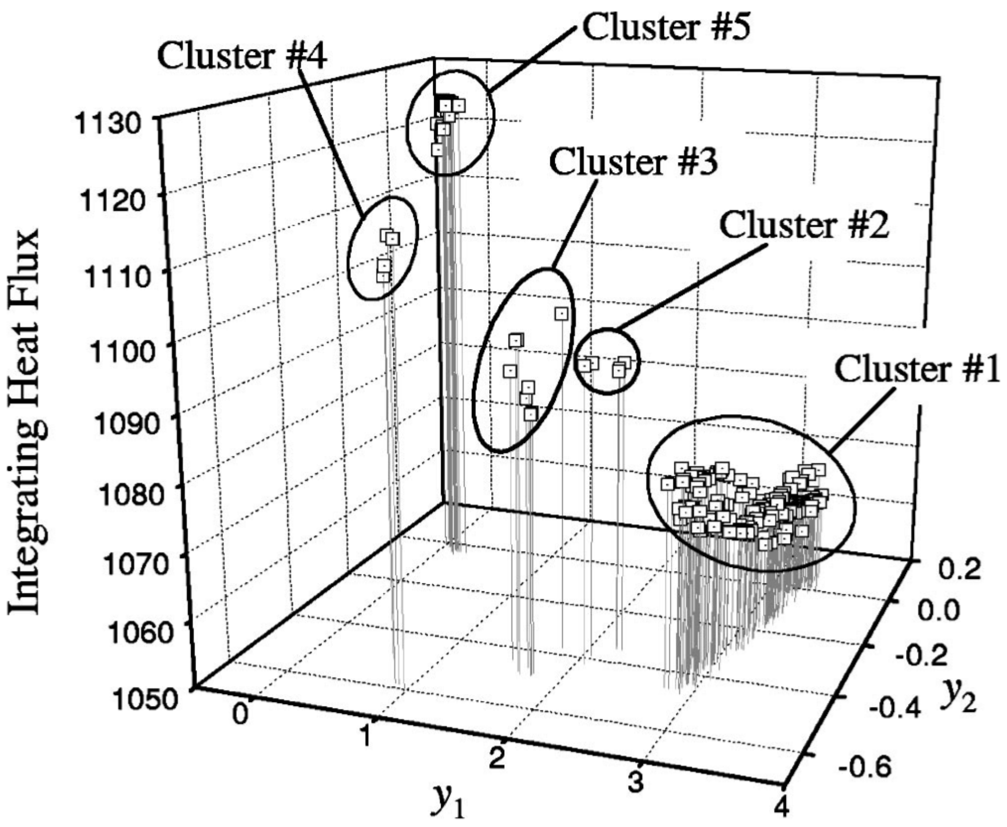

Figure 6.2: Visualisation of the clustering procedure in [8].

insightful features of the solutions.
The information extracted by these procedures is however implicit (mostly presented in a visual form), and ideally we want to have explicit representations of the data so that the optimisation algorithm can reduce its search space autonomously.

An application of clustering to single-objective turbine blade optimisation is studied in [8]. In this optimisation problem, the design space had 90 dimensions, rendering the clustering and visualisation very difficult. For that reason, dimensionality reduction was first applied in the form of a Karhunen-Loève transformation, which produces two new design variables that are a linear combination of the 90 original variables. The outcome of the clustering is illustrated in Figure 6.2: we see that five clusters arise, where more than 70% of the solutions reside in cluster 1. We also see that there are other distinct groups of solutions, clustered around higher values of the integrating heat flux. Though not globally optimal, these solutions appear to represent different, but similar designs for the blade, and might thus be interesting in case the designs from cluster 1 cannot be realised in practice.

6.3. Clustering

In the case where labels are unknown, we can still extract relevant information about the data, by using unsupervised machine learning methods. *Clustering* is one such example and tries to identify groups of similar data points, without knowing what the distinctive feature(s) is (are). In the case of space pruning, clustering can identify clusters of trajectories which are often spread in a quasi-periodic way due to the repetitive motion of celestial bodies [41]. Of course, we do not have any a priori labels for these clusters, as the clusters represent groups of trajectories that belong to a certain launch window. These regions can then be used as new bounds for subsequent optimisation runs, where the search domains can be constrained in a linear way (yielding rectangular boxes) or using non-linear constraints (yielding arbitrary shapes).

Cluster existence, uniqueness, and other philosophical issues

Due to its unsupervised nature, clustering faces some unique issues, some of which are more philosophical in nature. We can ask ourselves the following questions on clustering a given data set:

- Do clusters actually *exist* within the data set?
- Is there a 'natural' clustering for the data at hand?
- When is a given partitioning of the data set into clusters a *good* one, i.e. when does the clustering possess a high quality?
- Does a universally applicable clustering method exist?
- Etc.

The first question is an *existence* question and the second relates to a notion of *uniqueness* (is there a unique, best cluster?). The third question, then, asks whether it is possible to and how to *measure* the quality of a clustering. Fourthly, we ask ourselves the question, is there a so-called *free lunch* (see also Section 4.3.2)? One might think that there are concrete, satisfying answers to these questions, but reality is very different. In fact, satisfying answers do not exist, although one can make some sensible statements. For example, cluster quality assessment differs from situation to situation, so researchers should be explicit in the requirements they set for high quality clusters [42]. Similarly, the existence of clusters in a data set cannot be identified in a binary fashion, but one can use some methods (such as those outlined in [43]) to find a more nuanced answer in terms of the degree of clusterability.

Lastly, a universally applicable clustering method that always works does not exist. This result is very similar to the No Free Lunch Theorem by Wolpert and Macready [19], treated in more detail in Section 4.3.2. There are, however, algorithms that match up well with a given problem; this means that one can still make statements about a clustering algorithm's performance, when one is explicit about the specific situation in which it is applied. A second corollary is that the algorithm's performance heavily depends on the metric used to judge it by [20].

Having looked at these fundamental and very important questions, we will now turn our attention to the different types of clustering algorithms that are commonly used in knowledge discovery.

Clustering algorithms

There is a plethora of clustering methods, differing in performance and complexity, and like optimisation algorithms, their application depends on the distribution and nature of the data. Another distinction can be made in the nature of the data supply: is the data set static or a dynamic stream? In the latter case, online clustering algorithms are often used as a data stream comes with its own complications: the clustering should be stable but also flexible enough to accommodate new data points and if necessary, increase/decrease the number of clusters. In the former case, offline algorithms suffice as the clustering is done on a fixed data set.

Different (static) clustering types are compared in Table 6.1. This table is not exhaustive, so the reader is referred to [11] for a comprehensive review of traditional and more modern clustering algorithms. Each type of clustering has its own advantages and disadvantages, such that the choice is heavily dependent on the nature of the data set and the number of data points. See also Figure 6.3 for a graphical overview of the performance of some clustering algorithms in different situations.

Flat vs non-flat geometry

Some algorithms, like K-means, are highly efficient and simple in their use, but do not work well with 'non-flat geometry' of the data. The term *non-flat geometry* refers to manifolds that have a non-zero curvature[1], e.g. spirals or concentric circles. In those instances, Euclidean distances might not be the best option for clustering. Figure 6.3 contains a few examples of such geometries: the top two rows contain instances of such geometry. If we would use simple Euclidean distances, we would obtain clusterings as the one in the leftmost column (MiniBatch KMeans). The reason for this is that two

[1]Source: `http://mathworld.wolfram.com/FlatManifold.html`, accessed on 22/08/2019.

Table 6.1: Comparison of the most common offline clustering types, along with examples for each category. See Table 22 in [11] for a comprehensive comparison.

Type	**Examples**	**Advantages**	**Disadvantages**
Centroid	K-means, PAM, CLARA	Efficient	Sensitive to outliers and K, drawn to local optimum, does not work well with non-flat geometry
Density	DBSCAN, OPTICS, Mean-shift	Non-flat geometry and high efficiency	Sensitive to parameters; does not work well with uneven density distributions
Distribution	DBCLASD, GMM		
Hierarchy	BIRCH, CURE, ROCK	Works with different kinds of shapes	K is a preset; high time complexity
Kernel	SVC, kernel K-means, MMC	Works well with high-dim. feature spaces	Sensitive to kernel type and parameters; high time complexity

points, belonging to two very different clusters, might be at a similar (Euclidean) distance from some centroid. Using this concept, we find clusters that are often far from the 'true' clustering.

To solve this problem, one could resort to other notions of cluster cohesion, e.g. point density or linkage. We see these techniques in action in Figure 6.3, looking at the DBSCAN, OPTICS, and Ward columns. It is clear that the two non-flat geometries fare well by these different clustering algorithms, and that the resulting clustering resembles our human intuition of what the 'true' clusters would be in this case. This also applies to the oblong groups of data in the fourth row (counted from above) in Figure 6.3. These types of clusters are encountered in trajectory optimisation, so investigating which clustering algorithms work best for these clusters is paramount. Again, we see that centroid-based algorithms like KMeans and MeanShift perform poorly, again

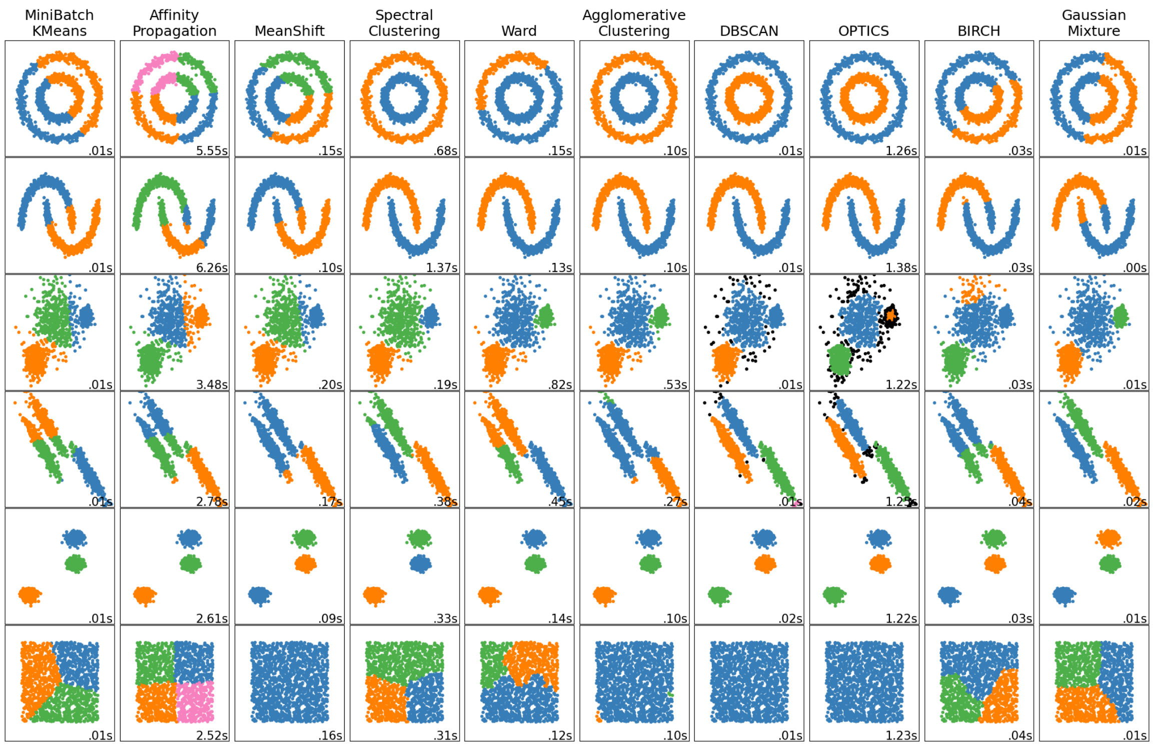

Figure 6.3: Comparison of clustering algorithms for different types of data set geometries[2]

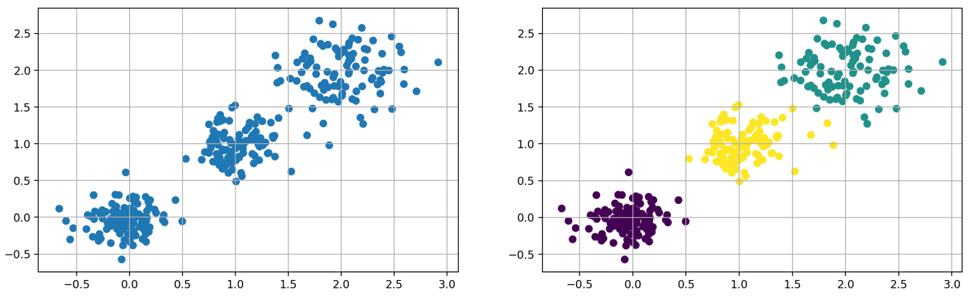

Figure 6.4: Clustering applied to three normally distributed data sets, yielding three classes (purple, yellow, and green).

Online clustering

The algorithms described before are all so-called offline clustering algorithms, as the entire sample set is considered static and known at runtime. In the case of assisted optimisation, this is not the case: we are dealing with a *dynamic* data set. The optimisation heuristic continuously evaluates new points in the search domain and these are subsequently fed into the *online* clustering algorithm. This poses an additional dilemma: we want the clustering to be both adaptive to new data points, but also stable, preventing the clusters from shifting around too wildly. In literature, this problem is referred to as the 'Stability-Plasticity Dilemma' [44] [45]. For most of the static algorithms listed above, there are online counterparts, with varying degrees of plasticity and stability. The nature of the problem will hence dictate which algorithm is best suited for the data at hand.

Cluster validation

How does one know whether the number of clusters is correct and whether the found clusters actually represent the data correctly? These questions relate to the concept of validity and are therefore of paramount importance to ask. In validating the clusters, one can answer the question based on internal and external measures. A third option of evaluating the validity of a clustering outcome is by comparing the results of different clustering algorithms, also called relative validation. For the second option, explicit labels for the data are necessary, so that the found clusters can be compared to the 'true', a priori, classification [46]. Of course, this contradicts the concept of unsupervised learning as there are usually no labels available. In that case, one could resort to human judgement, although this is often not possible in an automated process. For that reason, the discussion below will mostly focus on *internal* validation.

Several quantitative measures, i.e. indices, have been proposed in relevant literature. Examples of internal validity indices are the Cophenetic correlation coefficient (CPCC), the γ-statistic, and τ-statistic [47].

One approach could be to devise some mathematical function yielding a certain goodness fit, and evaluating this function for 1, 2, ..., c clusters. Two examples of this are the χ^2 and Kolmogorov-Smirnov statistics [44].

For a more detailed discussion of cluster validation, the reader is referred to Halkidi et al. [47] and Xu et al. [46].

[2]Source: `https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html`, retrieved on 13/10/2021.

III

Implementation

IN THE PREVIOUS PART we have seen the theory that forms the backbone of this thesis. In the coming chapters, we will dive deeper into the more practical side of the research. Chapter 7 will give a top-down overview of the software that was used to generate the trajectories and clusterings and describes the trade-off that was made for data storage. We will conclude this part with Chapter 8, which deals with the clustering algorithms in more detail.

7
Software

This chapter discusses the software that was used during this thesis. Section 7.1 treats the external libraries, written by third parties, that were used: Python, Tudat/TudatPy, and Pagmo/Pygmo. Section 7.2 then discusses the overall architecture, i.e., how the different components interact. Lastly, we discuss the database and the underlying trade-off that was made during the thesis in Section 7.3.

7.1. Third-Party Software

7.1.1. Python

Apart from the numerical propagation of orbits and/or Lambert targeting, the bulk of the analysis and research will be coded in Python. The decision to use Python instead of MATLAB, which is another widely used tool for (engineering) analysis and modelling, is based on five important advantages over MATLAB:

1. First of all, Python and its libraries are completely **free to use**. MATLAB is a commercial product and is therefore expensive. For the sake of reproducibility, everyone is able to download and code in Python without the need to pay a large amount of money for a license.

2. The existence of an **extensive package library** is a second key reason. Not only is the library vast, it is also actively maintained and developed. More specifically, Python features two of the most widely used machine learning libraries (scikit-learn and TensorFlow) in which many machine learning models and algorithms have been implemented.

3. Tudat's latest feature (as of Q3 2021) is the advent of a **Python wrapper interface called TudatPy**, enabling integration between simulation and analysis through an application programming interface (API)[1]. This way, users can call Tudat subroutines from Python code without having to write separate programs for simulation and analysis.

4. Another important consideration is the aspect of **portability**. Python can be run on virtually every major operating system (Windows, Linux, *nix, macOS, Android) and architecture (x86, x86-64, ARM)[2], whereas MATLAB is restricted to Windows, Linux and macOS on x86 and x86-64 architectures.

5. Lastly, there is an **active and large community** that develops, tests, and supports Python packages.

[1]https://tudat-space.readthedocs.io/en/latest/_src_about/about.html, retrieved on 5/11/2021.
[2]https://pythondev.readthedocs.io/platforms.html, retrieved on 5/11/2021.

7.1.2. Tudat and TudatPy

During the thesis, Tudat and its Python sibling TudatPy were used extensively to perform all trajectory-related calculations. Tudat is a C++ library built for Windows, macOS, and Linux that acts as a framework for astrodynamics research. It connects to various external libraries for ephemeris retrieval (NAIF SPICE), coordinate conversions (SOFA), Boost C++ (various tasks), and ESA's Pagmo/Pygmo optimisation libraries (see **??**).

Tudat's low-thrust components were heavily used to produce the hodographic shaping trajectories. In the first place, it provides a clear interface to the velocity functions that can be built using the atomic base functions and their primitives, which allows for easy evaluation of the trajectory in both velocity and position space. The library also encapsulates the integration of the trajectory, such that the ΔV can be extracted without much effort.

All features come with unit tests to verify the different functions and classes, so that the user need only test their own code. The library also has interfaces to the CSPICE and SOFA libraries, written by NASA's NAIF and the IAU, respectively. These provide accurate data and verified functionality regarding spacecraft, celestial bodies, and reference frames.

More information on Tudat and TudatPy, including code documentation and tutorials, is available online.[3] The code base can be found on GitHub.[4]

7.1.3. Pagmo and Pygmo

The optimisation part of the research was carried out with the Pagmo/Pygmo software libraries for C++ and Python, respectively. This section discusses the reasons behind this choice and the capabilities of these software packages.

Developed by the ESA Advanced Concepts Team, Pagmo and Pygmo are C++ respectively Python libraries for parallel global optimisation[5]. It provides an interface for optimisation problems and comes with several global and local optimisation algorithms supporting multi-objective problems and constraints. Examples include differential evolution (DE), non-dominated sorting genetic algorithm (NSGA), and multi-objective evolutionary algorithm with decomposition (MOEAD). It also features local optimisation algorithms as provided by the NLOPT, IPOPT, and SNOPT libraries, for which Pagmo contains wrapper classes and functions.

As the P stands for parallel, the library offers a so-called island model for the parallelisation of the optimisation. In this model, the entire population in for instance a genetic algorithm is split up in subpopulations, of which each is assigned to one 'island'. The island metaphor explains how migration is handled in this implementation: individuals can only recombine with other individuals on the same island, i.e. both individuals belong to the same subpopulation. There is however a mechanism of migration, invoked after some number of generations, that allows individuals from one island to migrate to another. This alteration of the original model allegedly decreases the algorithm's tendency to stagnate and promote the creation of a more diverse population [48]. This in turn can lead to finding a better optimum and/or faster convergence. It should also be noted that the island model does not restrict the user in her or his choice of algorithm; instead, any optimisation algorithm based on populations can be used if it adheres to the Pagmo API specifications.

Furthermore, the Tudat software package comes with a Pagmo/Pygmo interface included, simplifying the modelling and subsequent optimisation workflow even further. Based on the host operating system (Unix-like or Windows), the optimisation can be run in parallel in a multithreading or multiprocessing environment. In this architecture there is one caveat: all the libraries in the application that run in parallel, should be coded in an asynchronous way. This is not the case with e.g. the CSpice library, used for fetching ephemeris data of celestial bodies, which is used extensively in the Tudat package. For that reason, the optimisation application should invoke CSpice before entering parallel mode, avoiding threading issues when multiple instances of the program want to access the same memory space.

Again, verification routines are provided with the code, so that the user can check whether all functionality works as intended after installation.

[3] `https://tudat-space.readthedocs.io/`
[4] Tudat: `https://github.com/tudat-team/tudat`; TudatPy: `https://github.com/tudat-team/tudatpy`
[5] `https://esa.github.io/pagmo2/index.html`, retrieved on 3/7/2019.

7.2. Software architecture

An overview of the software architecture is shown in Figure 7.1. As can be seen, the application is split up in a server and a client component. The core component of the software is executed on the client side, which interfaces the server side that is used as a means of data persistence (see Section 7.3). The green part on the client side is responsible for trajectory computation and contains the optimisation back-end as well. These two modules communicate through the optimisation module, which was written as part of the yellow core part. The optimisation module in turn communicates with the pruning and analysis & visualisation modules. The pruning module is responsible for clustering, cluster validation, and storing trajectories, clusterings, and other data in the database back-end on the server side. Finally, the analysis & visualisation module allows for knowledge extraction by means of plots, tables, and debugging routines.

7.3. Data persistence

As discussed in the previous section, the data storage part of the software fulfils an important role. Many different combinations of variables gave rise to a very large number of trajectories that needed to be simulated, requiring efficient storage. At the same time, the trajectory storage should also allow for quick and efficient data retrieval. Thirdly, given the highly interconnected structure of trajectories, transfer cases, and clusters, the storage should be able to reflect these connections and use them to store, change, and delete data.

These requirements led to a trade-off between different solutions that are commonly used. The following options were identified:

- **Text file storage:** save all trajectories and clusterings in text files on a hard disk. This means that numbers are also stored as strings, implying that they will generally take up more bytes than when stored in binary form. Examples include plain `.txt` files, `.csv` files, and other such formats. Advantages include easy processing, compatibility with spreadsheet programs (if the `.csv` format is used), and human readability. Of course, this method also has some severe drawbacks, such as excessive storage usage, lack of relational structure, and no support for object storage.
- **Binary file storage:** instead of storing all data structures as Unicode strings, one could also opt for binary storage. This means that non-string types are first converted to binary data using certain data types (e.g. 8-bit integer, 32-bit float), such that more information can be stored compared to text-based files. For this reason, binary files offer more efficient data storage, but are much less transparent to humans, unless processed by a dedicated program. These files also require consistent conversion from and to binary using the appropriate data types: when a wrong type is assumed, the data read or written is wrong and will cause further problems.
- **Non-relational database:** upgraded version of binary files, allowing efficient storage of complex and/or diverse data in a database, without the hassle of creating intricate relationships between the different data structures. Advantages include compatibility with many cloud services, storage efficiency, and support for complex data sets. The downsides are, most importantly, the setup cost, lack of relational structure, and dependence on a certain framework.
- **Relational database:** most complex form of data storage (at least among the four discussed here). It combines binary data storage with the power of relationships between data structures, allowing for complex data retrieval and storage commands, e.g. using SQL technology. Apart from these advantages, it comes at the cost of a more complicated setup procedure, the need for DB access libraries (e.g. APIs for programming languages), and understanding of SQL or related query languages.

These four data storage methods, along with their advantages and downsides, are summarised in Table 7.1. Given the mentioned requirements (need for storage of large amounts of data, efficient data retrieval), and the presence of a highly interconnected structure inherently present in the data, a relational database was chosen as the storage solution for this research, in the form of a PostgreSQL database hosted in the cloud. The latter allows for redundancy in case of local computer failures and

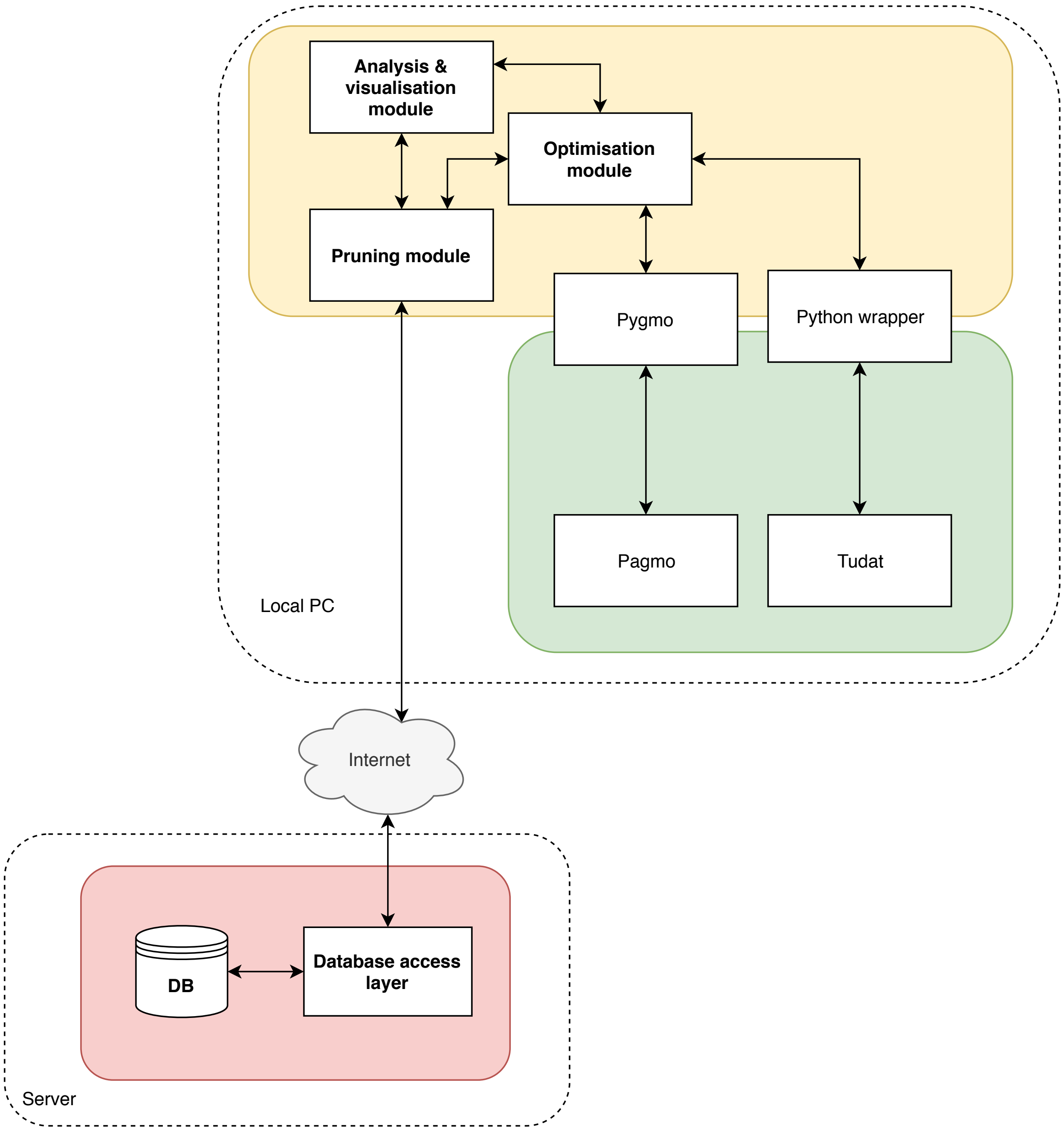

Figure 7.1: Overview of the proposed software architecture. Modules in bold will be written for this research; all other libraries are

Table 7.1: Data storage trade-off table.

Storage type	Storage efficiency	Relational structure	Compatibility	Setup difficulty
Text files	–	–	–/+	+
Binary files	–/+	–	–	–/+
Non-relational DB	+	–	+	–/+
Relational DB	+	+	+	–

independence of local storage space. For the structure of the database, i.e., the database schema, the reader is referred to Appendix B.

8 Clustering

In this chapter, we will study the implementation of the clustering component of the software package. Where possible, existing packages were used to avoid both reinventing the wheel multiple times and the need to verify and validate these implementations. Section 8.1 deals with the implementation of the 'conventional' clustering algorithms, such as DBSCAN, KMeans, GMM, and so forth. Next, more details about the the dip-test are given in Section 8.2.

8.1. Conventional clustering algorithms

One of the most popular machine learning packages for Python is `scikit-learn`[1]. It contains the implementation of most supervised and unsupervised models and can work with NumPy for vectorised computations, reducing CPU time in computationally intensive tasks. Clustering, support vector machines, artificial neural networks, and decision trees are all examples of supported methods. It also includes various validation indices, data preprocessing techniques (such as scaling and dimension reduction), and various other related functionalities. This combination makes this a very versatile Python package, well suited for a myriad of different machine learning tasks; for these reasons `scikit-learn` was also used in this thesis.

Any missing functionalities were supplemented with other third-party packages, such as HDBSCAN, which comes with the eponymous `hdbscan` Python package. All used libraries were either verified by their authors, or verified with a simple test problem. In some cases, all code had to be written from scratch or heavily adapted to ensure compatibility with the rest of the software, such as some validation indices. These functions were tested with simple cases mentioned in the scientific papers presenting these indices.

8.2. The dip-test and SkinnyDip

As mentioned in the scientific paper (see Chapter 2), the dip-test performs an important task in clustering the oblong-shaped clusters that are often encountered in trajectory optimisation problems. The dip-test was first coined by [49] and is used to test the hypothesis that a empirical distribution function is unimodal; i.e., it has only one 'peak'. By comparing the empirical *cumulative* distribution function (ECDF) to some benchmark distribution, the dip-test attaches a confidence value to the outcome of the hypothesis test. See Hartigan's paper for more details on the exact definition of the method, as well as the numerical implementation of the algorithm.

The original dip-test was however not suitable to detect multiple clusters of data, represented by multiple peaks in the empirical distribution function (EDF). Furthermore, it only works for univariate distributions, so applications to higher-dimensional clusterings were impossible.

[1]See also `https://scikit-learn.org`, retrieved on 3/10/2021.

This problem was solved by Maurus and Plant in their 2016 paper [50]. They propose an algorithm, called SkinnyDip, that repeatedly applies the dip-test on the ECDF that arises when the clustering data is projected onto one axis and converted into a cumulative distribution. The outcome of the dip-test tells us two things: whether *at least* one peak (cluster) was detected, and where it is located. Using some confidence value, we can extract the interval that is attached to this peak. If a peak and its interval are found, the overall interval is split at the inner interval bounds, such that three parts remain: left of the peak, the peak itself, and right of the peak. The first and last are then fed back into the algorithm, so each sub-interval is again checked for multimodality of the ECDF, until no more peaks are found.

References

[1] A. Shirazi, J. Ceberio, and J. A. Lozano, “Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions,” *Progress in Aerospace Sciences*, vol. 102, pp. 76–98, 2018, cited By :3 Export Date: 19 June 2019.

[2] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, “A review of optimization techniques in spacecraft flight trajectory design,” *Progress in Aerospace Sciences*, 2019, export Date: 19 June 2019.

[3] D. Gondelach, “A hodographic shaping method for low-thrust trajectory design,” Master’s thesis, Delft University of Technology, 2012. [Online]. Available: http://resolver.tudelft.nl/uuid:6a4f1673-88b1-4823-b2ef-9d864c84ab11

[4] A. E. Petropoulos and J. M. Longuski, “Shape-based algorithm for automated design of low-thrust, gravity-assist trajectories,” *Journal of Spacecraft and Rockets*, vol. 41, no. 5, pp. 787–796, 2004, cited By :159 Export Date: 3 July 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-7544249616&doi=10.2514%2f1.13095&partnerID=40&md5=e7c8b7fc312a1d2e63b47c07448f38d5

[5] D. J. Gondelach and R. Noomen, “Hodographic-shaping method for low-thrust interplanetary trajectory design,” *Journal of Spacecraft and Rockets*, vol. 52, no. 3, pp. 728–738, 2015, cited By :27 Export Date: 15 July 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84930165163&doi=10.2514%2f1.A32991&partnerID=40&md5=371cbf3d3dd4e379564e5096629521f0

[6] K. F. Wakker, *Fundamentals of Astrodynamics*. Delft: Institutional Repository Library, Delft University of Technology, 2015.

[7] T. Y. Chen and J. H. Huang, “Application of data mining in a global optimization algorithm,” *Advances in Engineering Software*, vol. 66, pp. 24–33, 2013.

[8] M. J. Jeong, B. H. Dennis, and S. Yoshimura, “Multidimensional clustering interpretation and its application to optimization of coolant passages of a turbine blade,” *Journal of Mechanical Design*, vol. 127, no. 2, pp. 215–221, 2005. [Online]. Available: http://dx.doi.org/10.1115/1.1830047

[9] Navigation and Ancillary Information Facility, “An overview of reference frames and coordinate systems in the spice context,” 2020. [Online]. Available: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/17_frames_and_coordinate_systems.pdf

[10] W. Larson and J. Wertz, *Space Mission Analysis and Design*, 3rd ed., ser. Space Technology Series. El Segundo, CA: Microcosm Press, 2005.

[11] D. Xu and Y. Tian, “A comprehensive survey of clustering algorithms,” *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015. [Online]. Available: https://doi.org/10.1007/s40745-015-0040-1

[12] J. T. Betts, “Survey of numerical methods for trajectory optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998. [Online]. Available: https://doi.org/10.2514/2.4231

[13] A. V. Rao, “A survey of numerical methods for optimal control,” *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[14] L. Bouwman, “Gaussian process models for preliminary low-thrust trajectory optimization,” Master’s thesis, Delft University of Technology, 2019.

[15] S. Kemble, *Interplanetary Mission Analysis and Design.* Springer Berlin Heidelberg, 2006. [Online]. Available: https://books.google.nl/books?id=m-k-qI6PsuUC

[16] M. Pontani, "Optimal low-thrust hyperbolic rendezvous for earth-mars missions," *Acta Astronautica*, 2019.

[17] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming.* Siam, 2010, vol. 19.

[18] A. Bressan and G. Facchi, "Trajectories of differential inclusions with state constraints," *Journal of Differential Equations*, vol. 250, no. 4, pp. 2267–2281, 2011.

[19] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997, cited By :4251 Export Date: 26 June 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0031118203&doi=10.1109%2f4235.585893&partnerID=40&md5=fac8c56be911367d556066800e863066https://ieeexplore.ieee.org/ielx1/4235/12703/00585893.pdf?tp=&arnumber=585893&isnumber=12703&ref=

[20] T. Joyce and J. M. Herrmann, *A Review of No Free Lunch Theorems, and Their Implications for Metaheuristic Optimisation.* Cham: Springer International Publishing, 2018, pp. 27–51. [Online]. Available: https://doi.org/10.1007/978-3-319-67669-2_2

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, Conference Proceedings, pp. 1942–1948 vol.4.

[22] F. Alonso Zotes and M. Santos Peñas, "Particle swarm optimisation of interplanetary trajectories from earth to jupiter and saturn," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 1, pp. 189–199, 2012.

[23] D. Izzo, *Global Optimization and Space Pruning for Spacecraft Trajectory Design.* Cambridge University Press, 2010, pp. 178–201, cited By :22 Export Date: 29 May 2019 Correspondence Address: Izzo, D.; European Space Agency, Advanced Concepts TeamNetherlands. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84932180996&doi=10.1017%2fCBO9780511778025.008&partnerID=40&md5=a7c2876b7da5f67481f7ccd011f406f8https://www.cambridge.org/core/books/spacecraft-trajectory-optimization/global-optimization-and-space-pruning-for-spacecraft-trajectory-design/D8D16263E67D6EAD641FD6873B72B3AA

[24] H. Yang, W. You, S. Li, and X. Jiang, "Optimal trajectories for a dual-spacecraft outer planet mission," in *UNKNOWN*, P. Singla, B. A. Jones, R. M. Weisman, and B. G. Marchand, Eds., vol. 167. Univelt Inc., 2018, Conference Proceedings, pp. 1185–1196.

[25] M. Ceriotti and M. Vasile, "Mga trajectory planning with an aco-inspired algorithm," in *60th International Astronautical Congress 2009, IAC 2009*, vol. 6, 2009, Conference Proceedings, pp. 4357–4370.

[26] G. Radice and G. Olmo, "Ant colony algorithms for two-impulse interplanetary trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1440–1444, 2006.

[27] D. Izzo, "Lambert's problem for exponential sinusoids," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 5, pp. 1242–1245, 2006, cited By :44 Export Date: 3 July 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33749640639&doi=10.2514%2f1.21796&partnerID=40&md5=8dd22b9eeb26a6fcd58252f07b4813de

[28] B. J. Wall and B. A. Conway, "Shape-based approach to low-thrust rendezvous trajectory design," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 1, pp. 95–101, 2009. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/1.36848

[29] D. M. Novak and M. Vasile, “Improved shaping approach to the preliminary design of low-thrust trajectories,” *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 1, pp. 128–147, 2011, cited By :55 Export Date: 11 July 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-78650959637&doi=10.2514%2f1.50434&partnerID=40&md5=5f9767f8e3955d2753b7c396afeba812

[30] E. Taheri and O. Abdelkhalik, “Shape-based approximation of constrained low-thrust space trajectories using fourier series,” *Journal of Spacecraft and Rockets*, vol. 49, no. 3, pp. 535–545, 2012. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84862649437&doi=10.2514%2f1.A32099&partnerID=40&md5=3a5adff2ab3eb6eeab98fed1ff316eed

[31] ——, “Initial three-dimensional low-thrust trajectory design,” *Advances in Space Research*, vol. 57, no. 3, pp. 889–903, 2016, cited By :24 Export Date: 15 July 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84957439075&doi=10.1016%2fj.asr.2015.11.034&partnerID=40&md5=6210fc9a3661af5afdac3b7461a82803

[32] P. De Pascale and M. Vasile, “Preliminary design of low-thrust multiple gravity-assist trajectories,” *Journal of Spacecraft and Rockets*, vol. 43, no. 5, pp. 1065–1076, 2006, cited By :76 Export Date: 11 July 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33750515616&doi=10.2514%2f1.19646&partnerID=40&md5=455ad5a9b6d86273d24ab84b05fdfa52

[33] B. De Vogeleer, “Automatic and fast generation of sub-optimal and feasible low-thrust trajectories using a boundary-value pseudo-spectral method,” Master’s thesis, Delft University of Technology, 2008.

[34] B. Wall, “Shape-based approximation method for low-thrust trajectory optimization,” 2008. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2008-6616

[35] J. Alzubi, A. Nayyar, and A. Kumar, “Machine learning from theory to algorithms: An overview,” in *Second National Conference on Computational Intelligence (NCCI 2018)*, A. George, V. V, A. S. Pandian, and A. C. Donald, Eds., vol. 1142. Institute of Physics Publishing, 2018, Conference Proceedings, cited By :6 Export Date: 26 June 2019.

[36] R. Liu, A. Agrawal, W. K. Liao, A. Choudhary, and Z. Chen, “Pruned search: A machine learning based meta-heuristic approach for constrained continuous optimization,” in *2015 8th International Conference on Contemporary Computing, IC3 2015*, J. Amudha, D. Gupta, J. Zola, N. Nanjangud, A. Pathak, S. K. Prasad, T. Ramesh, M. Parashar, K. Kothapalli, P. Bangalore, S. Chaudhary, and K. V. Dinesha, Eds. Institute of Electrical and Electronics Engineers Inc., 2015, Conference Proceedings, pp. 13–18.

[37] S. Bandaru, A. H. C. Ng, and K. Deb, “Data mining methods for knowledge discovery in multi-objective optimization: Part a - survey,” *Expert Systems with Applications*, vol. 70, pp. 139–159, 2017.

[38] ——, “Data mining methods for knowledge discovery in multi-objective optimization: Part b - new developments and applications,” *Expert Systems with Applications*, vol. 70, pp. 119–138, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417416305474

[39] R. Liu, A. Agrawal, W. Liao, and A. Choudhary, “Search space preprocessing in solving complex optimization problems,” in *2014 IEEE International Conference on Big Data (Big Data)*, 2014, Conference Proceedings, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/ielx7/6973861/7004197/07154118.pdf?tp=&arnumber=7154118&isnumber=7004197&ref=

[40] Y. Sato, K. Izui, T. Yamada, and S. Nishiwaki, “Data mining based on clustering and association rule analysis for knowledge discovery in multiobjective topology optimization,” *Expert Systems with Applications*, vol. 119, pp. 247–261, 2019.

[41] D. R. Myatt, V. M. Becerra, S. J. Nasuto, and J. M. Bishop, “Advanced global optimisation tools for mission analysis and design,” European Space Agency, the Advanced Concepts Team, Report 03-4101a, 2004. [Online]. Available: http://www.esa.int/gsp/ACT/doc/ARI/ARI%20Study%20Report/ACT-RPT-MAD-ARI-03-4101a-GlobalOptimisation-Reading.pdf

[42] C. Hennig, “What are the true clusters?” *Pattern Recognition Letters*, vol. 64, pp. 53–62, 2015.

[43] A. Adolfsson, M. Ackerman, and N. C. Brownstein, “To cluster, or not to cluster: An analysis of clusterability methods,” *Pattern Recognition*, vol. 88, pp. 13–26, 2019. [Online]. Available: https://doi.org/10.1016/j.patcog.2018.10.026

[44] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[45] S. Furao, T. Ogura, and O. Hasegawa, “An enhanced self-organizing incremental neural network for online unsupervised learning,” *Neural Networks*, vol. 20, no. 8, pp. 893–903, 2007.

[46] R. Xu, D. C. Wunsch, and I. C. I. Society, *Clustering*, ser. Ieee Press Series on Computational Intelligence. Hoboken, N.J. Piscataway, NJ: Wiley ; IEEE Press, 2009.

[47] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, “On clustering validation techniques,” *Journal of Intelligent Information Systems*, vol. 17, no. 2, pp. 107–145, 2001.

[48] D. Izzo, M. Ruciński, and F. Biscani, “The generalized island model,” in *Studies in Computational Intelligence*, V. Francisco Fernandez, P. Jose Ignacio Hidalgo, and L. Juan, Eds. Springer-Verlag, 2012, vol. 415, pp. 151–169.

[49] J. Hartigan and P. Hartigan, “The dip test of unimodality,” *Annals of Statistics*, vol. 13, no. 1, pp. 70–84, 1985.

[50] S. Maurus and C. Plant, “Skinny-dip: Clustering in a sea of noise,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-Augu, pp. 1055–1064, 2016.

[51] D. A. Vallado and W. D. MacClain, *Fundamentals of Astrodynamics and Applications*, 2nd ed., ser. Space Technology Library. El Segundo, CA: Microcosm Press, 2001, vol. 12.

A Reference frames and coordinate systems

This appendix treats the reference frame and coordinate sets that were used during the research.

A.1. Inertial reference frame

The reference frame for describing interplanetary transfers used in this thesis is the J2000 frame. It is an inertial reference frame, centred on the Solar System Barycentre (SSB), whose z-axis is aligned with the Earth's rotation axis as of January 1, 2000. The frame is shown in Figure A.1, and we see that its x-axis is pointing towards the vernal equinox ♈. It is almost the same as the widely known ICRF, or International Celestial Reference Frame, and differs only by a negligible amount [51].

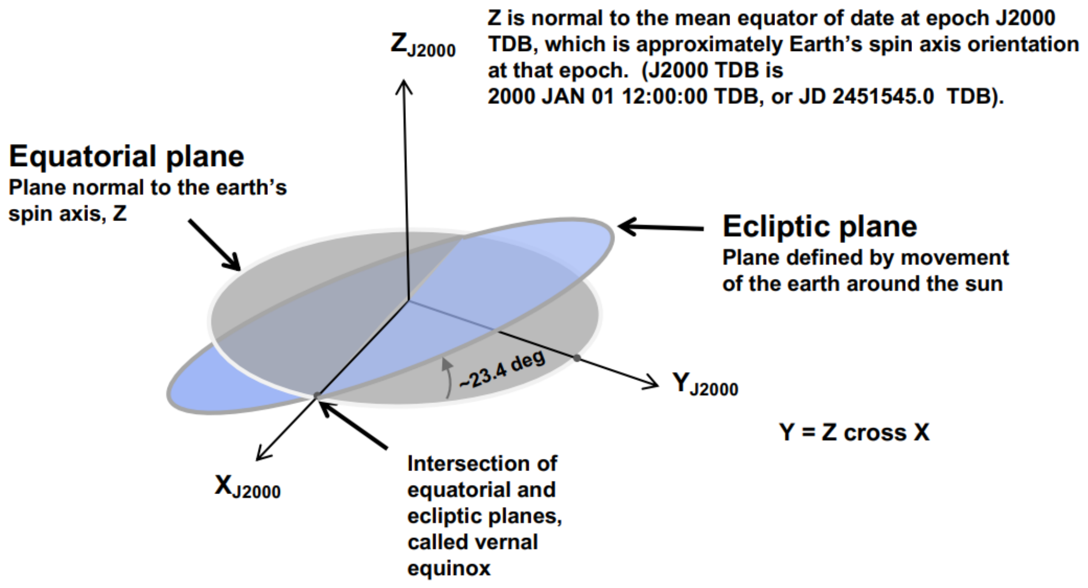
Z_{J2000}
Z is normal to the mean equator of date at epoch J2000 TDB, which is approximately Earth's spin axis orientation at that epoch. (J2000 TDB is 2000 JAN 01 12:00:00 TDB, or JD 2451545.0 TDB).
Equatorial plane
Plane normal to the earth's spin axis, Z
Ecliptic plane
Plane defined by movement of the earth around the sun
~23.4 deg
Y_{J2000}
Y = Z cross X
X_{J2000}
Intersection of equatorial and ecliptic planes, called vernal equinox

Figure A.1: Depiction of the J2000 reference frame, which is nearly equal to the ICRF frame [9]

This frame was used since it is inertial, i.e. it does not exhibit rotation or translational acceleration, so that the equations of motion (see Appendices A.2 and A.3) are devoid of pseudoforces like the Coriolis and centrifugal forces.

A.2. Cartesian coordinates

This coordinate system uses the widely used x, y, and z coordinates to represent positions, velocities, and accelerations; it is also shown in Figure A.2. It is the most straightforward system for describing dynamics, although it might not be the most useful when dealing with rotational motion, such as encountered in astrodynamics. In this frame, the equations of motion (EoM) can be written as:

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3}\mathbf{r} = \frac{\mathbf{T}}{m} = \mathbf{f} \tag{A.2.1}$$

where

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A.2.2}$$

and $||\mathbf{r}|| = r$. μ is the gravitational parameter of the central body, in our case the Sun, and is approximately equal to $1.33 \cdot 10^{20}$ m^{3}s^{-2}.

A.3. Cylindrical coordinates

As mentioned in Section 5.2, the hodographic shaping method makes extensive use of a cylindrical coordinate system. This system uses a radius r, angle θ, and vertical distance z to locate some point $P(r, \theta, z)$ in three-dimensional space. Both r and θ lie in the horizontal xy-plane, as can be seen in Figure A.2. It is also noted that the orthogonal basis vectors $\hat{\mathbf{r}}$, $\hat{\boldsymbol{\theta}}$, and $\hat{\mathbf{z}}$ rotate along with the location of point P, such that $\hat{\boldsymbol{\theta}}$ is always tangential to the imaginary cylinder that touches P.

In this cylindrical basis, the equations of motion are:

$$\begin{aligned} \ddot{r} - r\dot{\theta}^2 + \frac{\mu}{s^3}r &= f_r \\ r\ddot{\theta} + 2\dot{r}\dot{\theta} &= f_\theta \\ \ddot{z} + \frac{\mu}{s^3}z &= f_z \end{aligned} \tag{A.3.1}$$

with

$$s = \sqrt{r^2 + z^2} \tag{A.3.2}$$

In other words, d is the Euclidean distance from the spacecraft to the centre of the reference frame, i.e. the SSB.

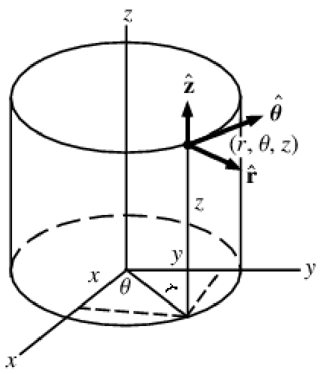

Figure A.2: The Cartesian and cylindrical coordinate systems.

B
Database schema

Main Layout

Table celestial_bodies

* Pk	id	integer
	name	"public".celestialbody
	type	varchar
	spice_name	varchar
Indexes		
Pk	celestial_bodies_pkey	id

Table clusterings

* Pk	id	integer
	run_id	integer
	clustering_algo	"public".clusteringalgorithms
	sil	float8
	db	float8
	ch	float8
	n_clusters	integer
	run_time	float8
	param1	float8
	param2	float8
	param3	float8
Indexes		
Pk	clusterings_pkey	id
Foreign Keys		
	clusterings_run_id_fkey (run_id) ref runs (id)	

Table hodographic-trajectories

* Pk	id	integer
	departure_date	float8
	time_of_flight	float8
	number_of_revolutions	integer
	delta_v	float8
	max_acceleration	float8
	departure_state	bytea
	arrival_state	bytea
	radial_velocity_function_components	_bytea
	normal_velocity_function_components	_bytea
	axial_velocity_function_components	_bytea
	radial_free_coefficients	_float8
	normal_free_coefficients	_float8
	axial_free_coefficients	_float8
	run_id	integer
Indexes		
Pk	hodographic-trajectories_pkey	id
Foreign Keys		
	hodographic-trajectories_run_id_fkey (run_id) ref runs (id)	

Table runs

* Pk	id	integer
	random_seed	integer
	algorithm	"public".pagmoalgorithm
	grid_res_x	integer
	grid_res_y	integer

Table runs

	dep_body_id	integer
	arr_body_id	integer
Indexes		
Pk	runs_pkey	id
Foreign Keys		
	runs_arr_body_id_fkey (arr_body_id) ref celestial_bodies (id)	
	runs_dep_body_id_fkey (dep_body_id) ref celestial_bodies (id)	

Table trajectory_clusterings

* Pk	trajectory_id	integer
* Pk	clustering_id	integer
	cluster_number	smallint
Indexes		
Pk	trajectory_clusterings_pkey	trajectory_id, clustering_id
Foreign Keys		
	trajectory_clusterings_clustering_id_fkey (clustering_id) ref clusterings (id)	
	trajectory_clusterings_trajectory_id_fkey (trajectory_id) ref hodographic-trajectories (id)	

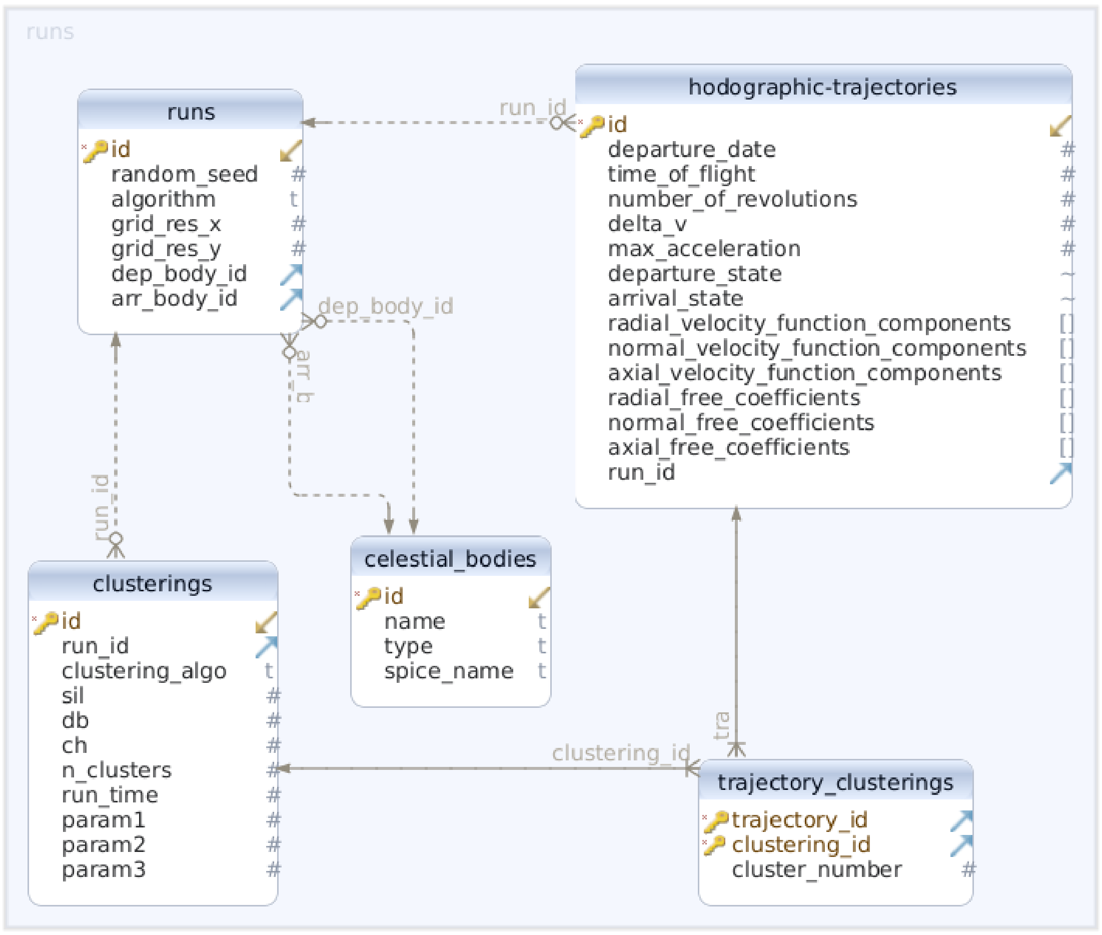

Figure B.1: Database schema used during this thesis.