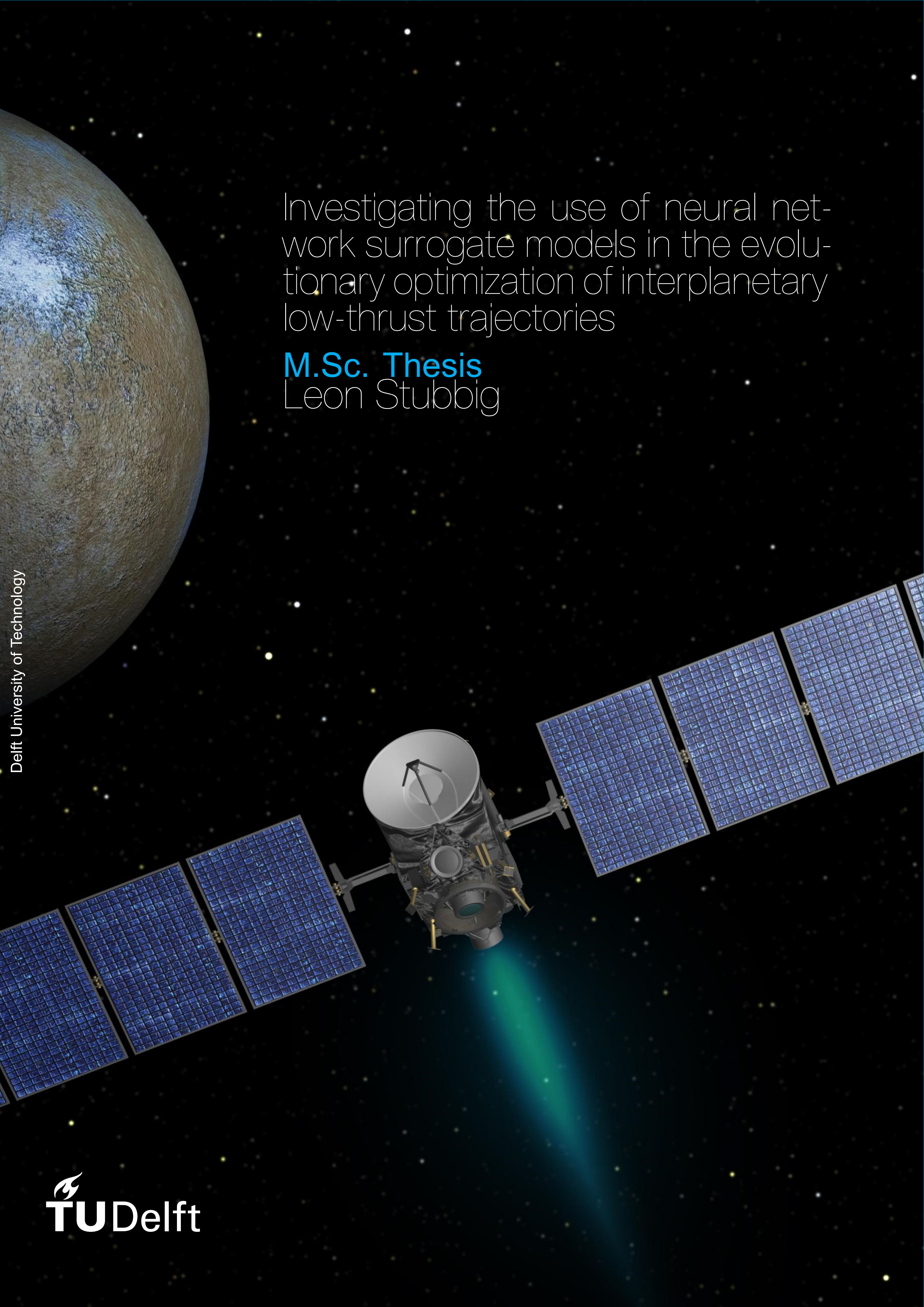


Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories

M.Sc. Thesis

Leon Stubbig

Delft University of Technology

Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories

M.Sc. Thesis

by

Leon Stubbig

to obtain the degree of

Master of Science

at the Astrodynamics and Space Missions Section,
Delft University of Technology,
to be defended publicly on Friday October 18, 2019 at 9:30 am.

Student number:	4721217	
Thesis committee:	Prof.dr.ir. P.N.A.M. Visser,	Committee chair
	K.J. Cowan, MSc MBA,	Supervisor
	Dr. S. Speretta,	External examinator

The cover image is showing an illustration of the Dawn spacecraft in front of the dwarf planet Ceres.
Image credit: NASA/JPL-Caltech

Preface

With handing in this thesis various things come to an end. Not only does it conclude a long research project which, including the Literature Study, has been a large part of my life for the past 11 months, it also ends my time in Delft and my time of being a student. I want to thank a number of people without whom this would not have been possible.

First and foremost, I want to thank my supervisor, Ir. Kevin Cowan. You pointed me towards shaping methods and neural networks, starting off a long process of learning and interesting research. Thank you for supervising me on short notice and taking the time for often longer than planned meetings. I also want to thank the members of the thesis committee, Prof. Pieter Visser and Dr. Stefano Speretta. I hope you are going to enjoy reading the report. The whole space department put together a great master's program, which I will be happy to recommend in the future.

My family deserves a special thank you as well. My parents have supported me continuously: financially, emotionally, and with muscle power when moving around Germany and Europe. These successful studies would not have been possible without you. Thank you, Mira, for being a great sister. I am lucky to have all of you!

Finally, I want to thank my friends in Delft who have made my stay here so much more friendly and enjoyable. Padmini, thank you for being a wonderful person and making my life better. I was lucky to have a great group of friends from Aerospace and beyond, you know who you are. I hope we all manage to stay in touch as much as possible. A shout-out goes to Pablo, who worked on similar topics for his thesis and owes me a crate of beer once Elon Musk lands a rocket on Mars.

Leon Stubbig
Delft, October 2019

Abstract

Building on recent advances in the fields of low-thrust trajectory optimization based on shaping methods, Artificial Neural Networks, and surrogate models in Evolutionary Algorithms, an investigation into a novel optimization routine is conducted. A flexible Python tool to evaluate linked trajectories in a two-body model based on the hodographic shaping method is implemented and used to develop an evolutionary optimization approach where a Genetic Algorithm is assisted in finding new candidate solutions by a surrogate model. This surrogate is constructed from previous fitness function evaluations using Machine Learning, specifically by training an Artificial Neural Network. After deriving suitable (hyper-)parameters for the Genetic Algorithm and the Artificial Neural Network an experimental investigation into the algorithm's performance is conducted with a focus on the design of the surrogate for low-thrust trajectory problems. Two example problems based on the Dawn trajectory and the GTOC2 problem are studied. The surrogate approach is able to find good new candidate solutions, i.e. solutions that improve the population's overall fitness, especially when the surrogate is designed to approximate the shaping computation. Additionally, the use of a surrogate pretrained on a general data set of low-thrust transfers is tested and found to considerably improve the initial quality of the model, meaning that more good candidate solutions are found early on, accelerating the algorithm's convergence.

Contents

Nomenclature

Abbreviations and Acronyms

2D	two-dimensional
3D	three-dimensional
AAS	American Astronautical Society
ABC	Artificial Bee Colony Algorithm
ACT	Advanced Concepts Team
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
AIAA	American Institute of Aeronautics and Astronautics
ANN	Artificial Neural Network
API	Application Programming Interface
BC	boundary condition
Bobyqa	Bound optimization by quadratic approximation
Cobyla	Constrained optimization by linear approximation
CPU	Central Processing Unit
DE	Differential Evolution
DE1220	Self-Adaptive Differential Evolution
DoF	Degree of Freedom
EA	Evolutionary Algorithm
ESA	European Space Agency
GPR	Gaussian Process Regression
GA	Genetic Algorithm
GTOC	Global Trajectory Optimization Competition
GTOC2	The second Global Trajectory Optimization Competition
GPU	Graphics Processing Unit
JPL	Jet Propulsion Laboratory
MAPE	Mean Absolute Percentage Error
MJD2000	Modified Julian Date 2000
ML	Machine Learning
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NLP	Non-linear Programming
NM	Nelder-Mead simplex
PSO	Particle Swarm Optimization
ReLU	Rectified Linear Unit
SOI	Sphere of Influence
SPICE	Spacecraft Planet Instrument Camera-matrix Events (NASA toolkit)
SPK	SPICE kernel file format
SVM	Support Vector Machine
ToF	Time of Flight

Symbols

a	neuron activation	–
b	bias in the ANN	–
c	coefficient	–
C_3	characteristic energy	$km^2\,s^{-2}$
d	flyby distance	m
e	eccentricity	–
$\mathbf{e}$	coordinate system axis	–
f	thrust acceleration	$m\,s^{-1}$
i, k	counters	–
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	unit vectors	–
n	number of base functions	–
N	number of heliocentric revolutions	–
$\mathbf{P}$	position vector	depends on coordinate system
r	radius	m
r, θ, z	cylindrical coordinates	[m, rad, m]
r, φ, θ_1	spherical coordinates	[m, rad, rad]
s	distance to center	m
t	time	s
T	thrust	$m\,s^{-2}$
v	base function	$m\,s^{-1}$
V	velocity magnitude	$m\,s^{-1}$
$\mathbf{V}$	velocity vector	$m\,s^{-1}$
V_∞	hyperbolic excess velocity	$m\,s^{-1}$
w	weight in the ANN	–
x	feature in the ML model	–
$\mathbf{x}$	state vector	depends on coordinate system
x, y, z	Cartesian coordinates	[m, m, m]
β	flyby plane angle	rad
δ	bending angle	rad
ΔV	velocity increment	$m\,s^{-1}$
μ	gravitational parameter	$m^3\,s^{-2}$
σ	activation function	–
ψ	transfer angle	rad
Υ	vernal equinox	–

Subscripts

0	initial
f	final
i	index
in	incoming
j, k	neuron indices
max	maximum
min	minimum
out	outgoing
p	planet
r, θ, z	cylindrical components
total	aggregation of multiple values
x, y, z	Cartesian components
♂	Mars
♀	Venus

Superscripts

l	layer index
$\cdot$	derivative with respect to time
$\cdot\cdot$	second derivative with respect to time
$\sim$	in planetocentric coordinates
$'$	condition after flyby

1 Introduction

Low-thrust propulsion systems have gained increasing popularity in recent years not only for Earth-orbiting satellites, for example, CubeSats [42] and orbit-raising to geostationary orbit [14], but also for interplanetary missions where their increased efficiency enables missions previously not feasible. The first interplanetary application of a low-thrust propulsion system was the Deep Space 1 spacecraft which was launched in 1998 and successfully intercepted asteroid 9969 Braille and comet 19P/Borrelly using solar electric propulsion [35]. Later examples include the European Smart-1 mission that orbited the moon and the Dawn space probe that explored Ceres and Vesta in the asteroid belt.

In comparison to chemical propulsion, the design of low-thrust trajectories is more complex. The spacecraft thrusters have to be operated almost continuously to gain enough ΔV from the low-thrust propulsion system. Thruster burns can therefore no longer be approximated as impulsive shots and the control problem becomes continuous. A solution to this optimization problem requires approaches from optimal control theory. The two most common approaches are indirect and direct methods [3]. Both are local methods that converge to a local optimum close to the initial guess. They are also highly computationally expensive. For example, in 2009 Wall and Conway reported computation times ranging from 2 h to 11 h for the calculation of one optimal trajectory visiting three asteroids in succession using a Non-linear Programming (NLP) solver [49].

In order to be able to globally explore the design space of complex trajectories, a variety of shaping methods have been developed, starting with the exponential sinusoid by Petropoulos and Longuski [32]. These methods flip the problem of propagating a low-thrust trajectory around by prescribing a shape to the trajectory and evaluating the dynamics of the problem a posteriori. They also do not require a precise initial guess but can, in turn, be used to initialize a local search. Analytical expressions for the shaping of either position or velocity are generally preferred which leads to a quick and analytical expression for the control history by evaluating the dynamics in the two-body system, which is generally a reasonable assumption for the preliminary design of interplanetary transfers. A variety of methods has been developed in recent years, e.g. [1, 8, 29–31, 39, 48], that differ in the expression for the shape, the handling of the boundary conditions (BCs) at departure and arrival, and the ability to handle out-of-plane motion and constraints. Here, the hodographic shaping method [13] is employed as it was found to be well suited for the design of 3D, linked trajectories due to its ability to directly satisfy BCs on position, velocity, and Time of Flight (ToF).

With the availability of computationally efficient approximations, it is then possible to develop global optimization approaches that sample wide ranges of the parameter space of multi-phase trajectories. As the governing parameters can be discrete and continuous, and the derivative of the fitness function is generally not available, meta-heuristics have been the most successful approach to tackle these problems. A particular type of meta-heuristic, Evolutionary Algorithms (EAs) including algorithms based on swarm intelligence, has been shown to be particularly successful in interplanetary trajectory optimization [24, 41]. Here, a population of candidate solutions is evaluated with respect to a fitness measure and biology-inspired mechanisms are employed to create the following generation. This evolution-like procedure leads to an overall increase in fitness of the population and has a high chance of converging to a global optimum, even though there is no theoretical guarantee of convergence. In this work, a classical Genetic Algorithm (GA) is employed as a baseline to develop the novel algorithm.

A separate advancement has taken place in recent years in the field of Machine Learning (ML): The availability of large amounts of data and the success of deep Artificial Neural Networks (ANNs) in image recognition [22] have caused a boom in research and applications in the field [23]. Nowadays, ANNs are the best solution to many problems including pattern recognition, natural language processing, and computer vision [27]. Neural Networks have the exciting property that they are, given one hidden layer of enough neurons, capable of approximating any continuous function to an arbitrary accuracy; they are universal approximators [17]. Neural Networks can therefore be very useful in an EA as a surrogate model [19, 20, 52], that uses the data that becomes available from the large number of fitness evaluations to approximate some computationally expensive function and gain information about the problem. The benefit of using a surrogate can either be a speedup in the computation or an accelerated convergence as the approximation smooths the potentially very rugged fitness landscape [2].

1.1. Research question and goal

The overall goal of this work is to employ the efficient design of trajectories using shaping methods and the increasing power of Artificial Neural Networks to build a tool for preliminary trajectory design:

- *To improve the optimization of linked, low-thrust trajectories based on a shaping method by incorporating Artificial Neural Network models into an Evolutionary Algorithm.*

In order to achieve this objective, research questions have been formulated:

- *How can the optimization of linked, low-thrust trajectories based on shaped transfers be improved by Artificial Neural Network models?*

This main question resulted in multiple subquestions to gradually achieve the stated goal:

- *Can a neural network surrogate model be useful in the evolutionary optimization of low-thrust trajectories?*
- *Can an Artificial Neural Network be used to approximate the fitness function in an Evolutionary Algorithm?*
- *How can the surrogate be integrated into an existing Genetic Algorithm?*
- *Is it possible to generate the surrogate fully online?*
- *How can the surrogate be improved based on prior knowledge about the specific problem?*

1.2. Report outline

The main findings of the conducted research are presented in the form of a paper manuscript in Chapter 2. Extended conclusions and recommendations for future work follow in Chapter 3. As the work builds on research from diverse areas, Chapter 4 includes background information on the hodographic shaping method, Artificial Neural Networks, and Genetic Algorithms including the use of surrogates. Appendix A presents the employed frames, coordinates and conversions. The verification of the implemented models and functions is shown in Appendix B, while validation is considered in Chapter C. The thesis is concluded by additional information about the algorithm (average convergence and computational effort), the resulting trajectories (thrust acceleration profiles), and data from the investigated optimization problems (GA settings and problem bounds) in Appendix D.

2 Paper

The paper manuscript is written in the style of the American Astronautical Society (AAS) conference template[1] and included here. It is ready for submission at the 2020 AAS/AIAA Astrodynamics Specialist Conference.

[1]http://www.univelt.com/FAQ.html#submission

(Preprint) AAS XX-XXX

IMPROVING THE EVOLUTIONARY OPTIMIZATION OF INTERPLANETARY LOW-THRUST TRAJECTORIES USING A NEURAL NETWORK SURROGATE MODEL

Leon Stubbig[*] and Kevin Cowan[†]

Building on recent advances in the fields of low-thrust trajectory optimization based on shaping methods, Artificial Neural Networks, and surrogate models in Evolutionary Algorithms, an investigation into a novel optimization routine is conducted. A flexible Python tool to evaluate linked trajectories in a two-body model based on hodographic shaping is implemented and used to develop a novel evolutionary optimization approach where a Genetic Algorithm is assisted in finding new candidate solutions by an online surrogate. The algorithm and different surrogate designs are experimentally investigated on two example problems based on the Dawn trajectory and the GTOC2 problem. Employing the surrogate yields new candidate solutions that improve the population's fitness especially when the surrogate is used to approximate the shaping computation. Additionally, the use of a surrogate pretrained on a general data set of low-thrust transfers is tested and found to considerably improve the initial quality of the model, meaning that more good candidate solutions are found early on, accelerating the algorithm's convergence.

INTRODUCTION

Low-thrust propulsion systems have gained popularity in recent years not only for Earth-orbiting satellites but also for interplanetary missions where their increased efficiency enables missions previously not feasible. Examples include NASA's Dawn mission which concluded in 2018 after a post-launch velocity increment, ΔV, of almost $11\,\mathrm{km\,s^{-1}}$, the highest of any spacecraft flown to date.[1] The design of low-thrust trajectories is more complex than for chemically propelled spacecraft because low-thrust engines have to be operated almost continuously to accumulate the required ΔV. The change in velocity can therefore no longer be approximated as a discrete event and continuous methods mainly derived from optimal control theory have to be employed.[2] As these methods are computationally intensive and require a good initial guess for convergence, so-called shaping methods have been developed specifically for the preliminary design phase. The first shape that was recognized to be a suitable representation of a low-thrust transfer was the exponential sinusoid,[3] followed by a variety of other shapes and methods. Shaping methods efficiently generate transfers by assuming an analytical function that satisfies the applicable boundary conditions and evaluating the dynamics of the problem afterwards, avoiding excessive iterations and computation time. The generated transfers are inherently sub-optimal but are very useful as an initial guess for local optimization and, depending on the chosen shaping function, have been shown to yield realistic thrust profiles well suited for a preliminary exploration of the design space.

The ambitious nature of modern space missions, which often involve multiple flybys, target bodies, and science goals, further complicates the preliminary mission design phase. Multiple trajectory phases have to be linked, greatly increasing the number of mission concepts that have to be considered. The total mission ΔV then depends on each phase being flown as efficiently as possible and every improvement in one phase allows for an increase in the total science return, e.g. by increasing the length of scientific observations at another target. A variety of evolutionary optimization algorithms are successful in finding global optima.[4,5]

[*]M.Sc. Student, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, leon.stubbig@gmail.com
[†]Education fellow + Lecturer, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, k.j.cowan@tudelft.nl

This class of algorithms does not require gradient information and is built around populations of candidate solutions that are evaluated and 'evolved' to create new, more capable solutions. By balancing local and global exploration the chances are high that the global optimum can be found. Here, a classical Genetic Algorithm (GA) is employed to develop the approach, which can be extended to other population-based algorithms in future work.

The goal of this work is to improve the preliminary optimization of linked low-thrust trajectories by including an approximate model in evolutionary optimization. This approach is known in the literature as surrogate modeling.[6] The main idea is to make use of previous fitness evaluations by constructing an approximation of the computationally expensive fitness function. The model can potentially be used at a variety of places in the optimization algorithm, the most straightforward being the replacement of the original, costly fitness function for parts of the evolution. Another possibility is to evaluate the surrogate for new candidate solutions to be fed back into the original population. There are many options to construct the surrogate, ranging from simple linear fits to intricate statistical models. Here, we investigate the use of Artificial Neural Networks (ANNs). In the wake of the recent boom in Machine Learning (ML) ANNs have been heavily developed[7] and represent a promising approach to extract the maximum of information from the available data set.

The paper is structured as follows: First, the astrodynamical model used to represent the individual low-thrust transfers and the approach to building linked trajectories are introduced. Then, the data-driven surrogate approach based on ANNs is presented. The global optimization algorithm is shown in the following section, showcasing several possible applications for the surrogate. Finally, the constructed algorithm is applied to two example problems and conclusions are drawn.

ASTRODYNAMICS MODEL

In this work, the hodographic shaping method[8] is employed to represent interplanetary low-thrust transfers. This particular method was developed to facilitate the rapid, preliminary computation of low-thrust trajectories. In comparison to other shaping methods, it is particularly suited for the development of three-dimensional (3D), linked, trajectories because it allows to satisfy 3D boundary conditions at departure and arrival on position, velocity, and Time of Flight (ToF). It therefore enables the straightforward construction of multi-phase trajectories when combined with functions to evaluate ephemerides and flybys in a linked conics approach, see the example trajectory in Figure 1. We implemented the time-driven version[8] of the shaping method in a software tool for preliminary low-thrust mission design that was written in Python to provide an interface to the available stack of optimization and data science/ML packages. This section presents the implemented astrodynamics models that can be flexibly combined to build linked trajectories.

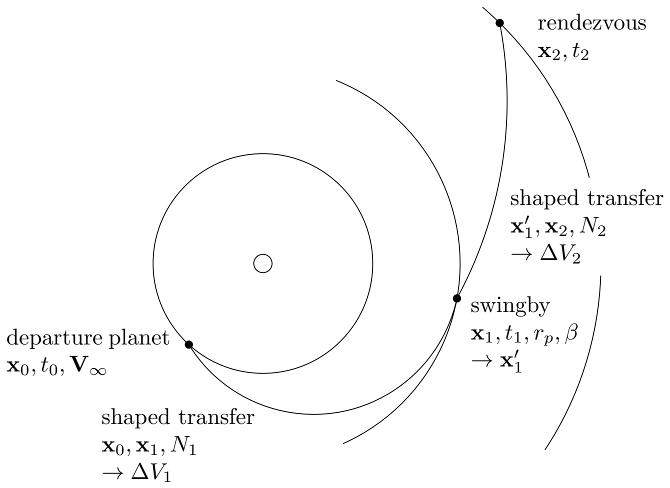

rendezvous
$\mathbf{x}_2, t_2$
shaped transfer
$\mathbf{x}'_1, \mathbf{x}_2, N_2$
$\to \Delta V_2$
swingby
$\mathbf{x}_1, t_1, r_p, \beta$
$\to \mathbf{x}'_1$
departure planet
$\mathbf{x}_0, t_0, \mathbf{V}_\infty$
shaped transfer
$\mathbf{x}_0, \mathbf{x}_1, N_1$
$\to \Delta V_1$

Figure 1. An example trajectory with one flyby and rendezvous

In hodographic shaping, the low-thrust trajectory is represented by three velocity functions in cylindrical coordinates. Each of these velocity functions $V_\square$ describes the velocity's time evolution in this direction ($\mathbf{e}_r$, $\mathbf{e}_\theta$, or $\mathbf{e}_z$) and is assembled by a variable number of analytically integrable and differentiable base functions v_i which are multiplied by shape coefficients c_i:

$$V_\square = \sum_{i=1}^{n} c_i v_i(t), \tag{1}$$

where t is time and n is the number of base functions. Radius r, vertical distance z, and polar angle θ can be computed from the respective base function by analytical integration. Similarly, the respective accelerations are calculated by analytically differentiating the base function. The values of 3 coefficients per velocity function are determined from the boundary conditions on position, velocity, and ToF, while any additional free parameters have to be chosen by an optimization algorithm. Here, we follow the original implementation[8] in using a total of 6 free parameters, i.e. $n = 5$, and the recommended base functions[*]. The hodographic shaping method then computes the ΔV necessary to fly a transfer assuming a two-body system and taking the initial and final state vectors, the number of heliocentric revolutions N, and the ToF as input.

The orbital models are kept simple and follow the capabilities (and limitations) of the chosen shaping method in a linked conics approach. The orbit's initial characteristic energy C_3 provided by the launch vehicle is applied to the departure planet's velocity $\mathbf{V}_\mathrm{p}$ to compute the initial velocity condition of the interplanetary transfer $\mathbf{V}_1$:

$$V_\infty = \sqrt{C_3}, \qquad \mathbf{V}_\infty = V_\infty \begin{pmatrix} \sin\varphi\cos\theta_1 \\ \sin\varphi\sin\theta_1 \\ \cos\varphi \end{pmatrix}, \qquad \mathbf{V}_1 = \mathbf{V}_\mathrm{p} + \mathbf{V}_\infty, \tag{2}$$

where V_∞ is the spacecraft's hyperbolic excess velocity and azimuthal angle φ and polar angle θ_1 denote the impulsive shot's direction in spherical coordinates attached to the spacecraft. As the Sphere of Influence (SOI) is small in comparison to the scale of interplanetary trajectories, its size is assumed to be negligible. Additionally, the launch vehicle's burn time is assumed to be short, leading to the used approximation where any V_∞ is applied instantly as an impulsive shot at the departure position.

Planetary flyby maneuvers are implemented in a similar fashion.[9] The flyby planet's SOI is assumed to be small and the change in velocity is assumed to happen over a short time. To compute the spacecraft's velocity after the flyby its heliocentric velocity in Cartesian coordinates $\mathbf{V}_\mathrm{in}$ at the location of the flyby is projected into the planetocentric reference frame:[10]

$$\tilde{\mathbf{V}}_\mathrm{in} = \mathbf{V}_\mathrm{in} - \mathbf{V}_\mathrm{p}, \tag{3}$$

where the tilde denotes velocities in the Cartesian planetocentric frame. The angle between incoming and outgoing planetocentric velocity, the turn angle δ, is then computed from the eccentricity of the flyby hyperbola e:

$$e = 1 + \frac{r_\mathrm{p}}{\mu_\mathrm{p}} \tilde{\mathbf{V}}_\mathrm{in}^2, \qquad \delta = 2\arcsin\left(\frac{1}{e}\right), \tag{4}$$

where r_p is the periapsis of the flyby hyperbola and μ_p is the planet's gravitational parameter. The planetocentric outgoing velocity $\tilde{V}_\mathrm{out}$ is then computed from the turn angle and the orientation of the flyby plane β, the latter being an input parameter that is measured in the B-plane between the aiming point B, the planet, and a plane parallel to the ecliptic passing through the planet:[10]

$$\tilde{\mathbf{V}}_\mathrm{out} = \tilde{V}_\mathrm{in} \left(\mathbf{i}\cos\delta + \mathbf{j}\cos\beta\sin\delta + \mathbf{k}\sin\beta\sin\delta\right), \tag{5}$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are unit vectors defined as:

$$\mathbf{i} = \frac{\tilde{\mathbf{V}}}{|\tilde{\mathbf{V}}|}, \qquad \mathbf{j} = \frac{\mathbf{i}\times\mathbf{V}_\mathrm{p}}{|\mathbf{i}\times\mathbf{V}_\mathrm{p}|}, \qquad \text{and} \qquad \mathbf{k} = \mathbf{i}\times\mathbf{j}. \tag{6}$$

[*] $V_r = V_\theta =$ C Pow Pow2 PSin05 PCos05 and $V_z =$ CosR5 P3CosR5 P3SinR5 P4CosR5 P4SinR5

Finally, the outgoing velocity is projected back into the heliocentric reference frame:

$$\mathbf{V}_{\text{out}} = \tilde{\mathbf{V}}_{\text{out}} + \mathbf{V}_{\text{p}}, \tag{7}$$

yielding the spacecraft's Cartesian velocity in the heliocentric frame $\mathbf{V}_{\text{out}}$. Powered flybys are not considered as the low-thrust propulsion system is assumed to have a negligible effect on the spacecraft's state during the relatively short duration of the flyby.

These models are complemented by functions to convert positions, velocities, and state vectors between the cylindrical reference frame used during shaping and the Cartesian frame employed for flybys and impulsive shots. Ephemerides are pulled from NASA's planetary data system[11] using ESA's Pykep package[*]. Dwell times at planetary bodies are represented by an *orbit following* behavior. Here, the spacecraft is assigned the same state vector as the body it is visiting while not expending any energy.

MACHINE LEARNING MODEL

During the evolution of a population-based optimization algorithm the fitness function is evaluated many times. In the conventional use case these data points get discarded as soon as a population member is replaced. Some algorithms exist where the use of a history of previous evaluations has been proved to be beneficial. For example, in Particle Swarm Optimization (PSO) each particle keeps track of the best previously visited position in the search space which is then used to guide its path in the following generations. The classical GA, as well as many other optimization algorithms, does not employ such a feature and we manually construct a surrogate model to make use of the data points generated by the continuous evaluation of the fitness function.

As mentioned before, there is a variety of ways to construct a model from data. Previous research on surrogate assisted optimization has mainly been done on polynomial models, Gaussian Process Regression (Kriging), Artificial Neural Networks (ANNs), and Support Vector Machines.[12] After fitting the model to the available data set it provides a means to approximate the fitness function of new solution vectors. In this section, the approach to building this model is described, while the following section showcases the optimization approach and the model's use case therein.

A data set generated by the application of the astrodynamics model described in the previous section will generally consist of a set of parameters describing the trajectory (Departure and arrival dates, V_∞, ToF, flyby parameters) related to figures of merit (ΔV_i, ΔV_{total}, max. thrust). While it is necessary to reduce the size of the input vector as much as possible for the optimization run, i.e. to avoid redundant inputs, this is not automatically true for the ML model. For example, a linked trajectory's timing is efficiently encoded using a departure date and the ToFs of subsequent phases, meaning that the first transfer's arrival date coincides with the second transfer's departure date. The initial and final state vectors are then derived from ephemerides and possibly impulsive shots or the flyby model linking the transfers. The ML model may however benefit from the full state vectors as inputs as the conversion from date to state vector is non-trivial but computationally fast. We therefore experimented with building predictive models for different inputs sets, termed *feature engineering* in ML.

The most straightforward model is trained directly on the optimization's parameter vector. In case of the first example problem, which is derived from the Dawn mission, the parameter vector consists of the launch date, the initial V_∞, the arrival velocity at the flyby planet, the flyby distance and plane angle, the ToF of each transfer as well as the duration of the stay at Vesta (11 real inputs). Evaluating the trajectory, i.e. computing the optimization problem's fitness function, yields a total ΔV necessary to fly this mission. An initial data set to develop the model is created by running a random search using uniformly distributed input parameters from the set bounds. This data set is split into training, validation, and test sets, where training and validation sets are employed during model creation and the model's performance is evaluated on the test set. This approach aims to avoid overfitting to any of the individual data sets and to measure generalization ability on previously unseen data.[13]

[*]D. Izzo, esa/pykep, Version 2.3, 2019, `https://esa.github.io/pykep/`

To assess a model's predictive quality, two figures of merit are computed. The first one is Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{a_i - f_i}{a_i} \right|, \tag{8}$$

where f_i are predictions, a_i are the true values and n is the number of test samples.

The second figure of merit is derived from the models' application: In order to guide a population-based algorithm, the model needs to distinguish good from bad candidate solutions more than accurately predict the correct output value to guide the algorithm towards convergence. Thus, we define a *sorting accuracy* which represents the ratio of samples whose prediction correctly placed them at the top of the test set. In the following, we use a measure of "Top 25 out of 100" meaning that the reported sorting accuracies are the share of the best 25 samples that the predictive model correctly placed in the top 25 out of a total of 100 test samples.

The features in the data set have different ranges and units, necessitating scaling.[13] For example, the departure date is given as a Modified Julian Date 2000, while the flyby plane's angle is given in radians. Scaling is therefore necessary to generate model coefficients of a similar scale, which enables faster learning in ANNs.[13] The two standard options for scaling are Standardization and Normalization: Standardization removes the mean and scales the data to unit variance. This approach presumes that the data follows a normal distribution which cannot be guaranteed here. It was therefore chosen to use the other option of linear normalization to input and output features:

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}(\max - \min) + \min, \tag{9}$$

where $x_{\min}$ and $x_{\max}$ are the per-feature minimum and maximum values, respectively, and $[\min, \max]$ is the range of the scaled data. The input features are scaled to a range of [0, 1], and the output values are scaled to [-1, 1]. The values for $x_{\min}$ and $x_{\max}$ can be automatically extracted from the data set or manually set. The latter was done if the range of input features was known a priori from the bounds of the optimization problem. For cases where the bounds of variables are not known the maximum and minimum values are determined from the data itself.

A linear regression model* is fit to subsets of the initial data set using a least-squares fit and serves as a baseline for more sophisticated models. The performance of the linear regression model is shown in Figure 2. It can be seen that linear regression provides a poor fit, even for a large number of samples. The maximum achieved sorting accuracy is 60 %, while the MAPE of the regression reaches 37 % for a sample size of 2000. Note that the reported samples size is the full data set which was split into training and validation sets at a ratio of 80/20. Regarding the model's performance, it is especially of interest that the model does not improve after increasing the number of training samples beyond 500, clearly indicating that a linear model is not sufficient for the data at hand. The linear relationship is simply not flexible enough to represent the non-linear nature of this problem.

In the following, we therefore investigate the use of Artificial Neural Networks (ANNs) on the given regression problem. ANNs have a long history dating back to the 1950s but received a renewed interest in the past decade when advances in training and initialization algorithms made it possible to train deep networks consisting of multiple hidden layers.[7] Neural Networks now represent the best solution to many problems in ML ranging from pattern recognition to natural language processing, with a variety of specialized flavors and architectures. They have also been used in the area of surrogate modeling,[15] where they are recommended for high-dimensional problems with limited amounts of data.[12] In comparison to other methods, they are very flexible in input and output shapes, size, and computational effort. Here, we rely on a classical fully-connected, feed-forward ANN to perform the non-linear regression task of estimating the necessary ΔV to fly a certain trajectory. A feed-forward ANN consists of an input layer, a variable number of hidden layers, and an output layer. The number of nodes, called neurons, in the input and output layers is determined by the regression problem, while the number of neurons in the hidden layers is flexible. In recent years a number of software

*Normalization and the linear model have been implemented using the scikit-learn package (MinMaxScaler and LinearRegression)[14]

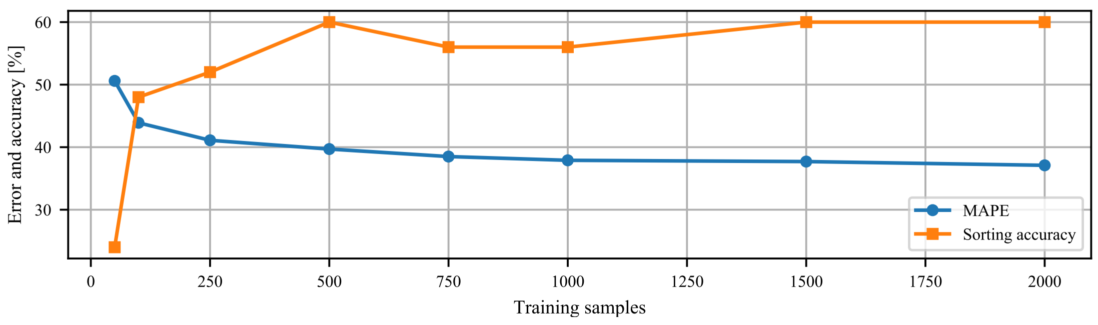


Figure 2. Regression accuracy of a linear regression model

frameworks have been published that allow for a fast implementation of ANNs. Here, we use the Keras API[*] in combination with the Theano backend[†].

In order to define a suitable network architecture, several parameters were set a priori based on literature and preliminary experiments. Firstly, the search for a suitable ANN was restricted to feed-forward neural networks with a low to medium number of layers. As the task is a non-linear regression on a comparatively small, structured data set, there is no need for a specialized deep architecture. Keeping the number of free parameters in the network to a reasonable size also affects the time needed for training and is one of the measures taken against overfitting, which is a common problem especially when training a large network on a limited amount of data. The network's weights are randomly initialized using the uniform Glorot method, where each weight is drawn from a uniform distribution between limits depending on the number of neurons in the connected layers.[16] Each bias is initialized as zero.

The Rectified Linear Unit (ReLU) has been shown to learn faster than other activation functions[7] and is used in all hidden layers. The activation of the output layer is chosen to be linear, in order to not restrict the range of the regression output. The ANN is trained using the Adaptive Moment Estimation (Adam) optimizer.[17] It is used with a Mean Squared Error (MSE) loss, a batch size of 10, and an 80/20 split of training and validation data. To combat overfitting we employ early stopping with a patience of 130 epochs[‡], meaning that training is stopped once the validation error does not improve for more than 130 epochs. The returned model is then the one that exhibited the minimal validation error. The learning rate is reduced periodically employing a similar heuristic: If the validation loss does not improve for more than 50 epochs the learning rate is reduced by 80 %, augmenting local convergence during training. Reducing the learning rate when the loss plateaus also allows to start training at a learning rate of 5×10^{-3}, which is higher than the default of 1×10^{-3} and accelerates the initial learning.

Finally, the relationship between the number of hidden layers, neurons per hidden layer and the size of the training data set is investigated in a grid search. We vary the ANN architecture from 1 to 5 hidden layers of size 16 to 128. The size of the data set is increased from 50 to 1500 samples, showing the performance gain that can be expected when more data becomes available. Each architecture was trained 5 times from random initial states with different seeds and averaged to show performance and training time. Looking at the MAPE and sorting accuracy shown in Figure 3 it is apparent that more data is the single most important factor in increasing the model's performance. In comparison to the linear model, the ANN is able to heavily profit from more data and reaches MAPEs below 15 % for most medium and large architectures. ANNs with 1 and 2 hidden layers perform worse in terms of MAPE while deeper architectures perform fairly similar. The number of hidden layers had a bigger effect on performance than the number of neurons per layer. It was decided to perform most experiments with an ANN of 3 hidden layers with 64 neurons each (3/64). Staying at the lower

[*]F. Chollet et al., Keras, Version 2.2.4, 2019, `https://keras.io`
[†]Theano Development Team, Theano, Version 1.0.3, 2019, `http://deeplearning.net/software/theano/`
[‡]An epoch has passed once the ANN was trained on each training example once.

end of possible network sizes significantly reduces the time needed for training to converge and is therefore beneficial for the setting at hand. For example, using 1000 samples the 3/64 network concluded training on average after 44 s while a bigger 3/128 network needed about 5 times as long (199 s*) to converge. During online use of the surrogate, the ANN was trained from scratch for each new data set. In practice, it is also possible to start with a trained model and only train a few more epochs when new data becomes available. As the training is already performed in batches and new data comes from the same distribution the network then increases its performance. This approach is especially valuable for larger data sets and was employed when a pretrained model was used during the optimization.

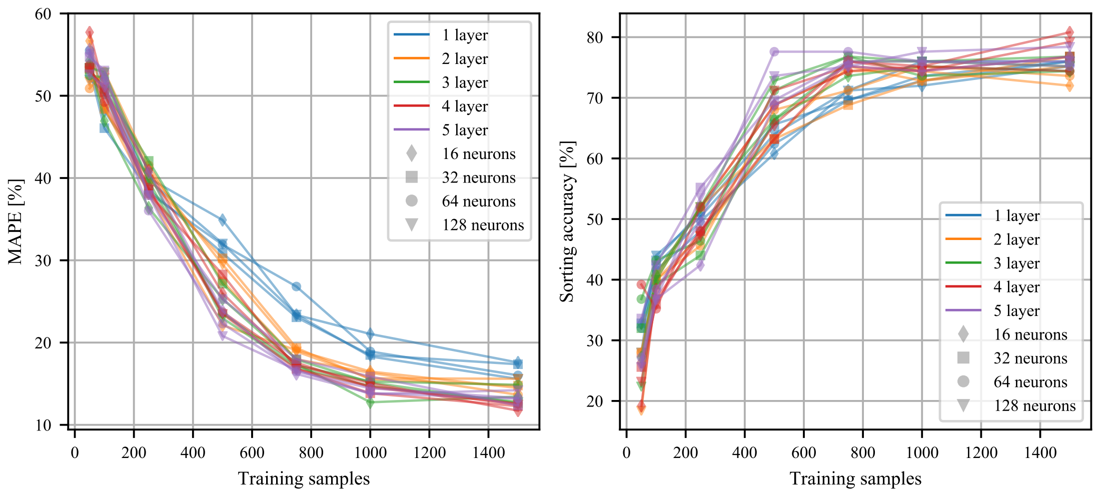


Figure 3. Resulting regression errors and sorting accuracy of the ANN architecture grid search

After choosing the 3/64 architecture, we test the model's robustness with regard to the fixed parameters. This is done by varying batch size and learning rate around the chosen values. The results are reported in terms of regression error and sorting accuracy in Figure 4. It can be seen that the chosen values for the initial learning rate (5×10^{-3}) and batch size (10) perform well especially for sample sizes of 500 and above. A bigger batch size or smaller learning rate would overall reduce the accuracy of the trained model. For smaller data sets a small gain in accuracy could be gained by smaller batch sizes. Here, we stick to one architecture in order to keep the approach consistent.

As shown in the following, it can also be beneficial to train on other features in the data set than the plain candidate solutions. For example, the relationship between the date and a planet's state is non-linear but can be implemented as a fast lookup of tabulated ephemeris data. Training directly on state vectors can therefore simplify the estimation problem, thus reducing the amount of necessary training data, while requiring very little computational effort. This approach is taken when the ANN is used to estimate the cost of a single transfer. Here, the initial and final state vectors in cylindrical coordinates as well as the ToF are chosen as inputs. This way we include the prior knowledge of the main features that determine the cost of a transfer. It is also beneficial that the cylindrical state is nearly periodic over the course of an orbit, meaning the model mainly has to interpolate between known values. The polar angle θ is encoded as $\cos\theta$ and $\sin\theta$, in order to provide two continuous input features. As the number of input/output parameters and the type of problem are very similar we are using the same ANN architecture found above for these related problems as well.

In conclusion, using Artificial Neural Networks to build non-linear regression models is a flexible and performant method that easily outperforms the baseline linear method, c.f. Figures 2 and 3, and is able to quickly improve when more data is available.

*All reported computation times are from a current laptop (CPU: Intel Core i7-7700HQ@2.8 GHz, GPU: Nvidia Quadro M1200)

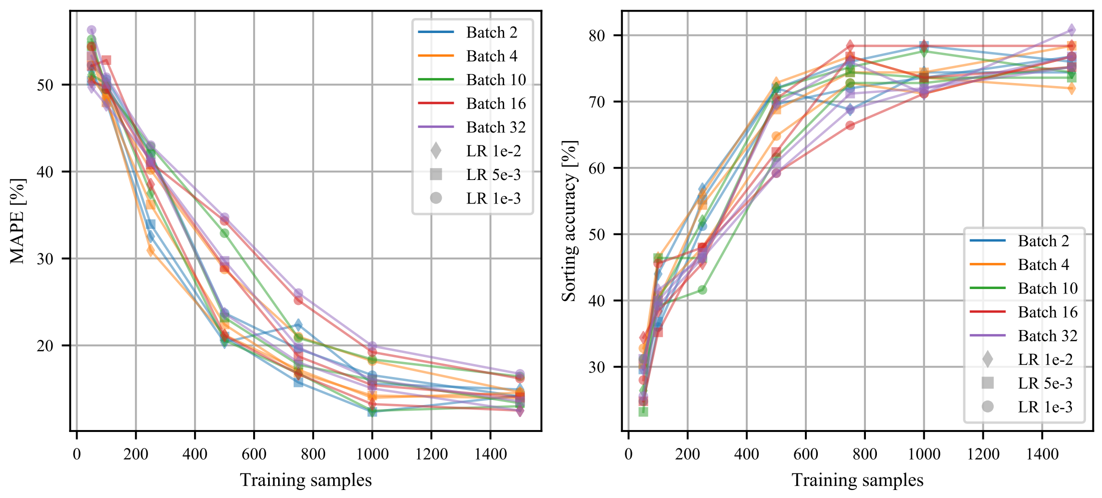


Figure 4. Investigation of the chosen ANN architecture's robustness for variations in learning rate and batch size

OPTIMIZATION APPROACH

The optimization process is structured in two nested loops. The inner loop optimizes the free parameters of the shaping functions defining each individual low-thrust transfer. Here, the Nelder-Mead simplex algorithm was chosen after several algorithms were tested. The outer loop performs the global optimization, i.e. setting the departure date, flight times, and parameters that define the linked trajectory. There, a classical GA was chosen and used to test different ways to include the ML model. Both optimization loops are presented in this section.

Shape Optimization

The shaping method takes the initial and final state vectors that correspond to a given departure date and ToF and N as inputs. It then computes the trajectory's ΔV. As mentioned above, we chose to employ two Degrees of Freedom (DoF) in each velocity function, totaling six free parameters. These parameters are determined by a Nelder-Mead simplex algorithm (NM), following the original paper which investigated the use of the Differential Evolution (DE) and NM algorithms and found that the latter performs favorably.[8] This result was confirmed in the preparations for this work where a variety of global and derivative-free local optimization algorithms were run on a representative example transfer, see Figure 5. The algorithms were taken from the Pygmo[*] and NLopt[†] Python packages and run using the default parameters. The population size for the global algorithms was set to 20. It was found that the problem is well suited for local optimizers, especially as the lowest-order solution represents a suitable initial guess that is always available. Out of the tested local optimizers, see Figure 6, Bobyqa[18] and NM[19] both consistently found the optimum, with Bobyqa converging faster initially but Nelder-Mead simplex reaching the optimum in fewer fitness evaluations. It was therefore decided to stick to the latter to determine the free parameters in the hodographic shaping method.

The NM stopping criterion was set to a relative fitness tolerance of 10^{-3} and a maximum number of fitness evaluations of 10^4. Using an initial guess of 0 for each of the six free parameters, corresponding to the lowest-order solution that already satisfies the boundary conditions, the optimizer converges within 400 to 1000 iterations, not triggering the limit of function evaluations. The optimization of an individual transfer takes about 0.15 s to 0.3 s running on a single thread.

[*]F. Biscani and D. Izzo, esa/pagmo2, Version 2.10, 2019, `https://doi.org/10.5281/zenodo.1045336`
[†]S. Johnson, The NLopt nonlinear optimization package, Version 2.5.0, 2019, `https://github.com/stevengj/nlopt`

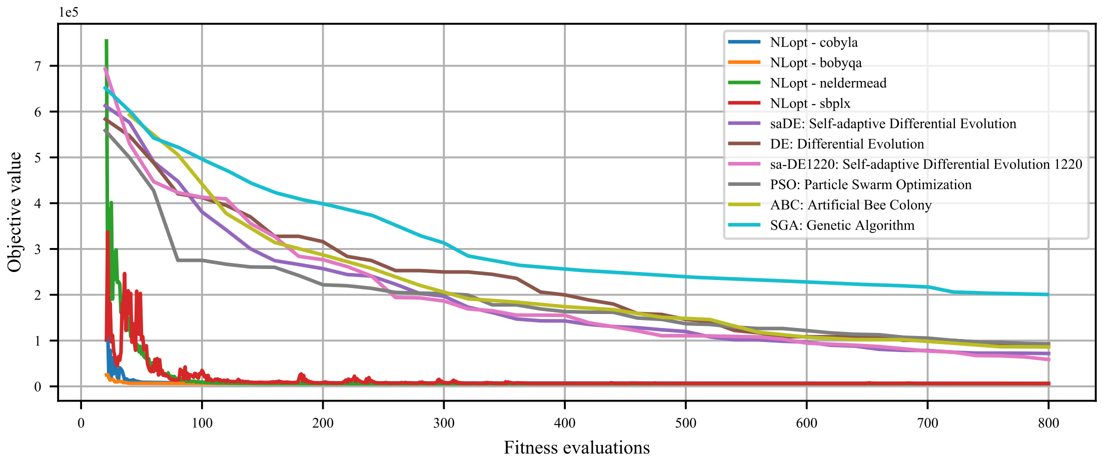


Figure 5. Comparison of the convergence behavior of local and global optimization algorithms on a hodographic shaping problem (average of 10 runs each, Earth-Mars transfer, N = 2, departure date = 9985..10075 mjd2000, ToF = 1010..1100 days, 6 DoF)

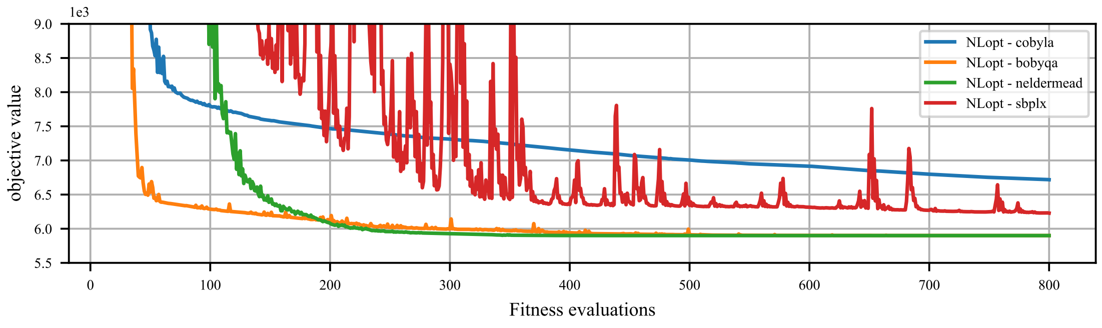


Figure 6. Showing convergence behavior of the local optimizers in detail (average of 10 runs each)

Global Trajectory Optimization

A GA, specifically Pygmo's implementation, was chosen to develop the surrogate assisted optimization tool. As the surrogate is applied on the level of individual fitness function evaluations the approach is also applicable to more advanced algorithms while having a straightforward behavior simplifies the development and serves as a baseline for future extensions. GAs have successfully been applied to the optimization of shaped, low-thrust trajectories for example by Wall and Conway.[20] Extended versions using a surrogate model have not yet been used in the context of shaped trajectory optimization. Pygmo is an optimization library for Python, which was developed by the Advanced Concepts Team at ESA. The provided GA is "rather classical", encoding the parameter vector in a so-called chromosome, and using selection, crossover, and mutation operations in combination with an elitist reinsertion strategy: After the genetic operators have been applied the combined pool of parent and offspring candidate solutions is sorted and only the best are selected to form the new population.

The GA has several control parameters. Similar to the ML model search a number of these have been chosen a priori while other have been explored by running the algorithm on an example problem with different settings: The selection scheme is a 'tournament' with a size of 2. This means that the algorithm chooses the best out of two random parent individuals for reproduction. The crossover method is 'exponential' with a probability of 0.9. In this method, a random point is selected in the parent chromosome and the partner genes are inserted from there with the set probability until it stops. Both of these choices follow the default settings in Pygmo. The mutation scheme was set to ‘uniform’ to increase the population’s diversity in comparison to for example drawing mutations from a Gaussian distribution.

Suitable settings for mutation probability and population size were found by averaging 5 runs of the first example problem, see the Dawn Problem below, each with different settings to show the expected convergence behavior. The results are shown in Figure 7, including a random search for comparison. The random search was set up drawing random values from uniform distributions within the problem bounds to explore the search space and to generate the data set used to develop the model. The number of generations computed for each setting was adapted to compute 3000 evaluations of the fitness function in total. Similar to the GA the shown plot is the average of 5 runs, taking the best individual solution as the optimum which gets updated as the random search progresses. It can be seen that small population sizes, as well as high mutation probabilities, are beneficial for the GA to converge. We therefore use a population size of 10 and a mutation probability of 10 % in the outer loop of the Dawn example problem. The same analysis was done for the second example, the Asteroid problem, leading to a population size of 50 and a mutation probability of 10 %.

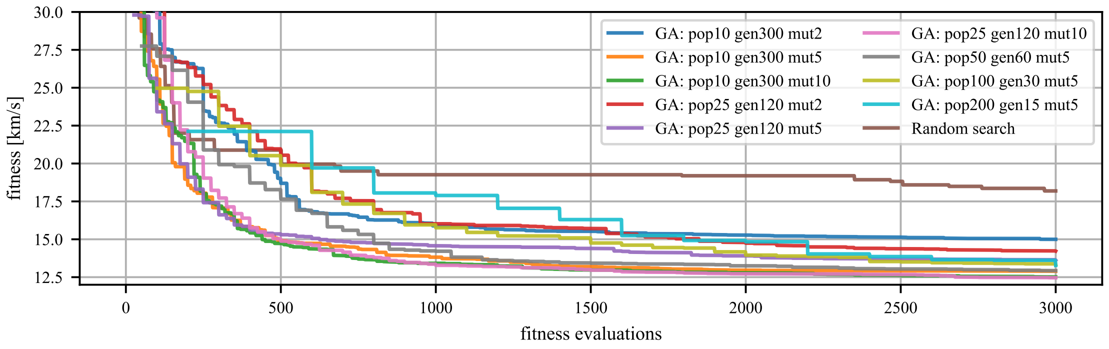


Figure 7. Comparison of different GA settings for population size and mutation probability, average of 5 runs each, optimizing the Dawn Problem

Including the Surrogate, Model Management, and Evolution Control

The GA converges to a good solution by combining individuals from the current generation to create new candidate solutions and keeping a random element in the crossover and mutation steps in order to globally explore the search space. As only better solutions are reinserted into the population the top solution improves over the generations. An interesting idea to improve the search for an optimum is to keep track of previous function evaluations and to build an approximate model using ML techniques, for example, based on the ANN presented earlier.[6,12,21] This approximate model is often called *surrogate* and makes it possible to utilize the information generated from previous function evaluations during the optimization, for example, to speed up convergence or to reduce the computational burden of a large number of fitness function computations. In spacecraft trajectory optimization, the first steps to include a surrogate model have been undertaken in the context of high-thrust missions using DE[22] and orbit transfers with constant thrust.[23]

The successful application of such a surrogate model approximating the fitness function depends on its accuracy to represent the original fitness landscape. The location of local optima is thereby more important than the absolute value of the approximation. Especially the existence of false optima introduced by the surrogate may throw the optimization off track and hinder convergence. It is therefore important to include a strategy for *model management* and *controlled evolution* which mainly includes the regular inclusion of the original fitness function to avoid the convergence towards a false optimum. There are many approaches to manage the inclusion of the surrogate, e.g. replacing the fitness function, filtering candidate solutions, or generating new solution.[6,12,21] Here, we present the approach developed for the given problem of optimizing linked low-thrust trajectories.

One of the challenges of a model that is trained on previous fitness function evaluations is the small size of the data set and the resulting poor initial quality of the surrogate, likely including false optima. During preliminary tests that used both original and surrogate evaluations in parallel, it was observed that a single false optimum in the surrogate was enough to derail the optimization as individuals with unrealistic small fitness values are never reevaluated with the original fitness function. The latter is a wanted behavior of the GA as it saves computational effort over time. We therefore chose a conservative approach to evolution control: The surrogate is trained in regular intervals on the growing data set and searched using a variety of optimization algorithms. Then, the best solutions found during the surrogate search are reevaluated using the original fitness function. Only if a solution is better than the previous GA population it gets reinserted, following the elitist reinsertion strategy of Pygmo's GA implementation. Then the GA continues with the original evolution, increasing the size of the data set and converging towards the true optimum. After a set number of generations, a *sequence*, a new ANN is trained and searched, potentially resulting in new candidate solutions.

As the evaluation of the surrogate is computationally cheap (about 120 µs per fitness evaluation vs. about 2.3 s for a fitness evaluation with 3 transfers), a total of 4 local (Cobyla, Bobyqa, Nelder-Mead, and Subplex) and 6 global optimization algorithms (GA, DE, DE1220, Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC)) are employed during the surrogate search to find new optima in the approximate model. Each local algorithm is run 3 times with different, random starting points in a multistart approach, totaling 17 different optimization runs. Each run is performed with default parameters which can be found in the Pygmo and NLopt documentations. The local algorithms are run until convergence with an absolute tolerance of 10^{-2} or a maximum of 5000 fitness evaluations. The global algorithms are run with a population size of 20 for 200 generations except for ABC which uses more fitness evaluations in the scouting phase[24] and is run for 100 generations. In terms of computational effort, one original fitness evaluation takes as long as running 3 local or 2 global optimizers using the surrogate. The total effort of searching for new candidate solutions utilizing the surrogate is therefore about 7 original fitness evaluations plus the effort of reevaluating the resulting candidates using the original fitness function.

The initial lack of data was identified as a major problem, see the Results section below, which is why we also investigated the use of a pretrained surrogate. Here, the surrogate is trained on a general data set containing precomputed low-thrust transfers. The more general, i.e. unrelated to the optimized trajectory, this data set is, the more easily it can be used for a variety of problems and does not have to be recomputed for each specific mission design problem. Pretraining improves the model's initial quality immensely but includes the additional effort of computing a general data set and pretraining the ANN. During the optimization the model is then only incrementally trained, meaning that it is not replaced but allowed to adapt to the region of interest by executing some optimization epochs on the new data. We observe incremental training times of 5 s to 60 s on the CPU, while pretraining a new model takes is a larger computational effort that is better suited for the GPU. Pretraining took in the order of 10 min for the two presented case studies. To avoid forgetting of previously seen data this *incremental* training is started at a lower initial learning rate of 10^{-4}. The learning rate is reduced on the detection of a validation loss plateau and training is stopped early as before. Both surrogate assisted optimization approaches, the fully online generation of the surrogate and the optional use of a pretrained model, are shown in Figure 8.

We are exploring three different points of approximation for the surrogate: The first approach is to replace the full fitness function evaluation by the surrogate, with the ML model being trained directly on the candidate solution vectors of the previous generations. This approach, termed *full surrogate*, is the most general and readily translates to other problems. We also test two other strategies that are adapted to the problem at hand in order to incorporate prior knowledge about problems concerning interplanetary transfers. The focus is therefore put on the computationally expensive part of the fitness function, i.e. the inner ΔV computation of individual low-thrust legs of the linked trajectory. The first approach is to replace the general shape optimization by an ML model, i.e. to train a *general transfer surrogate*, the other is to replace each transfer by a separate model, termed *individual transfer surrogate*. The advantage of the first one being the availability of more online training data, while a model estimating individual transfers has to learn a problem that is valid for a smaller input parameter space, therefore reducing the approximation problem's difficulty. An extension on the latter approach is the use of a model that is pretrained on a general set of similar transfers, which is also investigated.

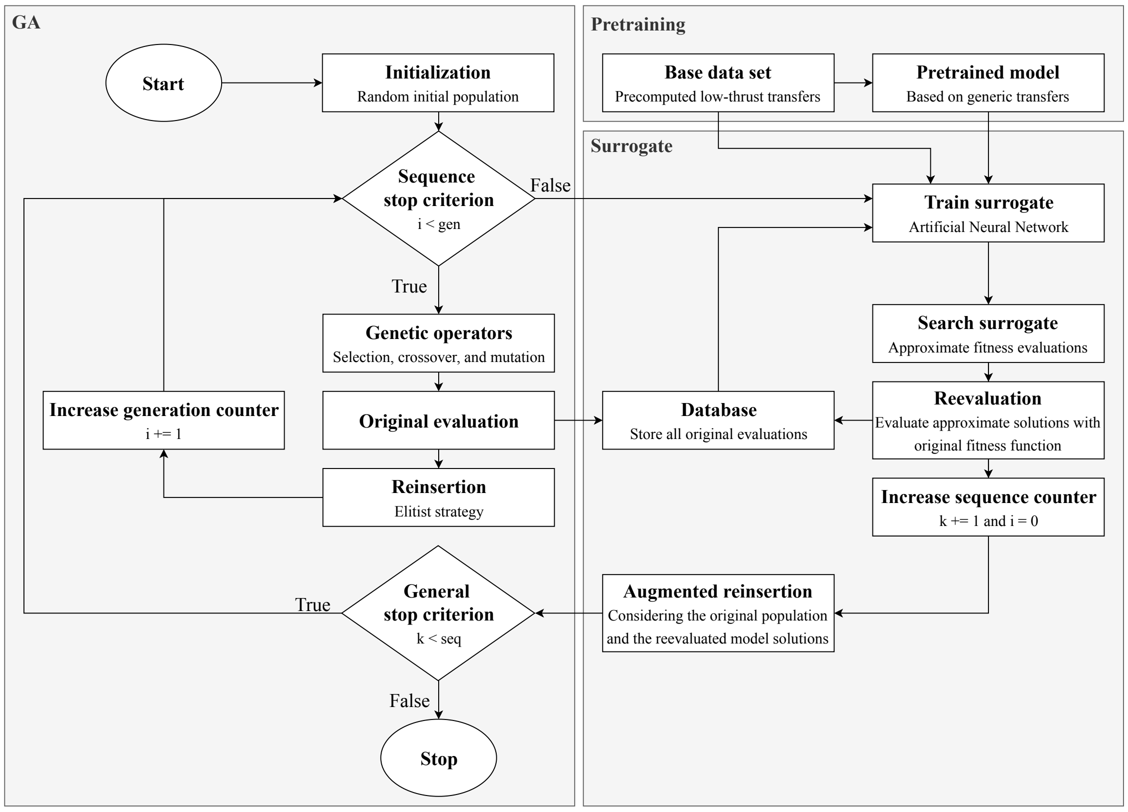

GA
Pretraining
Surrogate
Start
Initialization
Random initial population
Base data set
Precomputed low-thrust transfers
Pretrained model
Based on generic transfers
Sequence stop criterion
i < gen
False
Train surrogate
Artificial Neural Network
True
Genetic operators
Selection, crossover, and mutation
Search surrogate
Approximate fitness evaluations
Increase generation counter
i += 1
Original evaluation
Database
Store all original evaluations
Reevaluation
Evaluate approximate solutions with original fitness function
Reinsertion
Elitist strategy
Increase sequence counter
k += 1 and i = 0
True
General stop criterion
k < seq
Augmented reinsertion
Considering the original population and the reevaluated model solutions
False
Stop

Figure 8. The augmented GA using an ANN surrogate to find new candidate solutions

RESULTS

The presented approach is applied to two example problems that utilize low-thrust propulsion to explore the solar system. The first one is derived from the Dawn mission and involves a launch at Earth, a flyby at Mars, a stay at Vesta, and a transfer to Ceres. The second example is a simplified version of the problem posed for the second Global Trajectory Optimization Competition (GTOC2). It consists of four consecutive visits at different asteroids with a minimum stay time. Both problems and the results are presented here.

The Dawn Problem

The first example problem is inspired by NASA's Dawn mission which explored (4) Vesta and (1) Ceres in the asteroid belt. The spacecraft was launched in September 2007 and left Earth's SOI with a C_3 of $11.3\,\text{km}^2\,\text{s}^{-2}$.[25] Propulsion was achieved with three xenon ion engines with a specific impulse of 3100 s producing a maximum thrust force of 91 mN. With the spacecraft's launch mass of 1300 kg this translates to an initial maximum thrust acceleration at of $7 \times 10^{-5}\,\text{m}\,\text{s}^{-1}$ per thruster. Dawn's trajectory was designed with a duty cycle of 95 %, meaning that the continuous thrust imposed in the shaping model is a reasonable approximation. The mission's trajectory, including thrust (blue) and coast periods (black), is presented in Figure 9. After launch, the spacecraft used its low-thrust propulsion system to reach Mars, where an unpowered flyby was performed in February 2009 yielding a velocity change of $2.6\,\text{km}\,\text{s}^{-1}$.[26] The low-thrust transfer continues and Vesta is reached for a rendezvous in July 2011. The science phase in orbit around Vesta lasted one year until Dawn performed a transfer to Ceres which lasted from July 2012 to February 2015. Table 1 lists the flight and stay times as well as the boundary conditions defining the trajectory phases. In total the spacecraft achieved a ΔV of $11\,\text{km}\,\text{s}^{-1}$, including the interplanetary transfers and proximity operations.

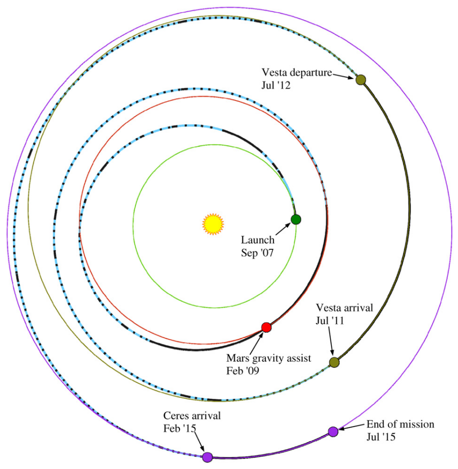


Figure 9. The trajectory of the Dawn mission[26]

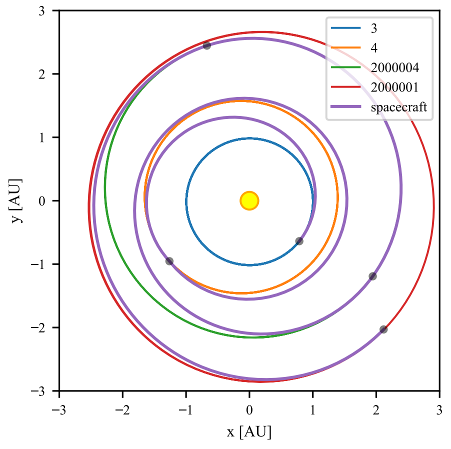


Figure 10. Optimized Dawn problem

Table 1. The Dawn mission: Phases and dates[26]

	Transfer 1	Transfer 2	Stay 1	Transfer 3
Departure body	Earth	Mars	Vesta	Vesta
Arrival body	Mars	Vesta	Vesta	Ceres
Departure condition	Orbit injection	Flyby	Orbit following	Rendezvous
Arrival condition	Flyby	Rendezvous	Orbit following	Rendezvous
Revolutions [ ]	0	1	–	0
Departure date [calendar date]	01 Sep 2007	12 Feb 2009	05 Sep 2011	01 Jul 2012
Departure date [mjd2000]	2800	3330	4265	4565
ToF [days]	530	935	300	945

An optimization problem was derived from Dawn's trajectory. Its free parameters and the corresponding bounds are presented in Table 2. Corresponding to the Dawn mission N was fixed to 0, 1, and 0 for transfers 1, 2, and 3, respectively.

As shown before, see Figure 7, the GA converges well on this problem. The overall best solution of $11.8\,\mathrm{km\,s^{-1}}$ was found during one of the runs with a population size of 10 and a mutation probability of 5 % after 2860 fitness function evaluations. All 5 runs reached fitness values below $15\,\mathrm{km\,s^{-1}}$ within 1400 fitness evaluations. The resulting trajectory is shown in Figure 10 and the corresponding parameter values are included in Table 2. It can be seen that the optimizer converges to a slightly earlier launch date. ToF 1 and 3 are chosen shorter while ToF 2 and the stay time at Vesta are longer in the optimized solution than the original trajectory. ToF 2 is close to the imposed upper boundary which was kept as is to not overly increase the total flight time. The maximum thrust acceleration is $2 \times 10^{-4}\,\mathrm{m\,s^{-2}}$, which is 70 % of the initial thrust acceleration of two Dawn ion engines. The thrust is therefore achievable by current technology without having imposed any upper limit during the optimization. The total ΔV is in the same order of magnitude as the original mission with Transfers 1, 2, and 3 contributing $2.46\,\mathrm{km\,s^{-1}}$, $5.76\,\mathrm{km\,s^{-1}}$, and $3.57\,\mathrm{km\,s^{-1}}$, respectively. Of course, this is only a rough comparison as the real mission had to deal with many more constraints, a full dynamical environment, and also included the science operations at both bodies. Overall we conclude that the optimization approach based on a linked conics approximation and hodographically shaped low-thrust transfers is able to capture the character of the original trajectory while being rapidly computed, and therefore represents a suitable approach to preliminary low-thrust trajectory optimization.

Full surrogate As a first augmentation approach, the surrogate is trained solely on data from the GA. One sequence was chosen to be 200 fitness evaluations, split into the evolution of a population of 10 individuals

Table 2. Optimization parameters of the Dawn Problem

Parameter	Unit	Lower bound	Upper bound	Best GA result
Departure date	mjd2000	2500	3500	2782.2
ToF 1	days	300	700	407.6
ToF 2	days	700	1100	1099.1
Stay 1	days	200	600	559.2
ToF 3	days	700	1100	912.7
Orbit insertion V_∞	$\mathrm{m\,s^{-1}}$	1.5×10^3	5×10^3	2.42×10^3
Mars arrival velocity (radial)	$\mathrm{m\,s^{-1}}$	-5×10^3	5×10^3	-2.21×10^3
Mars arrival velocity (tang.)	$\mathrm{m\,s^{-1}}$	1×10^4	5×10^4	2.17×10^4
Mars arrival velocity (vert.)	$\mathrm{m\,s^{-1}}$	-5×10^3	5×10^3	-6.37×10^1
Flyby radius r_p	m	$r_♂ + 1.5 \times 10^5$	$r_♂ + 1 \times 10^7$	$r_♂ + 4.71 \times 10^5$
Flyby plane rotation angle β	rad	0	2π	2.71

for 20 generations. The mutation probability was set to 10 %. Figure 11 shows the evolution of 5 runs. Here, the dots indicate the minimum value found during the surrogate utilization (training, finding new candidate solutions, reevaluating using the original fitness function). It can be seen that the reevaluated candidate solutions are never better than the current champion in the GA population. In fact, during these 5 runs the best model solution was never good enough to be reinserted into the original population. This means that the GA evolution is unaffected by the use of the surrogate and would converge similarly without any augmentation. We attribute this mainly to the lack of available data. The 200 data points available after the first sequence are not enough to construct a useful approximation of the fitness function and, while the model improves as more data becomes available, it never catches up with the improvements found by the GA. The 11-dimensional fitness function is highly non-linear, especially due to the Mars flyby. There are also interdependencies between the variables as a longer ToF in one phase shifts the departure and arrival dates of later phases, making it a difficult problem to learn on limited data.

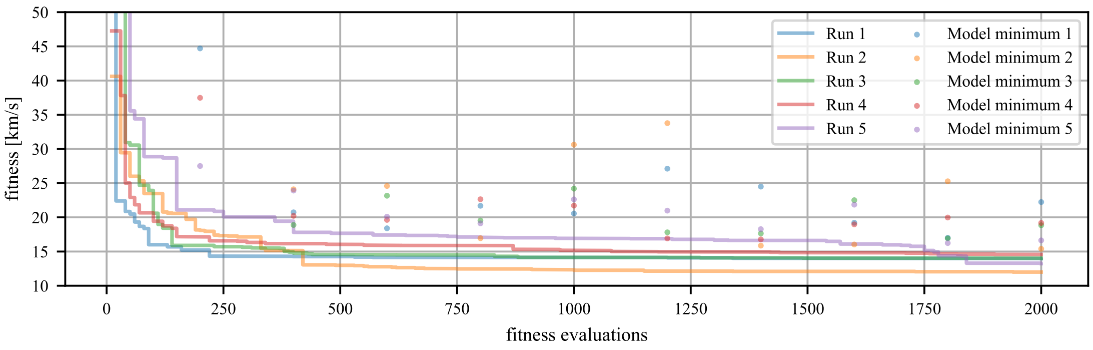


Figure 11. Full surrogate applied to the Dawn problem (Pop. 10, Gen. 20, Seq. 10)

General transfer surrogate The case is similar for the *general transfer surrogate*, which is trained on the inputs to the shaping method and estimates the cost of the low-thrust transfers with one model. The convergence of 5 runs is shown in Figure 12. The number of reinsertions into the original populations are indicated with a number next to the found minimum. Here, this occurred early during run 5 where the surrogate search resulted in an improvement of the global optimum. The candidate solutions found using the surrogate are better, consistently finding solutions below 25 $\mathrm{km\,s^{-1}}$ starting with the second sequence, but are generally not good enough to increase the fitness of the original population. The average MAPE and sorting accuracy of the final model are 23.1 % and 67.2 %, respectively.

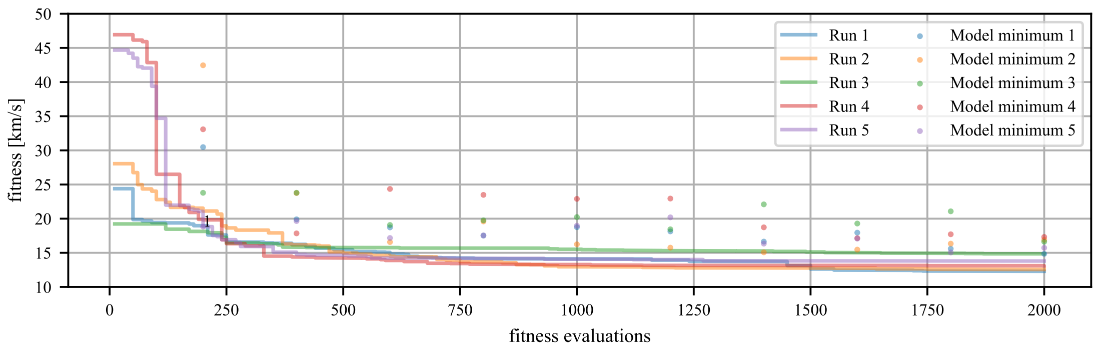


Figure 12. General transfer surrogate applied to the Dawn problem (Pop. 10, Gen. 20. Seq. 10)

Individual transfer surrogate A separate model is trained for each transfer, reducing the problem complexity. Especially the estimation for Transfer 3 improves, as the rendezvous conditions at departure and arrival simplify the approximated function. The effect is smaller for Transfers 1 and 2, which have more diverse boundary conditions due to orbit injection and the Mars flyby. The models' average final MAPE are 32.3 %, 11.2 %, and 2.4 % for Transfers 1, 2, and 3, respectively. The corresponding sorting accuracies reach 75.2 %, 89.6 %, and 96.0 %. The algorithm's convergence is shown in Figure 13. Occasional reinsertions happen, but the general effect on the GA is small.

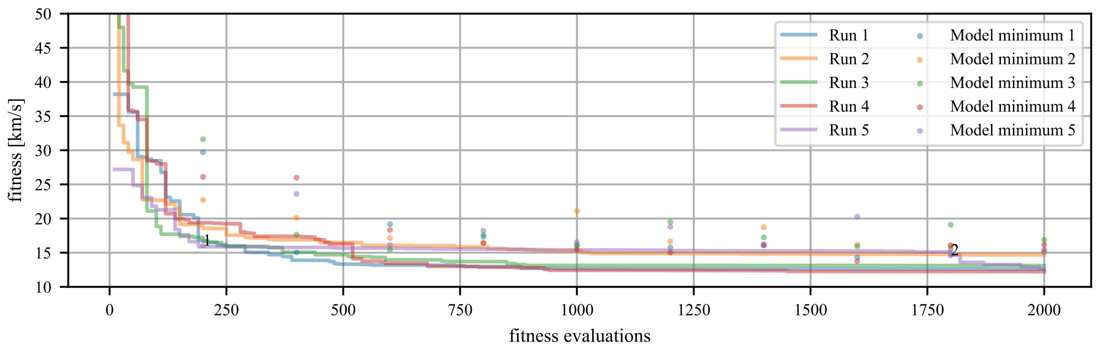


Figure 13. Individual transfer surrogate applied to the Dawn problem (Pop. 10, Gen. 20, Seq. 10)

Pretrained individual transfer surrogate Here, the initial quality of the surrogates is improved by pretraining the ANN on a data set comprised of transfers from Earth to Mars, Mars to Vesta, and Vesta to Ceres. We used the data from the random searches that were used to develop the ANN model. This data set was deemed to be general enough for a proof of concept as the range for the departure date comprises more than two synodic periods between Earth and Mars with the later epochs being additionally shifted by the ranges in the ToF and stay time at Vesta. As the data set is now much larger than during the online cases and the computational effort of pretraining the model is detached from the actual optimization run, the ANN was slightly increased in size to an architecture of 3/124. The used data set has a size of 15 000 trajectories of which 100 are used for the computation of MAPE and sorting accuracy. The remaining trajectories are split into training and validation sets at a ratio of 90/10. Based on this much larger data set the regression MAPE of the pretrained surrogate is greatly improved in comparison to the online generated variants: It reaches 3.5 %, 2.2 %, and 0.1 % for models 1, 2, and 3, respectively. The sorting accuracy reaches 100 % for all three models.

The algorithm was then run with a population size of 10 and 5 generations per sequence, see Figure 14. It can be seen that the surrogate improves the champions especially after the first sequence where all runs receive a new champion. Note that the minimum found during the random search was 16.6 km s^{-1}, meaning that the surrogate evaluation yields better results than a linear interpolation would. The surrogate also keeps improving due to the incremental training between sequences and more reinsertions keep improving the fitness of the original population after the following sequences. Furthermore, all runs converge to good optima below 12.5 km s^{-1} which was not the case during the runs of the original GA or the full surrogate approach.

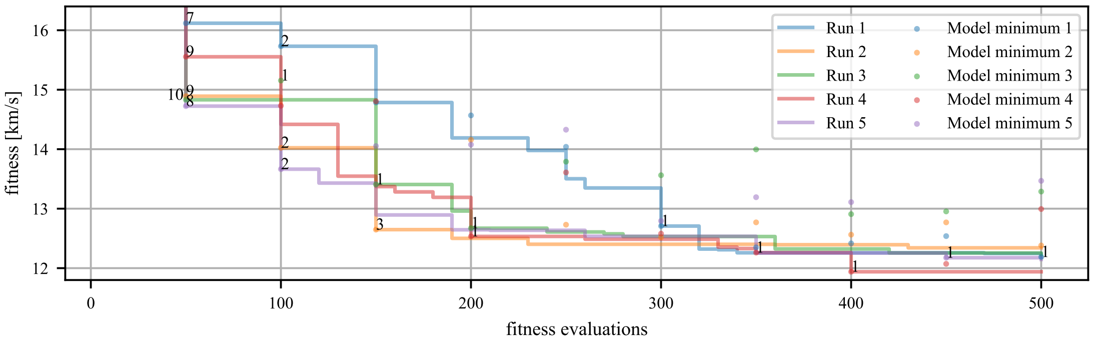


Figure 14. Pretrained individual transfer surrogate applied to the Dawn problem (Pop. 10, Gen. 5, Seq. 10)

The Asteroid Problem

As a second example, we apply the presented optimization to a multiple asteroid rendezvous mission, derived from the problem created by JPL for GTOC2[*]. The problem was to visit one asteroid out of 4 groups with a low-thrust mission in a given time-frame. The asteroid orbits are shown in Figure 15. All competition solutions visited an asteroid from Group 4 first and then went outwards to Groups 3, 2, and 1. We disregard the outer problem of finding a suitable sequence of asteroids to visit and concentrate on finding a low ΔV trajectory for given sequences. We also optimize directly for ΔV and not for the original fitness function which was the final mass over total ToF.

Previous work applied the spherical shaping method[27] to the winning trajectories of GTOC2,[28] keeping launch dates as well as ToFs and stay times constant. V_∞ was set to zero for the shaped solutions. Recreating these trajectories using the hodographic shaping method yields the results listed in Table 3 and shown in Figure 16 for the winning solution. It can be seen that the hodographic shaping method performs significantly better than the spherical method while not reaching the quality of the fully optimized winning solutions.

Table 3. Fixed solutions of the Asteroid problem

Winners[28]	N	ΔV_{total}	Spherical[28]	ΔV_{total}	Hodographic	ΔV_{total}
GTOC 1st	1, 2, 0, 0	20.1 km s^{-1}	Shaping 1	41.78 km s^{-1}	Shaping 1	39.56 km s^{-1}
GTOC 2nd	0, 3, 0, 0	19.4 km s^{-1}	Shaping 2	49.49 km s^{-1}	Shaping 2	27.78 km s^{-1}
GTOC 3rd	1, 2, 0, 0	23.3 km s^{-1}	Shaping 3	46.08 km s^{-1}	Shaping 3	33.81 km s^{-1}

Table 4 lists the parameters of the example problem derived from the GTOC2 setting. The asteroid sequence[†] as well as the N of each transfer (0, 0, 0, 0) are fixed and the optimizer is tasked to find optimal values for

[*] A. Petropoulos, Problem Description for the 2nd Global Trajectory Optimization Competition, 2006

[†] SPK IDs: 3258076, 2000060, 2000058, 2002959; Pykep indices: 815, 300, 110, 47

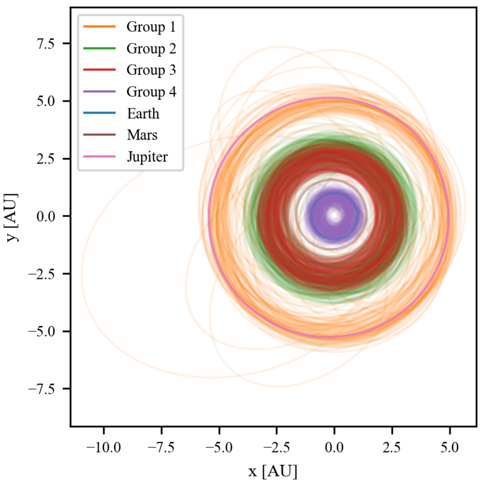


Figure 15. The GTOC2 asteroids

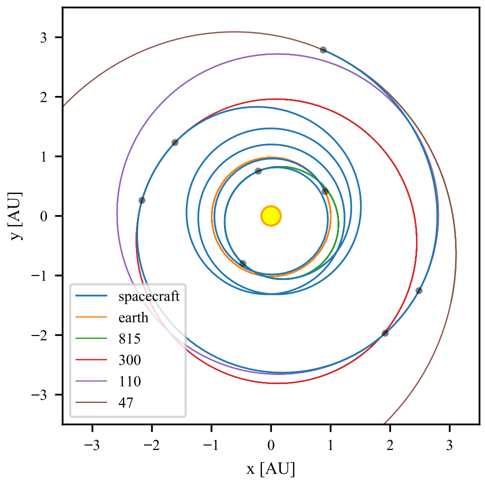


Figure 16. The recreated 1st trajectory

the departure date, flight and, stay times as well as magnitude and direction of the injection impulsive shot. The bounds for the departure window are set to 10 years. The bounds for ToFs are kept wide, shifting towards longer durations for later transfers, while the stay times are kept close to the minimum time of 90 days set in the competition. The parameters of the best solution found, see Table 4, yielded a total ΔV of 26.9 km s^{-1}.

Table 4. Optimization parameters of the Asteroids problem

Parameter	Unit	Lower Bound	Upper bound	Best GA result
Departure Date	mjd2000	5479	9129	8311.5
ToF 1	days	100	1700	106.8
Stay 1	days	90	110	90.0
ToF 2	days	200	1900	517.1
Stay 2	days	90	110	102.1
ToF 3	days	400	1900	1299.5
Stay 3	days	90	110	90.0
ToF 4	days	600	2000	1096.6
Injection magnitude V_∞	m s^{-1}	0	3.5×10^3	3.22×10^3
Injection angle φ	rad	0	π	2.19
Injection angle θ_1	rad	0	2π	2.91

Full surrogate The surrogate assisted optimizations are again conducted with sequences of 200 fitness function evaluations. The case is similar to the application of the full surrogate optimization to the Dawn problem: Even though the surrogate's quality improves as more data becomes available, it not high enough to assist the optimization, see Figure 17.

General transfer surrogate More reinsertions are observed when the surrogate is trained to approximate the cost of the low-thrust transfer, see Figure 18. This is partially due to the larger population size resulting in a higher chance for medium quality solutions to be better than some member of the GA population. But we also observe a number of very useful reinsertions where the reinsertion leads to a new champion, e.g. after each of the first 3 sequences of Run 1. Another aspect that leads to better results here in comparison to the Dawn case is that the transfers are more homogeneous: Only one departure includes an impulsive shot and there are no flybys. Additionally, there are four transfers per trajectory, resulting in more and better training data.

Individual transfer surrogate The results for the third variation of surrogate inclusion are shown in Figure 19. Similar to the general transfer surrogate useful candidate solutions are found in the surrogate especially in

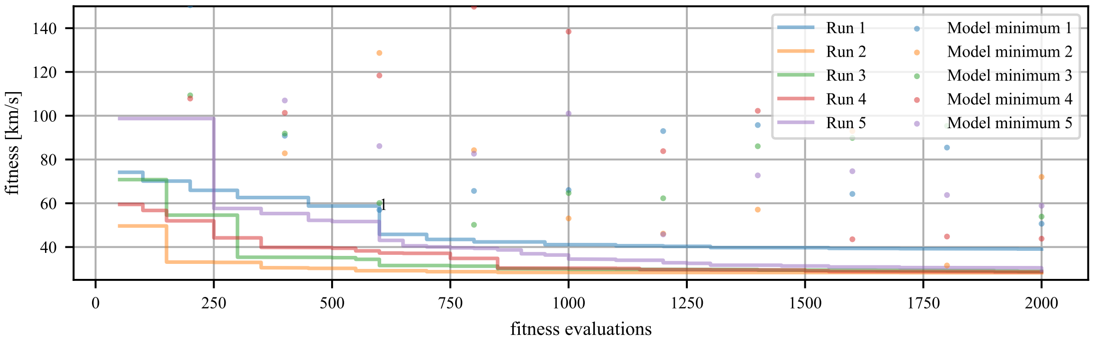


Figure 17. Full surrogate applied to the Asteroid problem (Pop. 50, Gen. 4, Seq. 10)

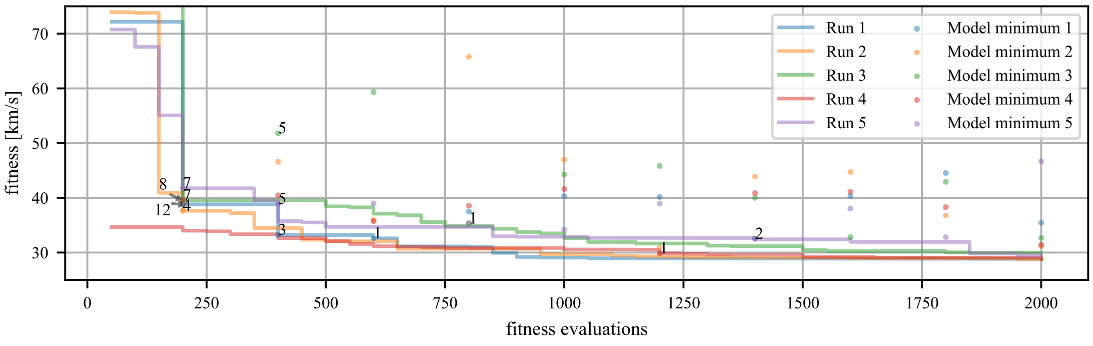


Figure 18. General transfer surrogate applied to the Asteroid problem (Pop. 50, Gen. 4, Seq. 10)

the beginning. But, reinsertions happen more slowly than for the general transfer surrogate and the overall algorithm does not reliably converge below 34 km s^{-1}. The suspected reason is again the reduced initial quality of the model due to less training data as there are now four models, each trained on samples from one transfer.

Pretrained general transfer surrogate Following the previous results, the pretrained approach is applied to the general transfer surrogate. Here, a new data set is created based on shaped transfers between separate from the predefined sequence in the optimization problem. Transfers are computed between random bodies from Earth to Group 4, Group 4 to 3, 3 to 2, and 2 to 1, with 110 000 samples per group resulting in a data set of 440 000 samples. The departure date is set to a range of [5479, 12053], representing the first 18 years of the competition window, and the ToF is varied in large windows of [50, 600] days for the first two transfers and [300, 1500] and [400, 1500] days for the two transfers further away from the Sun. The Earth to Group 4 transfer included a maximum V_∞ at launch of 3.5 km s^{-1} with φ in $[1/6\pi, 5/6\pi]$ and θ_1 in $[0, 2\pi]$, while the other three transfers have rendezvous boundary conditions. Due to the random inputs, some transfers have physically very unfavorable conditions leading to high ΔV (and a few NaN) values. We therefore filter the database for transfers below 1000 km s^{-1}, leaving 438 526 sample transfers. As previously the network was expanded to an architecture of 3/128 and the data set was split into training and validation sets at a ratio of 90/10. Training lasted for 275 epochs in 20 min on the GPU and the ANN reached a MAPE of 14.6 % and sorting accuracy of 88 %.

The algorithm is then run with sequences of 2 generations and its convergence is shown in Figure 20. The surrogate search after the first sequence results in a new champion for all runs and continues to produce useful solutions up until the 5th sequence. The result is a reliable convergence of all runs below 30 km s^{-1} within 850 fitness function evaluations.

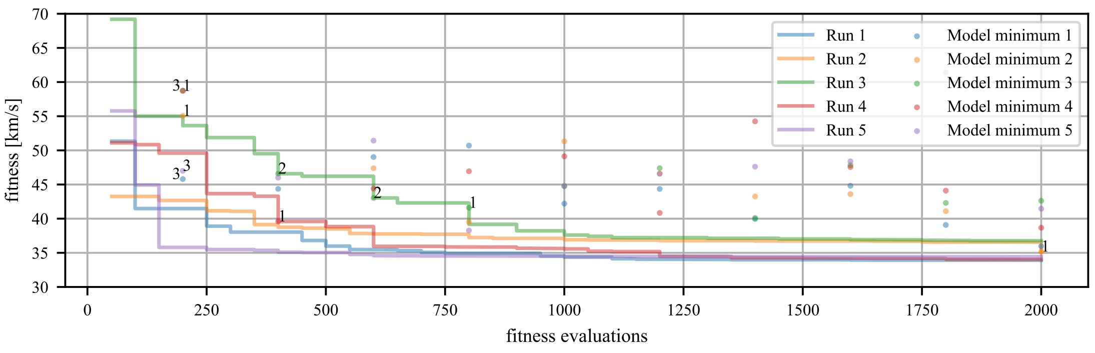


Figure 19. Individual transfer surrogate applied to the Asteroid problem (Pop. 50, Gen. 4, Seq. 10)

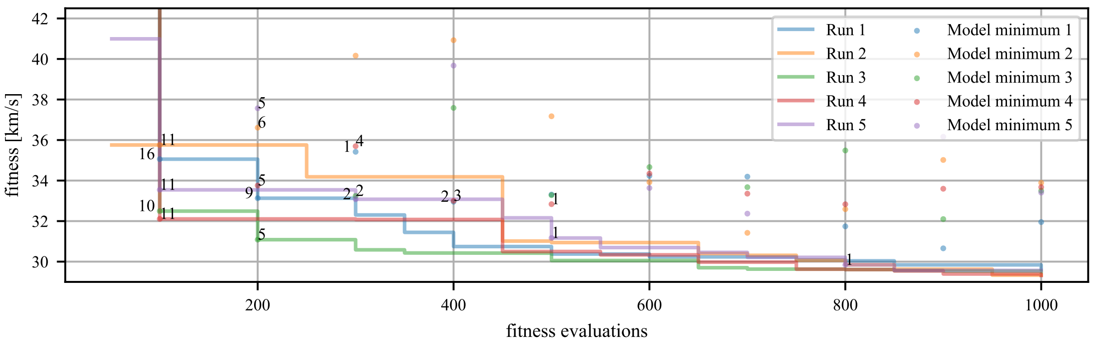


Figure 20. Pretrained general transfer surrogate applied to the Asteroid problem (Pop. 50, Gen. 2, Seq. 10)

CONCLUSION

A flexible Python tool for preliminary optimization of interplanetary low-thrust trajectories has been developed. It is employed in the development of a surrogate-assisted evolutionary optimization approach using an Artificial Neural Network to approximate the problem's fitness function and find new candidate solutions that are reinserted into the original population following an elitist strategy. Different ways to design the surrogate are investigated. The full surrogate approach of directly modeling the relationship between candidate solution and fitness suffers from the initially small size of the data set. Concentrating on the computationally expensive parts of the fitness function, the ΔV computation, and thereby simplifying the approximated problem and using more specialized transfer surrogates leads to better results as more useful new solutions are found especially early on, increasing the overall fitness of the population. Extending the online approach with a pretrained model that was trained on a general data set consisting of shaped transfers improved the initial quality of the model greatly, consistently yielding new best solutions. The pretrained model continued to be useful by being incrementally trained on the new data points becoming available during the optimization run, allowing the surrogate to adapt to the region of interest.

We conclude that the inclusion of a surrogate can improve the evolutionary optimization of low-thrust multi-phase trajectories and especially the availability of a pretrained model can greatly speed up the preliminary search for trajectories. For future work we recommend the further investigation of surrogates in this context.

REFERENCES

[1] C. E. Garner *et al.*, "Ion propulsion: An enabling technology for the dawn mission," *23rd AAS/AIAA Spaceflight Mechanics Meeting, Kauai, Hawaii*, 2013.

[2] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.

[3] A. E. Petropoulos and J. M. Longuski, "Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories," *Journal of Spacecraft and Rockets*, Vol. 41, No. 5, 2004, pp. 787–796.

[4] 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, 2018, pp. 76–98.

[5] S. Li, X. Huang, and B. Yang, "Review of optimization methodologies in global and China trajectory optimization competitions," *Progress in Aerospace Sciences*, 2018.

[6] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, Vol. 1, No. 2, 2011, pp. 61–70.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, Vol. 521, No. 7553, 2015, p. 436.

[8] D. J. Gondelach and R. Noomen, "Hodographic-shaping method for low-thrust interplanetary trajectory design," *Journal of Spacecraft and Rockets*, Vol. 52, No. 3, 2015, pp. 728–738.

[9] D. Izzo, "Global optimization and space pruning for spacecraft trajectory design," *Spacecraft Trajectory Optimization* (B. A. Conway, ed.), ch. 7, pp. 178–201, Cambridge University Press, 2010.

[10] D. Morante, M. Sanjurjo Rivo, and M. Soler, "Multi-Objective Low-Thrust Interplanetary Trajectory Optimization Based on Generalized Logarithmic Spirals," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 3, 2018, pp. 476–490.

[11] W. M. Folkner, J. G. Williams, D. H. Boggs, R. S. Park, and P. Kuchynka, "The planetary and lunar ephemerides DE430 and DE431," *Interplanetary Network Progress Report*, Vol. 196, 2014, pp. 1–81.

[12] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, Vol. 9, No. 1, 2005, pp. 3–12.

[13] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Neural networks: Tricks of the trade*, pp. 437–478, Springer, 2012.

[14] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830. Version 0.21.2.

[15] Y. Jin, M. Olhofer, and B. Sendhoff, "On evolutionary optimization with approximate fitness functions," *2nd Annual Conference on Genetic and Evolutionary Computation*, 2000, pp. 786–793.

[16] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations*, 2014.

[18] M. J. Powell, "The BOBYQA algorithm for bound constrained optimization without derivatives," *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 2009, pp. 26–46.

[19] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, Vol. 7, No. 4, 1965, pp. 308–313.

[20] B. J. Wall and B. A. Conway, "Genetic algorithms applied to the solution of hybrid optimal control problems in astrodynamics," *Journal of Global Optimization*, Vol. 44, No. 4, 2009, p. 493.

[21] J. Zhang *et al.*, "Evolutionary computation meets machine learning: A survey," *IEEE Computational Intelligence Magazine*, Vol. 6, No. 4, 2011, pp. 68–75.

[22] C. Ampatzis and D. Izzo, "Machine learning techniques for approximation of objective functions in trajectory optimisation," *IJCAI-09 Workshop on Artificial Intelligence in Space*, 2009, pp. 1–6.

[23] Y.-S. Hong, H. Lee, and M.-J. Tahk, "Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks," *Engineering Optimization*, Vol. 35, No. 1, 2003, pp. 91–102.

[24] M. Mernik *et al.*, "On clarifying misconceptions when comparing variants of the Artificial Bee Colony Algorithm by offering a new implementation," *Information Sciences*, Vol. 291, 2015, pp. 115–127.

[25] M. D. Rayman and K. C. Patel, "The Dawn project's transition to mission operations: On its way to rendezvous with (4) Vesta and (1) Ceres," *Acta Astronautica*, Vol. 66, No. 1-2, 2010, pp. 230–238.

[26] M. D. Rayman and R. A. Mase, "The second year of Dawn mission operations: Mars gravity assist and onward to Vesta," *Acta Astronautica*, Vol. 67, No. 3-4, 2010, pp. 483–488.

[27] 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, 2011, pp. 128–147.

[28] T. Roegiers, "Application of the Spherical Shaping Method to a Low-Thrust Multiple Asteroid Rendezvous Mission: Implementation, limitations and solutions," *M.Sc. Thesis, TU Delft*, 2014.

3 Conclusions and recommendations

This chapter provides extended conclusions resulting from revisiting the research questions as well as more detailed recommendations for future work.

The research questions defined at the beginning of this work were:

- *How can the optimization of linked, low-thrust trajectories based on shaped transfers be improved by Artificial Neural Network models?*
  The implemented approach of using the ANN that was trained on previous function evaluations provides a safe way to generate new candidate solutions that was shown to assist the algorithm's convergence, especially during earlier generations. As this was the only point of applying the surrogate no conclusion can be made if this is the best approach. By having more control over the GA, for example by regularly forcing reevaluations of the whole population, as well as having more detailed knowledge about the evolving quality of the surrogate, it may be possible to offload more computations to the model and employ it during the regular evolution.
- *Can a neural network surrogate model be useful in the evolutionary optimization of low-thrust trajectories?*
  The results of applying the presented algorithm to the linked, low-thrust example problems show that surrogate assisted evolutionary algorithms are well suited for this type of problem, which provides no gradient information and has a highly non-linear fitness function. The GA's settings are problem dependent and a preliminary search should be conducted to determine suitable parameters. The main contributing factor to the ANN's performance is the amount of available data which naturally increases during the evolution, leading to an increasing quality of candidate solutions found by utilizing the model. If these candidates are of sufficient quality to serve as a new, better member of the original population is depending on both the convergence of the GA as well as the quality of the model. The experiments show that there is a benefit in using a surrogate specialized on the low-thrust transfer and/or a general pretrained surrogate over a surrogate generated fully online.
- *Can an Artificial Neural Network be used to approximate the fitness function in an Evolutionary Algorithm?*
  Approximating the complete fitness function, i.e. the full surrogate approach, did not produce a benefit over an unassisted GA when applied to the two example problems. It is therefore recommended to adapt the surrogate approach to the problem at hand, which resulted in an increase in the quality of candidate solutions found utilizing the surrogate. A good approach is to focus on the expensive part of the fitness function and ways to simplify the approximation problem. In interplanetary trajectory optimization, this is generally the evaluation of the transfer legs, computing the velocity increment necessary to fly the suggested trajectory.
- *How can the surrogate be integrated into an existing Genetic Algorithm?*
  Being able to build on an existing GA implementation reduced the implementation effort but also imposed some limitations. The chosen approach of training and searching the surrogate separately from the GA evolution makes the approach broadly applicable to many population-based

algorithms, but may be missing out on better synergies from a deeper integration of the surrogate into the algorithm. The presented algorithm is therefore one solution that works for existing algorithms. Further research should include a custom implementation of the GA.

- *Is it possible to generate the surrogate fully online?*
  This question can be answered with a conditional yes. The general transfer surrogate provided a clearly improved convergence on the Asteroid problem, while none of the online surrogates improved convergence on the Dawn problem. In order to achieve an obvious immediate improvement, pretraining the surrogate on related (Dawn problem) or general (Asteroids) transfers was needed.

- *How can the surrogate be improved based on prior knowledge about the specific problem?*
  Two approaches were found that adapt the surrogate to the problem at hand and thereby increase its performance. The first one is to concentrate the approximation on the expensive part of the fitness function. Small helper functions, e.g. querying ephemeris data or coordinate transformations, are very fast to compute but make it more difficult to learn for the ANN. Keeping these helper functions is therefore beneficial and recommended. The second approach is to use pretraining. This way training is not started at zero but the ANN is already adapted to the type of problem, rapidly increasing the model's initial quality while also reducing the time needed for incremental training. Generating a general database of transfers and pretraining the ANN only has to be done once and can then be reused in the optimization of diverse trajectories. This is one of the main recommended directions for future work, see below.

The goal of *improving the optimization of linked, low-thrust trajectories based on a shaping method by incorporating Artificial Neural Network models into an Evolutionary Algorithm* was achieved and provides a starting point for a number of possible directions for further research in this area:

- The algorithm's speed in terms of wall-clock time can be improved by a better implementation of parallel computation. Both the GA and ANN training are well suited for parallelization. Pygmo is taking steps towards enabling finer levels of parallelization already, e.g. with the inclusion of a `batch fitness evaluator` in the most recent release (Version 2.11, August 2019). Neural Network libraries also include parallel computation abilities, for example, Theano uses 4 CPU threads by default. Training of larger networks is also well suited for Graphics Processing Unit (GPU) computation, offering opportunities to train a surrogate on the GPU while the GA evolves the population on the CPU, fully utilizing the capabilities of a desktop computer.

- Before diving further into the implementation there are improvements to be made to the algorithm itself. Currently, the surrogate is trained and searched at fixed intervals. The results have shown, however, that its most use is achieved early during the optimization. An improvement in terms of overall computational effort can therefore be made by spending most of the surrogate effort as soon as a base amount of data is available and tapering the effort off later on when the GA is finding smaller, local improvements. Local convergence of the algorithm could also be improved by including a local refinement step at regular intervals and the end, making sure that local minima are precisely found.

- The GA is a classical, well understood evolutionary algorithm. A large effort has been made to improve EAs in recent years, with a multitude of new algorithms, often developed for specific applications. As the implemented surrogate approach is not deeply integrated into the algorithm it easily translates to other population-based algorithms and it would be interesting to investigate how the choice of EA affects the combined performance.

- The most successful experiment was employing a pretrained general transfer surrogate, which also provides the most promise for future research. By generating a large database of low-thrust transfers in the solar system, including wide ranges on the number of revolutions and the velocity boundary conditions for impulsive shots and flybys, a very general model could be trained and used during preliminary trajectory optimization, further improving the initial quality of the surrogate and reducing the total number of function evaluations until convergence. The general model can be reused for a multitude of missions, quickly justifying the pretraining effort.

- Following the increased focus on pretraining a general model leads to opportunities surpassing the use of shaping methods. In fact, it could be possible that an ANN trained on a large database of optimal trajectories could yield a better approximation of a specific transfer's ΔV than the shaping computation, thereby replacing the shaping method as the model of choice for fast, preliminary optimization.

4 Background on the used methods

As the presented paper manuscript focuses on the novel algorithm and its application, this chapter provides some more background on the employed models and methods. Specifically the hodographic shaping method is presented in more detail, including the internal computations. The Genetic Algorithm as implemented in Pygmo is outlined and Artificial Neural Networks as implemented in Keras/Theano are presented, including details on training.

4.1. The hodographic shaping method

The hodographic shaping method was developed by Gondelach [12, 13]. Here, the version employing time as the input variable is used and presented. A version based on the polar angle was also developed but did not perform as well for problems going to outer targets.

The low-thrust trajectory is represented by three velocity functions. Each of these velocity functions $V_\square$ describes the velocity's time evolution in the corresponding direction ($\mathbf{e}_r$, $\mathbf{e}_\theta$, or $\mathbf{e}_z$), and is assembled by a variable number of analytically integrable and differentiable base functions v_i which are multiplied by shape coefficients c_i:

$$V_\square = \sum_{i=1}^{n} c_i v_i(t), \tag{4.1}$$

where t is time and n is the number of base functions. Radius r, vertical distance z, and polar angle θ can be computed from the respective base function by integration. Similarly, the respective accelerations are calculated by differentiating the base function:

$$r(t) = r_0 + \int_0^t V_r \mathrm{d}t, \qquad \ddot{r} = \dot{V}_r, \tag{4.2}$$

$$z(t) = z_0 + \int_0^t V_z \mathrm{d}t, \qquad \ddot{\theta} = \frac{\dot{V}_\theta}{r} - \frac{V_\theta V_r}{r^2}, \tag{4.3}$$

$$\theta(t) = \theta_0 + \int_0^t \frac{V_\theta}{r} \mathrm{d}t, \qquad \ddot{z} = \dot{V}_z. \tag{4.4}$$

Having chosen suitable base functions, these can all be computed analytically, except for $\theta(t)$ which has to be integrated numerically due to its dependence on the radial coordinate.

Three shape coefficients per velocity function can be computed with a linear system of equations from the boundary conditions on initial and final velocity and the difference in position:

$$V_r(0) = V_{r,0}, \qquad V_r(t_f) = V_{r,f}, \qquad \int_0^{t_f} V_r \mathrm{d}t = r_f - r_0, \tag{4.5}$$

$$V_\theta(0) = V_{\theta,0}, \qquad V_\theta(t_f) = V_{\theta,f}, \qquad \int_0^{t_f} \frac{V_\theta}{r} \mathrm{d}t = \theta_f = \psi + 2\pi N, \tag{4.6}$$

$$V_z(0) = V_{z,0}, \qquad V_z(t_f) = V_{z,f}, \qquad \int_0^{t_f} V_z \mathrm{d}t = z_f - z_0, \tag{4.7}$$

where the subscripts 0 and f indicate initial and final conditions, respectively, ψ is the transfer angle, and N is the number of revolutions.

For the radial and vertical directions the resulting linear system of equations can be solved analytically for the shape coefficients c using the base functions v and their integrals p:

$$\begin{bmatrix} v_1(0) & v_2(0) & v_3(0) \\ v_1(t_f) & v_2(t_f) & v_3(t_f) \\ p_1(t_f)-p_1(0) & p_2(t_f)-p_2(0) & p_3(t_f)-p(0) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} V_0 - \sum_{i=4}^{n} c_i v_i(0) \\ V_f - \sum_{i=4}^{n} c_i v_i(t_f) \\ P_f - P_0 - \sum_{i=4}^{n} \left[p_i(t_f) - p_i(0)\right] \end{bmatrix}, \tag{4.8}$$

where $V_{\square}$ and $P_{\square}$, are the boundary conditions on velocity and position, respectively.

The solution in the transverse direction involves the computation of a numerical integral due to Equation 4.3:

$$\int_0^{t_f} \dot{\theta}\mathrm{d}t = \int_0^{t_f} \frac{V_\theta}{r}\mathrm{d}t = \int_0^{t_f} \frac{c_1 v_1 + c_2 v_2 + c_3 v_3 + \sum_{i=4}^{n} c_i v_i}{r}\mathrm{d}t = \theta_f. \tag{4.9}$$

Using this relation and the equations for the boundary conditions on the initial and final velocity an equation depending only on c_3 can be derived:

$$\begin{bmatrix} v_1(0) & v_2(0) \\ v_1(t_f) & v_2(t_f) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} V_0 - \sum_{i=4}^{n} c_i v_i(0) \\ V_f - \sum_{i=4}^{n} c_i v_i(t_f) \end{bmatrix}, \tag{4.10}$$

$$\rightarrow \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = c_3 \begin{bmatrix} v_1(0) & v_2(0) \\ v_1(t_f) & v_2(t_f) \end{bmatrix}^{-1} \begin{bmatrix} -v_3(0) \\ -v_3(t_f) \end{bmatrix} + \begin{bmatrix} v_1(0) & v_2(0) \\ v_1(t_f) & v_2(t_f) \end{bmatrix}^{-1} \begin{bmatrix} V_0 - \sum_{i=4}^{n} c_i v_i(0) \\ V_f - \sum_{i=4}^{n} c_i v_i(t_f) \end{bmatrix}, \tag{4.11}$$

and writing the last equation in a shortened form:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = c_3 \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} + \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} \tag{4.12}$$

leads to an expression for c_3:

$$c_3 = \frac{\theta_f - \int_0^{t_f} \left[\left(L_1 v_1 + L_2 v_2 + \sum_{i=4}^{n} c_i v_i\right)/r\right]\mathrm{d}t}{\int_0^{t_f} \left[\left(K_1 v_1 + K_2 v_2 + v_3\right)/r\right]\mathrm{d}t}. \tag{4.13}$$

Plugging the found value for c_3 back into Equation 4.10 yields the remaining shape coefficients c_1 and c_2.

By substituting the individual terms into the Equations of Motion it is then possible to compute the required thrust acceleration $\mathbf{f}$ to fly the specified trajectory assuming a two-body system where the central body's gravitational attraction and the spacecraft's thrust are the only acting forces:

$$\ddot{r} - r\dot{\theta}^2 + \frac{\mu}{s^3}r = f_r, \tag{4.14}$$

$$r\ddot{\theta} + 2\dot{r}\dot{\theta} = f_\theta, \tag{4.15}$$

$$\ddot{z} + \frac{\mu}{s^3}z = f_z. \tag{4.16}$$

Here, μ is the central body's gravitational parameter and s is the distance between central body and spacecraft ($s = \sqrt{r^2 + s^2}$). The total velocity increment along the trajectory, ΔV, is computed by numerical integration:

$$\Delta V = \int_0^{ToF} f\mathrm{d}t. \tag{4.17}$$

The velocity functions (see Equation 4.1) can be constructed from an arbitrary number of base functions. As three coefficients per direction follow from the boundary conditions, any additional, so called free, coefficients have to be determined using an external optimization procedure.

Gondelach uses a custom nomenclature to encode the used base function for specific experiments. His most successful and recommended shapes in the lowest-order computation are CPowPow2, CPowPow2, and CosR5P3CosR5P3SinR5 for V_r, V_θ, and V_z, respectively. For the 6 Degree of Freedom (DoF) method he recommends CPowPow2+PSin05PCos05, CPowPow2+PSin05PCos05, and CosR5P3CosR5P3SinR5+P4CosR5P4SinR5, in the same order [13]. For the sake of clarity these base functions are spelled out in mathematical notation in Table 4.1. The time t in these equations is normalized with respect to the ToF.

Table 4.1: Employed base functions

Name	Equation		
C Pow Pow2	c_1	$+c_2 t$	$+c_3 t^2$
CosR5 P3CosR5 P3SinR5	$c_1 \cos(2\pi t(N + 0.5))$	$+c_2 t^3 \cos(2\pi t(N + 0.5))$	$+c_3 t^3 \sin(2\pi t(N + 0.5))$
PSin05 PCos05	$c_4 t \sin(0.5 t\pi)$	$+c_5 t \cos(0.5 t\pi)$	
P4CosR5 P4SinR5	$c_4 t^4 \cos(2t\pi(N + 0.5))$	$+c_5 t^4 \sin(2t\pi(N + 0.5))$	

4.2. Genetic Algorithms

Work on Genetic Algorithms (GAs) was pioneered by John Holland and his student David Goldberg [11] starting in the 1960s. They are a type of Evolutionary Algorithm (EA) and draw inspiration from nature by applying the principles of natural evolution to optimization. This way GAs are able to handle mixed integer programming, i.e. problems where an input variable is restricted to integer values and generally problems that are not well suited for classical optimization. Examples are discontinuous and highly non-linear fitness functions, stochastic, and nondifferentiable problems [51].

As one of the most general optimization algorithms, GAs have frequently been applied to space trajectory optimization [41] for more than 25 years [10]. Examples include the optimization of interplanetary patched-conics trajectories [10, 50], the search for global solutions and for free parameters in shaped low-thrust transfers [49], and the GTOC problems [24]. Figure 4.1 shows the general structure of a GA, as implemented in Pygmo [4].

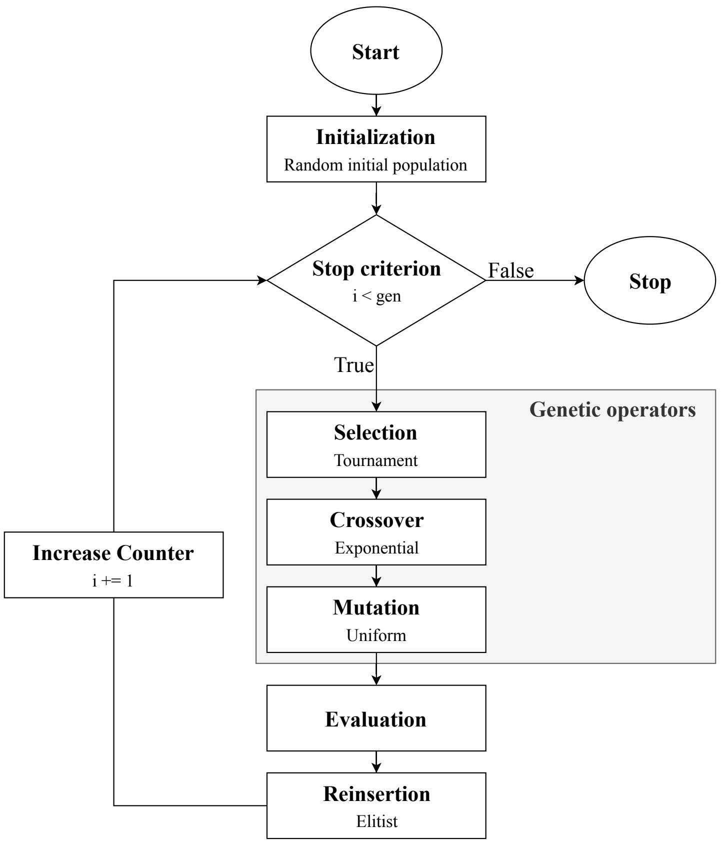


Figure 4.1: The Genetic Algorithm used as a baseline to develop the ML augmentation approach

The algorithm operates by evolving a population of *chromosomes*, also called *individuals*, which encode a set of input parameters into a single numerical vector. Evolution is achieved by the genetic operators of *selection*, *crossover*, and *mutation*. Selection is the process of choosing two parent chromosomes from the population for reproduction, i.e. the process of creating new *offspring* chromosomes. A variety of selection schemes are possible, ranging from purely random selection to repeatedly choosing the current best chromosomes (truncated). Here, a *tournament* selection with a size of 2 is performed, choosing the better of two random chromosomes. Crossover is the main method to perform reproduction. The parent chromosomes are combined by an exchange of genes, i.e. the problem's input parameters. Some implementations do this once, exchanging all genes after the randomly chosen crossover point (single), some look at each gene individually (binomial). Here, *exponential* crossover is performed: The algorithm iterates over the parent chromosomes' genes, starts exchanging genes with a set probability and stops the exchange when the set probability is triggered again. This way there are two random points in the chromosome that determine the range of exchanged genes. The final genetic operator is *mutation* which randomly changes the value of genes, introducing an element that does not depend on the parent chromosomes.

After a candidate population has been created it is evaluated with respect to the problem's fitness function. The initial and candidate populations are then compared and the new population is constructed from the best chromosomes. Pygmo implements reinsertion in an *elitist* way: A ranking based only on fitness value determined which chromosomes are transferred into the new population.

One of the main factors influencing the GA's performance on difficult problems is the balance of exploration and exploitation. The search space is explored by the crossover and mutation operators, generating new chromosomes that differ from previously evaluated ones. Good solutions are exploited through the selection of good population members, the elitist reinsertion, and the general approach of generating new chromosomes from the previous generation. The best parameters and settings are problem dependent and were derived as presented in Chapter 2.

4.2.1. Surrogate modeling

An interesting extension to EAs is *surrogate modeling* [19]. While classical EAs like the shown GA discard previous fitness function evaluations, surrogates aim to make use of that information by constructing an approximate model, which is used to guide the optimization and/or reduce the overall computational effort. Ways to generate, include, and employ the surrogate are diverse [20]. The most common application is to approximate the full or partial fitness function, which is especially useful for computationally expensive fitness functions. Other tasks include population initialization and augmentations to the genetic operations [52]. Additionally the surrogate can be employed at different levels in the EA loop, with the main categories being individual-, generation-, and population-based approaches [20].

This thesis research was preceded by a Literature Study [44], which presented the most common techniques to construct the surrogate, i.e. polynomial models, Gaussian Process Regression (GPR)/Kriging, Support Vector Machines (SVMs), and Artificial Neural Networks. It was concluded that feedforward ANNs with multiple layers represent the most promising approach for the application at hand due to their flexible size and learning, a surge of new tools in recent years, and a superior ability to generalize from (limited) data in comparison to other ML methods [23]. ANNs are presented in Section 4.3.

4.3. Feedforward Artificial Neural Networks

Artificial Neural Networks (ANNs) are inspired by biological neural networks and have been a concept in Artificial Intelligence (AI) and Machine Learning (ML) for a long time [40], starting with the multilayer perceptron in the 1950s [36]. Recently, advances in learning methods for deep ANNs, i.e. architectures with multiple hidden layers, have resulted in a boom of research, development, and successful applications [16, 23]. Currently, ANNs represent the best solution to a wide range of problems ranging from image recognition to natural language processing [27]. As the focus shifted from reproducing the function of the brain to solving specific problems, a number of specialized architectures have emerged. For example, Convolutional Neural Network included convolution and pooling layers, specializing on feature extraction in image data [22]. Here, non-linear regression on structured data is performed, resulting in a good application for the classical feed-forward ANN with fully connected layers [44].

Following its biological inspiration, ANNs consist of multiple connected layers of neurons. Each neuron computes the weighted sum of all input values a, and adds a bias b [27]:

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} - b_j^l, \tag{4.18}$$

where w_{jk} is the weight connecting neurons j of the current and k of the previous layer. The superscript denotes the current layer, where l is the current and $l-1$ is the previous layer. Then, the activation function σ is applied, computing the neuron's activation a:

$$a_k^l = \sigma\left(z_k^l\right), \tag{4.19}$$

and the layered computation is repeated for all neurons and layers until the output layer is reached.

These equations are vectorized and can be expressed in matrix form:

$$a^l = \sigma\left(w^l a^{l-1} + b\right), \tag{4.20}$$

resulting in efficient and fast computation on modern hardware, especially GPUs. Figure 4.2 shows a feed-forward ANN. The number of neurons in each layer and the number of hidden layers are flexible and should be chosen according to the task at hand.

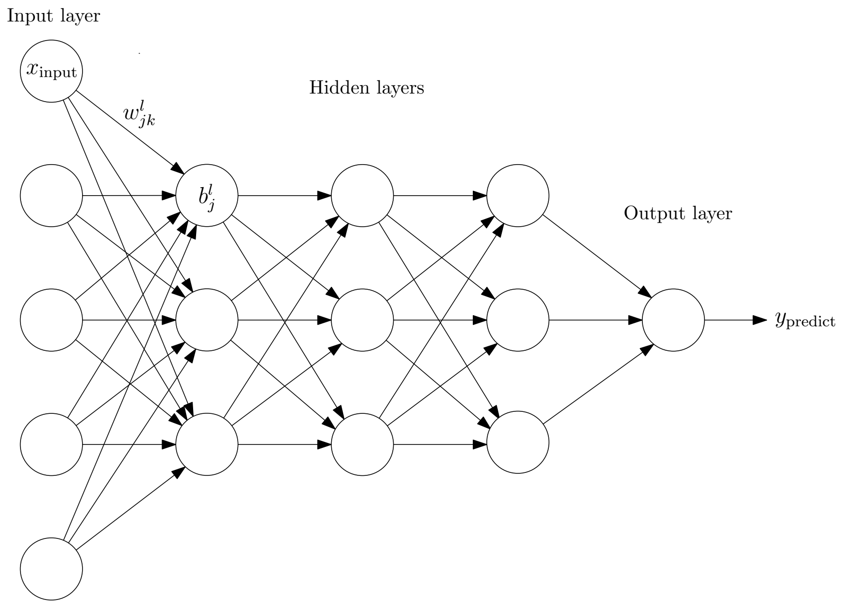


Figure 4.2: A feed-forward Artificial Neural Network

A number of different activation functions have resulted in good results in previous research [27]:

$$\text{sigmoid}(z) = \frac{1}{a + e^{-z}}, \qquad \tanh = \frac{2}{1+e^{-2x}} - 1, \qquad \text{ReLU} = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}, \tag{4.21}$$

with the Rectified Linear Unit (ReLU) emerging as the favorite choice for hidden layers in recent years [23]. In comparison to the sigmoid and tanh functions, it is faster computed, and has a constant, high gradient, which facilitates fast learning especially during early training [40].

The choice of activation function for the output layer depends on the network's intended purpose. Activation functions which lead to outputs in the range [0, 1] are preferred for classification tasks, as the output can then be interpreted as a probability or confidence in the chosen class. Examples are the sigmoid, tanh, and softmax functions. For regression tasks the output is usually chosen to be linear, in order to not restrict the range of possible outputs.

The central property of ANNs is that they can "learn" from data by carefully adapting their weights and biases in a training procedure. Here, the network is shown known pairs of input and output data and adapts its parameters based on the mismatch (loss) between prediction and known true output. The classical method to perform supervised learning is gradient descent in combination with back-propagation [38], which provides an efficient way of computing the gradient of the loss function with respect to the network's parameters [27]. In the wake of rising popularity of ANNs, a number of extended algorithms specialized on ANN training have been developed [37]. Here, the popular Adaptive Moment Estimation (Adam) algorithm is employed [21, 37]. In comparison to classic gradient descent it computes adaptive learning rates for each parameter and keeps a momentum measure based on running averages of the gradient and its second moment.

Additional measures taken to improve the ANN's learning are batch training, i.e. estimating the current gradient based on a batch of example data, learning rate decay and early stopping. These method as well as the choice of hyper-parameters, i.e. all parameters that are not learned from data, are explained in Chapter 2. The ANN was implemented using the Theano [45] and Keras [6] Python packages. An example of the progression of training and validation error for the presented approach is shown in Figure 4.3. Due to the employed output normalization the Mean Squared Error (MSE) training loss starts close to unity and improves during the following epochs almost monotonically while the validation loss exhibits more variation. During one epoch the network's parameters are updated once based on each training example, e.g. 200 training samples and a batch size of 10 lead to 20 updates of the weights and biases per epoch. Between epochs 70 and 120 the validation loss does not improve, meaning that the learning rate is reduced, which leads to a slight improvement in validation loss and a large reduction of the training loss and can be seen in the plot as smoother changes. Training and validation loss diverge now, signaling that the network is overfitting to the training data and losing generalization ability. As the validation loss does not improve further for the next 130 epochs, training is stopped and the ANN that resulted in the minimum validation loss is returned.

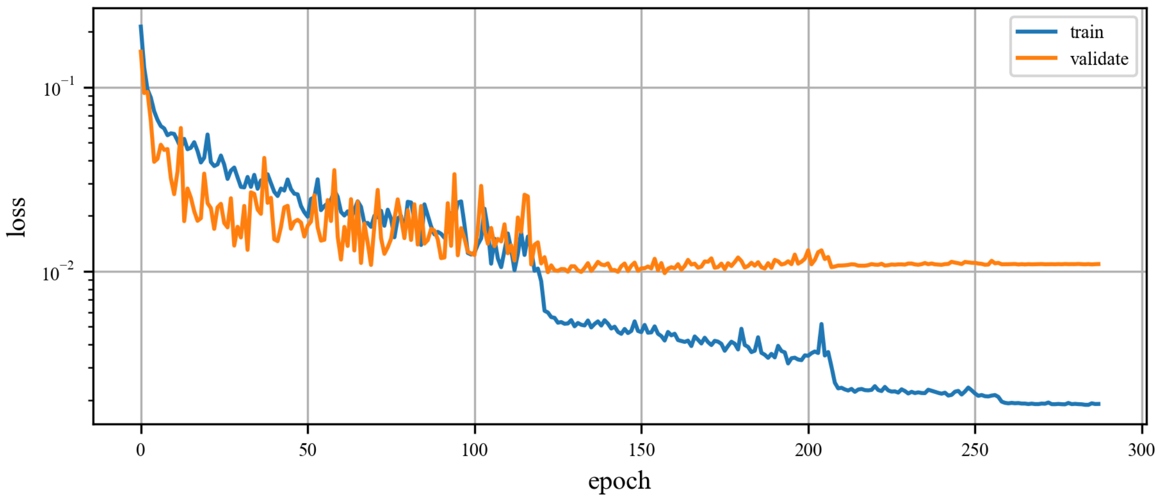
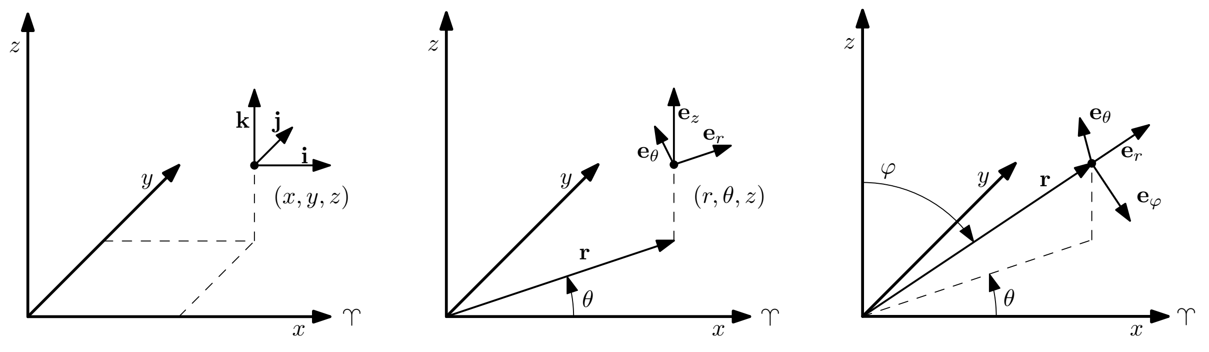


Figure 4.3: Example of the progression of training and validation loss using learning rate reduction and early stopping (500 samples, 1 hidden layer, 128 neurons)

A
Reference frame, coordinate systems, and conversions

In order to express positions and velocities in the solar system a reference frame is necessary. In this work an inertial reference frame centered at the Sun is used, following the `ECLIPJ2000` frame as implemented in SPICE [9]. The frame's x-y plane coincides with the mean ecliptic plane, with the x-axis pointing at the vernal equinox at the J2000 epoch, the z-axis pointing towards the north ecliptic pole, and the y-axis completing the right hand frame.

The coordinate system defines the approach to defining positions and velocities within the given reference frame. Here, mainly Cartesian and cylindrical coordinates are used, as shown in Figure A.1.



(a) Cartesian coordinates (b) cylindrical coordinates (c) Spherical coordinates

Figure A.1: Coordinate systems

The hodographic shaping method was formulated in cylindrical coordinates because radius, i.e. the distance from the Sun, and polar angle exhibit less variation over the course of an orbit than x and y, therefore making it more straightforward to find suitable shaping functions [12]. Radius r, polar angle θ, and vertical distance z can be computed from Cartesian coordinates as:

$$\begin{aligned}
r &= \sqrt{x^2 + y^2} \\
\theta &= \arctan \frac{x}{y} \\
z &= z.
\end{aligned}$$

Cartesian coordinates (x, y, z) are employed to compute the change in velocity during a planetary flyby and to apply general impulsive shots. They are related to spherical coordinates as:

$$
\begin{aligned}
x &= r\cos\theta \\
y &= r\sin\theta \\
z &= z.
\end{aligned}
$$

In the special case of expressing an initial impulsive shot at launch, ΔV_∞, spherical coordinates attached to the spacecraft are used in order to decouple the departure date and the impulsive shot's magnitude and direction during the optimization as much as possible. The shot is then converted into local cylindrical coordinates and added to the spacecraft's velocity:

$$
\begin{aligned}
\Delta V_r &= \Delta V_\infty \sin\varphi\cos\theta_1 \\
\Delta V_\theta &= \Delta V_\infty \sin\varphi\sin\theta_1 \\
\Delta V_z &= \Delta V_\infty \cos\varphi \\
\mathbf{V}_0 &= \begin{pmatrix} V_r \\ V_\theta \\ V_z \end{pmatrix} + \begin{pmatrix} \Delta V_r \\ \Delta V_\theta \\ \Delta V_z \end{pmatrix},
\end{aligned}
$$

where $\mathbf{V}_0$ is the spacecraft's velocity after the impulsive shot, usually the initial condition for the computation of the low-thrust transfer. The geometry is shown in Figure A.2.

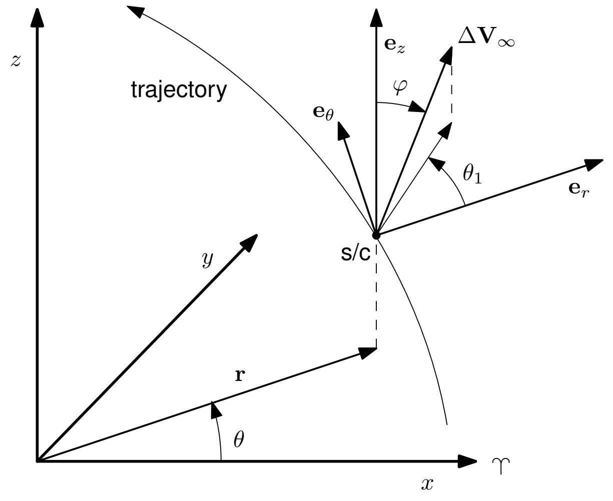


Figure A.2: The geometry of impulsive shots

Cartesian velocities can be converted to cylindrical coordinates using:

$$
\begin{aligned}
V_r &= \dot{r} = \frac{x\dot{x} + y\dot{y}}{\sqrt{x^2 + y^2}} \\
V_\theta &= r\dot{\theta} = \frac{x\dot{y} - y\dot{x}}{\sqrt{x^2 + y^2}} \\
V_z &= \dot{z}.
\end{aligned}
$$

Note that V_θ describes the spacecraft's velocity in the direction of e_θ (in m s^{-1}) and not the rate of change of the polar angle which is $\dot{\theta}$ (in rad s^{-1}).

The opposite conversion from cylindrical to Cartesian velocities is given as:

$$
\begin{aligned}
\dot{x} &= V_r\cos\theta - V_\theta\sin\theta \\
\dot{y} &= V_r\sin\theta + V_\theta\cos\theta \\
\dot{z} &= V_z.
\end{aligned}
$$

These conversions are implemented in the Python tool developed to conduct the presented research. The Astrodynamics code is available at
`https://github.com/lstubbig/hodographic-shaping-method-python`.

B Verification

The self implemented functions and methods have been verified with respect to sources in the literature. Specifically these are the shaping method, the flyby model, and coordinate transformations.

B.1. Hodographic shaping method

Here, both the implementation of the hodographic shaping method itself as well as the initial guess generation are verified.

B.1.1. Implementation

Gondelach performed a Grid Search of the launch window for a low-thrust transfer to Mars using the time-driven, lowest-order method. Here, the velocity shape is fully determined by the chosen base functions and the boundary conditions. His result is shown in Figure B.1 next to the replicated results in this work.

The settings of the hodographic shaping method and the best trajectories found both in the original thesis and the implementation in this work are listed in Table B.1. The table also includes the results of the higher-order method.

Table B.1: Reference solutions and my implementation

	V_r and V_θ	V_z	Dep. [mjd]	ToF [days]	ΔV [km/s]	$f_{\max}$ [10^{-4} m s^{-2}]
[12]	CPowPow2	CosR5 P3CosR5 P3SinR5	10025	1050	6.342	1.51
Here	CPowPow2	CosR5 P3CosR5 P3SinR5	10025	1050	6.348	1.514
[12]	CPowPow2 PSin05 PCos05	CosR5 P3CosR5 P3SinR5 P4CosR5 P4SinR5	9985	1100	5.771	1.50
Here	CPowPow2 PSin05 PCos05	CosR5 P3CosR5 P3SinR5 P4CosR5 P4SinR5	9985	1100	5.773	1.474

The best found trajectory, i.e. the transfer that needs the lowest total velocity increment, is shown in Figure B.2. Here, the three-dimensional trajectory as well as the corresponding thrust profile over time is shown. The upper plot is taken from Gondelach's original thesis and the lower plot was created using the Python code here.

Adding two more base functions per coordinate adds six degrees of freedom to the shape. Using a local optimization algorithm, here Nelder-Mead simplex, considerably improved trajectories can be found. The optimization uses zero as the initial guess for the coefficient of each additional base function, corresponding to the lowest-order solution. Figure B.3 shows the results of this grid search in comparison to the reference implementation of the algorithm.

The corresponding best trajectories found with the higher-order method (6 DoF) are compared in Figure B.4.

As Gondelach chose a relatively coarse amount of bins in his contour plots, Figures B.5a and B.5b show the results of my grid searches in more detail. Each of these “best” plots is comprised of grid searches for different numbers of revolution around the Sun ($N = [0, 5]$), choosing the best option for each launch date and Time of Flight. The individual plots for each N are shown in Figure B.6 and B.7. It is apparent that the optimizer does not fully converge for each point in the launch window. This is an observation that Gondelach made as well. It can be seen, that the optimizers mainly struggles for high-DeltaV trajectories after a change in the geometry of the problem (the range of the transfer angle is $[0, \pi)$ meaning that it resets in regular intervals). As these trajectories are unlikely to be global optima, they do not appear in the combined plot of the best trajectories.

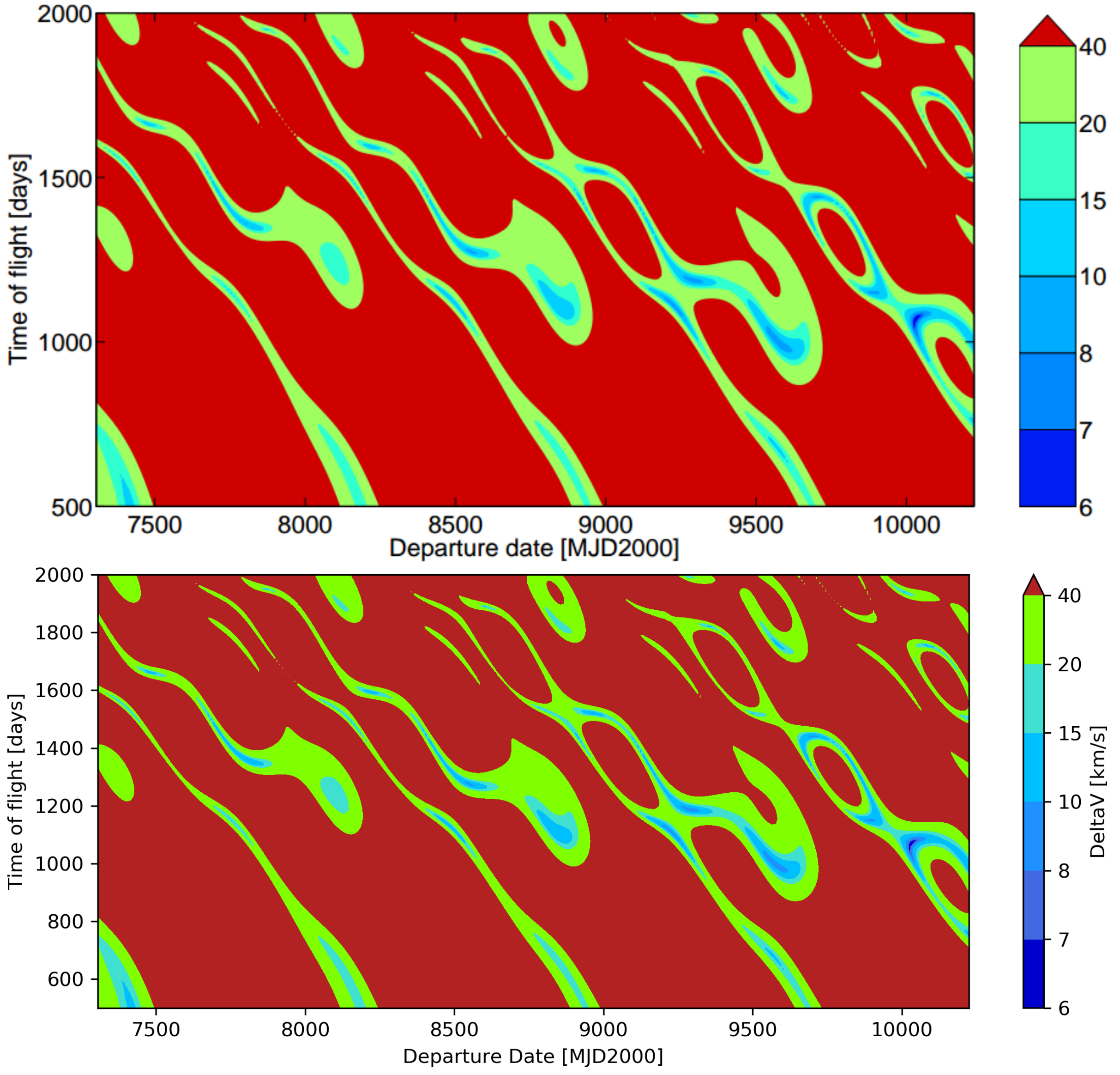
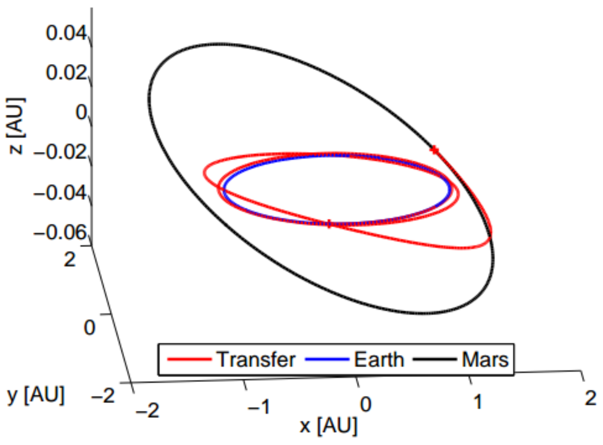
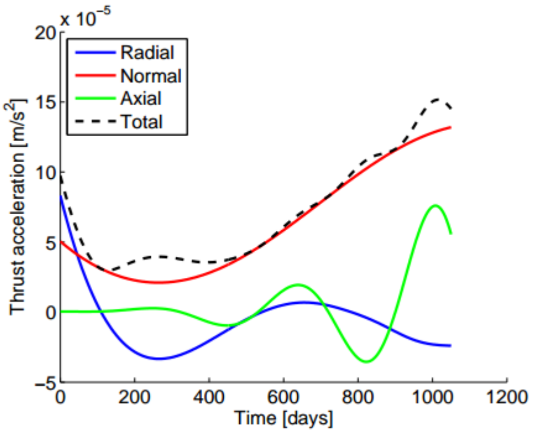


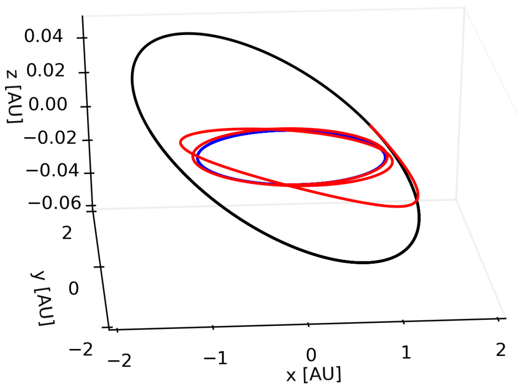
Figure B.1: Grid search result of the lowest-order method for a transfer to Mars (top from [12], bottom implementation in this work)





(a) Trajectory

(b) Thrust profile



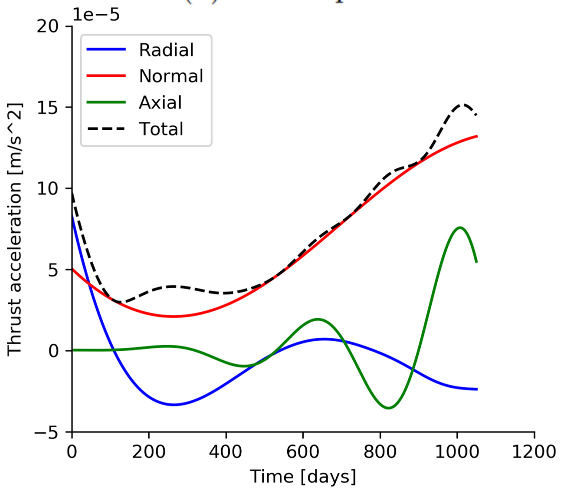


Figure B.2: Best found trajectory of the lowest-order method (top from [12], bottom implementation in this work)

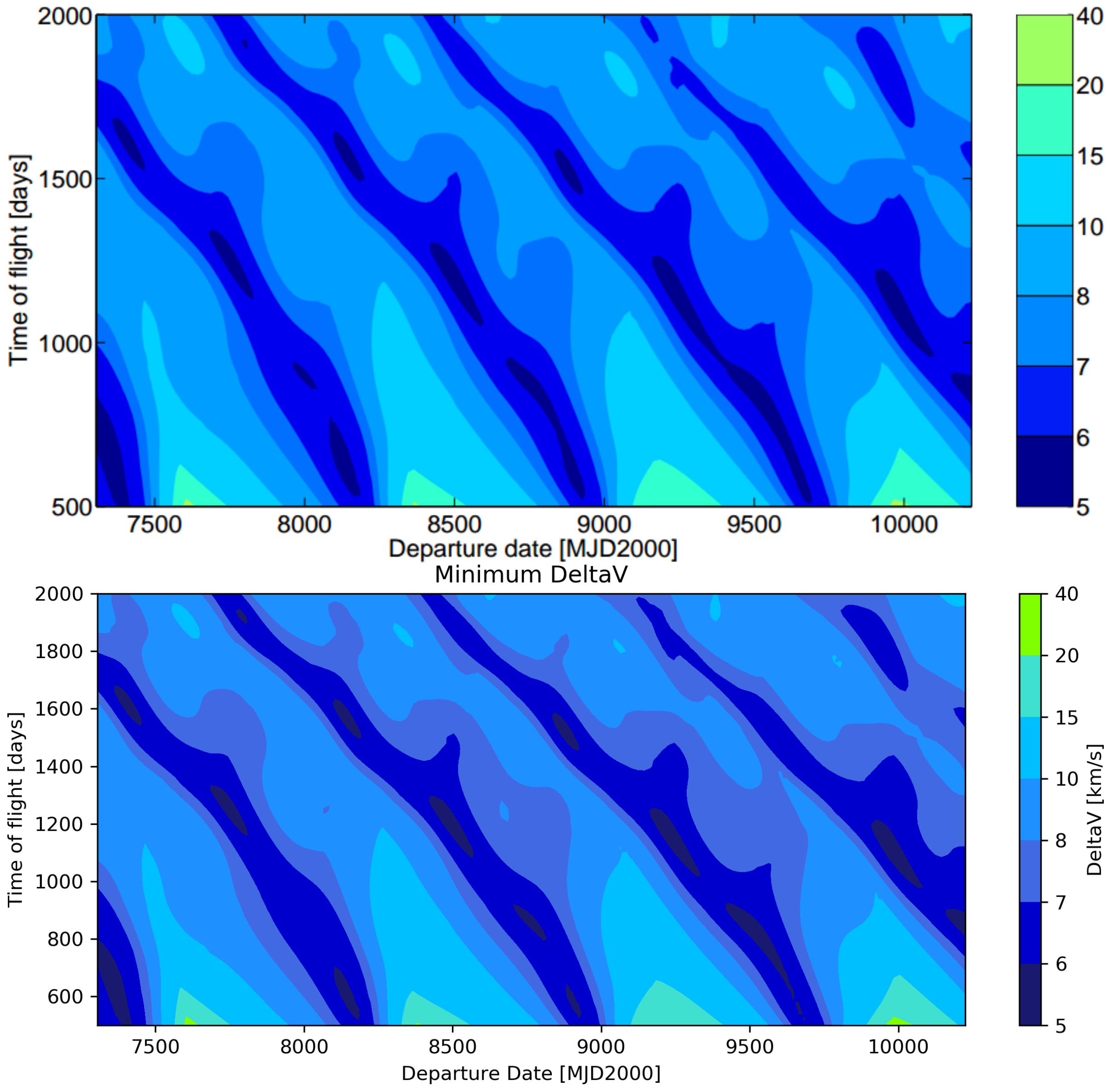
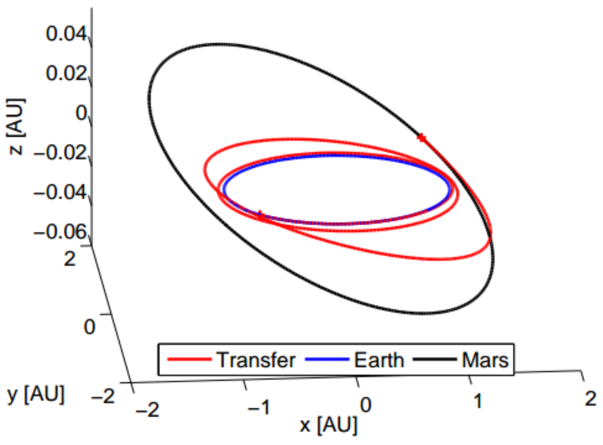
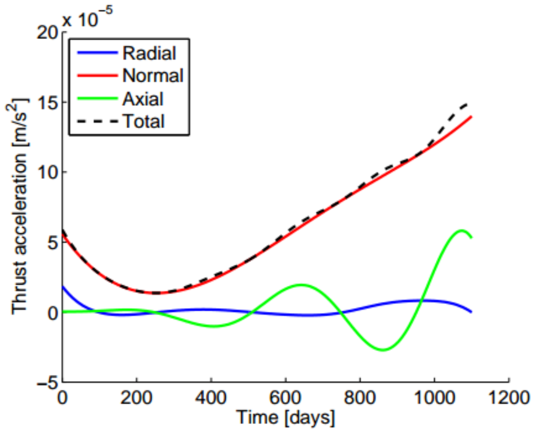


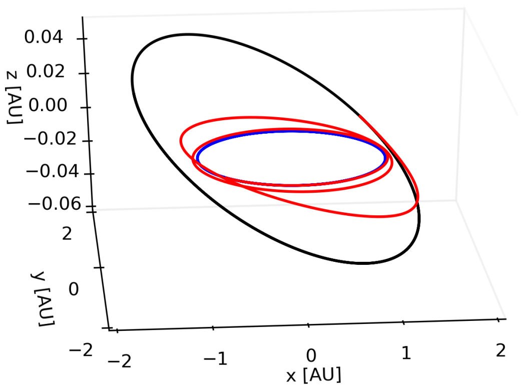
Figure B.3: Grid search result of the higher-order method (6 DOF) for a transfer to Mars (top from [12], bottom implementation in this work)



(a) Trajectory



(b) Thrust profile



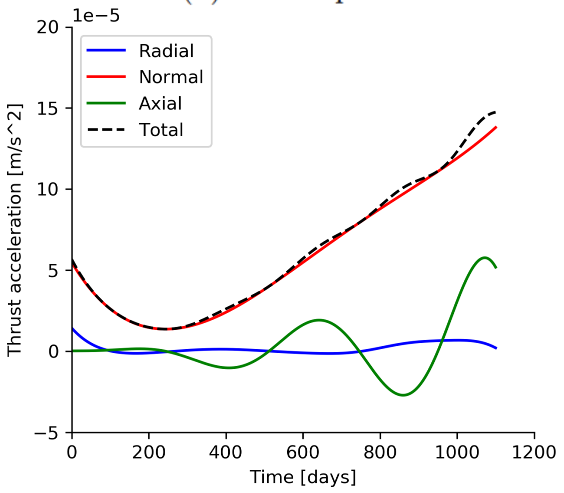
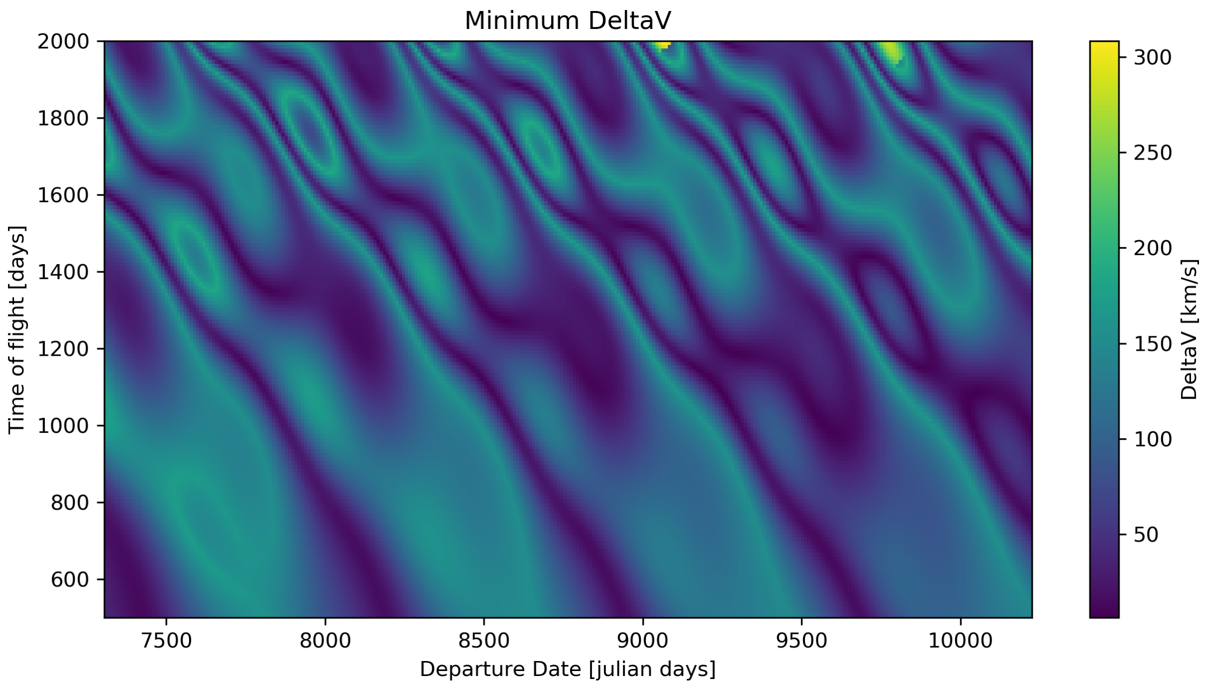
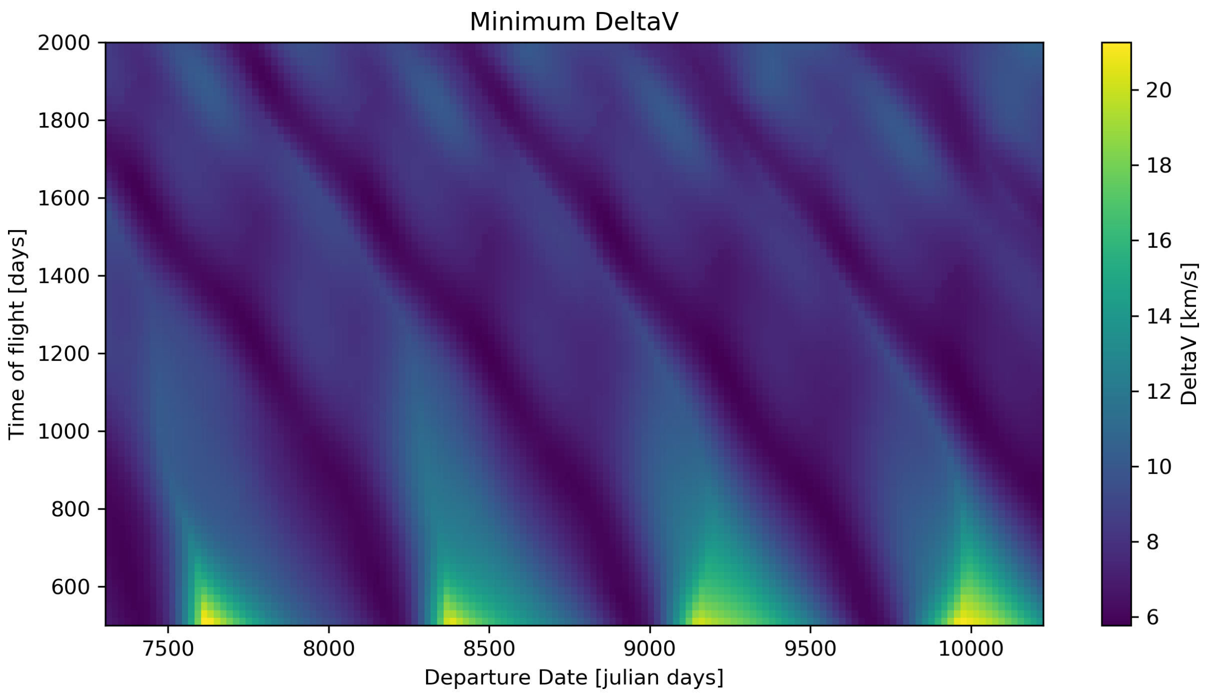


Figure B.4: Best found trajectory of the higher-order method (top from [12], bottom implementation in this work)



(a) Best transfer options to Mars in the chosen launch window for the lowest-order solution



(b) Best transfer options to Mars in the chosen launch window for the higher-order solution

Figure B.5: Detailed plots of the computed launch windows

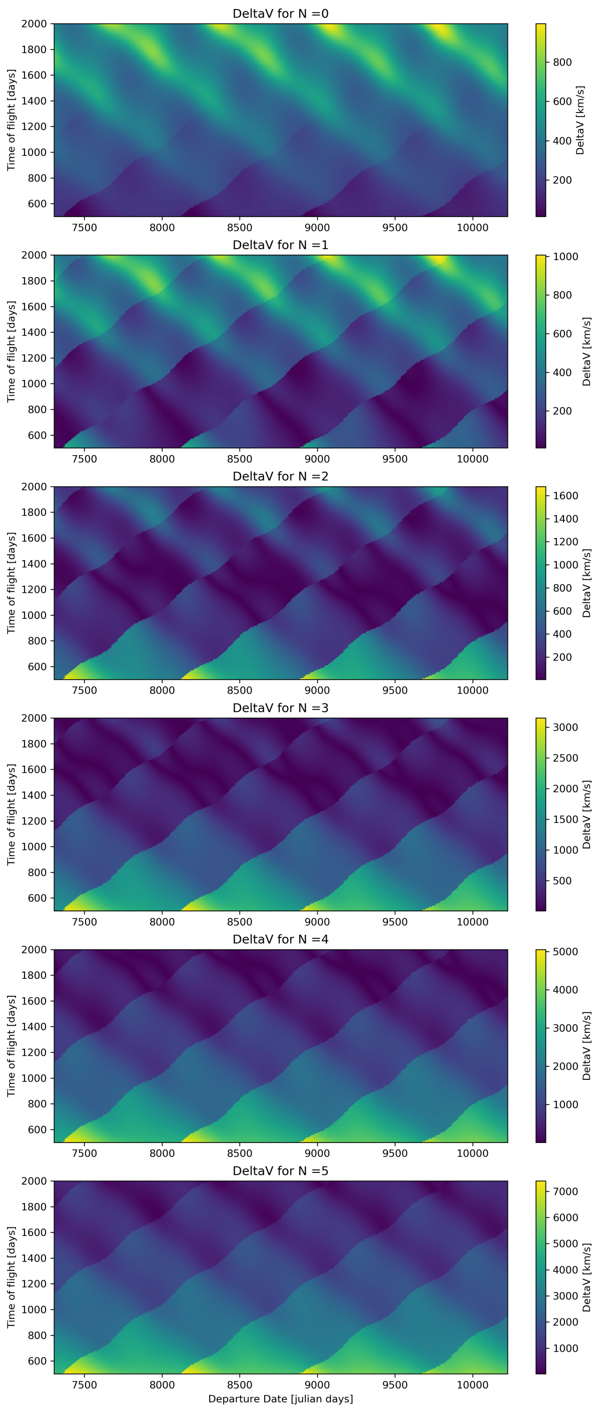


Figure B.6: Best transfer options to Mars in the chosen launch window for each N (lowest-order solution)

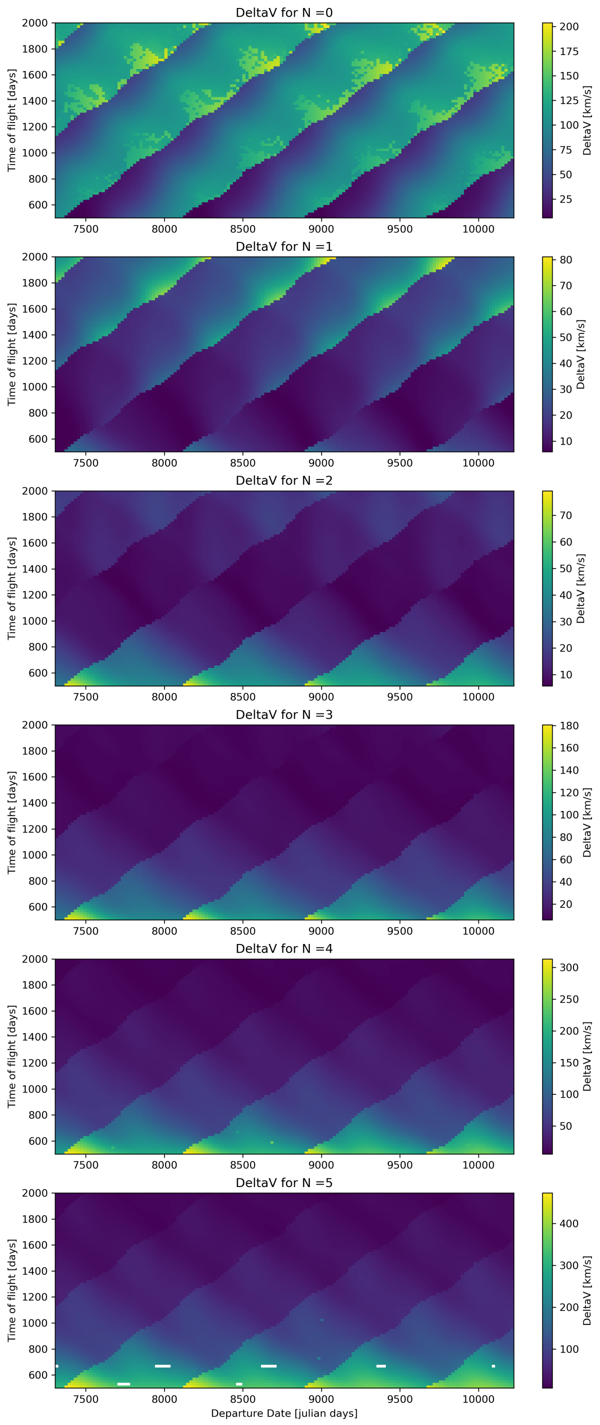


Figure B.7: Best transfer options to Mars in the chosen launch window for each N (higher-order solution)

B.1.2. Choice of an initial guess

In the paper manuscript the choice of using the Nelder-Mead simplex algorithm for the optimization of the shaped transfers was motivated with a comparison of different algorithms. Here, some additional remarks are made with respect to the initial guess used to start the shape parameters' optimization in a grid search.

The Nelder-Mead optimizer performs well but not perfectly. Gondelach recommends using the result of the previous optimization as the initial guess as a means to speed up the optimization. As the grid search gradually moves over the launch window, the best found solution and the corresponding shape parameters vary gradually. By using the previous result as the initial guess, the optimizer starts closer to the optimum and converges in fewer iterations. Strictly using the previous solution however results in a high number of artifacts in the ΔV-plot. After a change in the transfer's problem geometry the optimizer has trouble converging and using that wrong result to initialize the next computation causes the convergence difficulties to propagate in the grid search direction. This can be seen as the streaks in Figure B.9 (right). The grid search moves from low to high ToFs. As these effects happen in the worse areas of the transfer window, they are not as big of a problem in the combined result, see Figure B.8. Some obvious artifacts persist nevertheless.

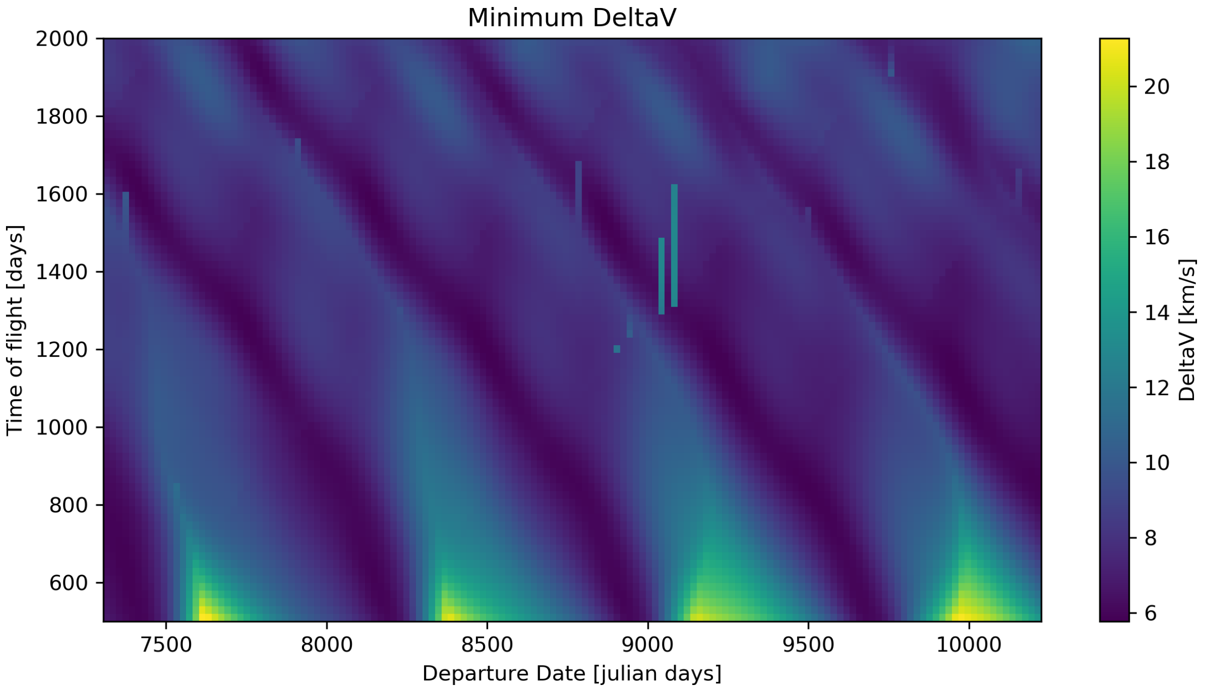


Figure B.8: Combined results, strictly using the previous trajectory as initial guess

The results of an approach to improving convergence is shown on the left of Figure B.9. Here, the initial guess was reset whenever a discontinuity of the transfer angle was detected [12], which is implemented as comparison between the current and previous transfer angle ψ:

$$\mathbf{c}_{\text{initial}} = \begin{cases} \mathbf{0} & \text{if } \psi_{\text{previous}} > \psi_{\text{current}} \\ \mathbf{c}_{\text{previous}} & \text{else} \end{cases} \tag{B.1}$$

In addition to previous work, a reset is also triggered if the previous optimization did not converge within a given time limit, stopping individual convergence problems to spread. The time limit has to be chosen based on experience of optimization runs on the given computer system; here about 5 s was useful to catch outliers while not disturbing regular optimization runs. This approach improves convergence behavior especially for trajectories with a higher number of revolutions, as can be seen in the plots for $N = 2$ and more in B.9 (left). The results for $N = 0$ seem to be slightly worse, the reasons for which are not clear. Looking at the results without any initial guess, see Figure B.7, the regions of longer ToF right after a geometry change for $N = 0$ are quite challenging for the optimizer and perhaps the resets have an adverse effect here. The improved method of resetting the initial guess at transfer

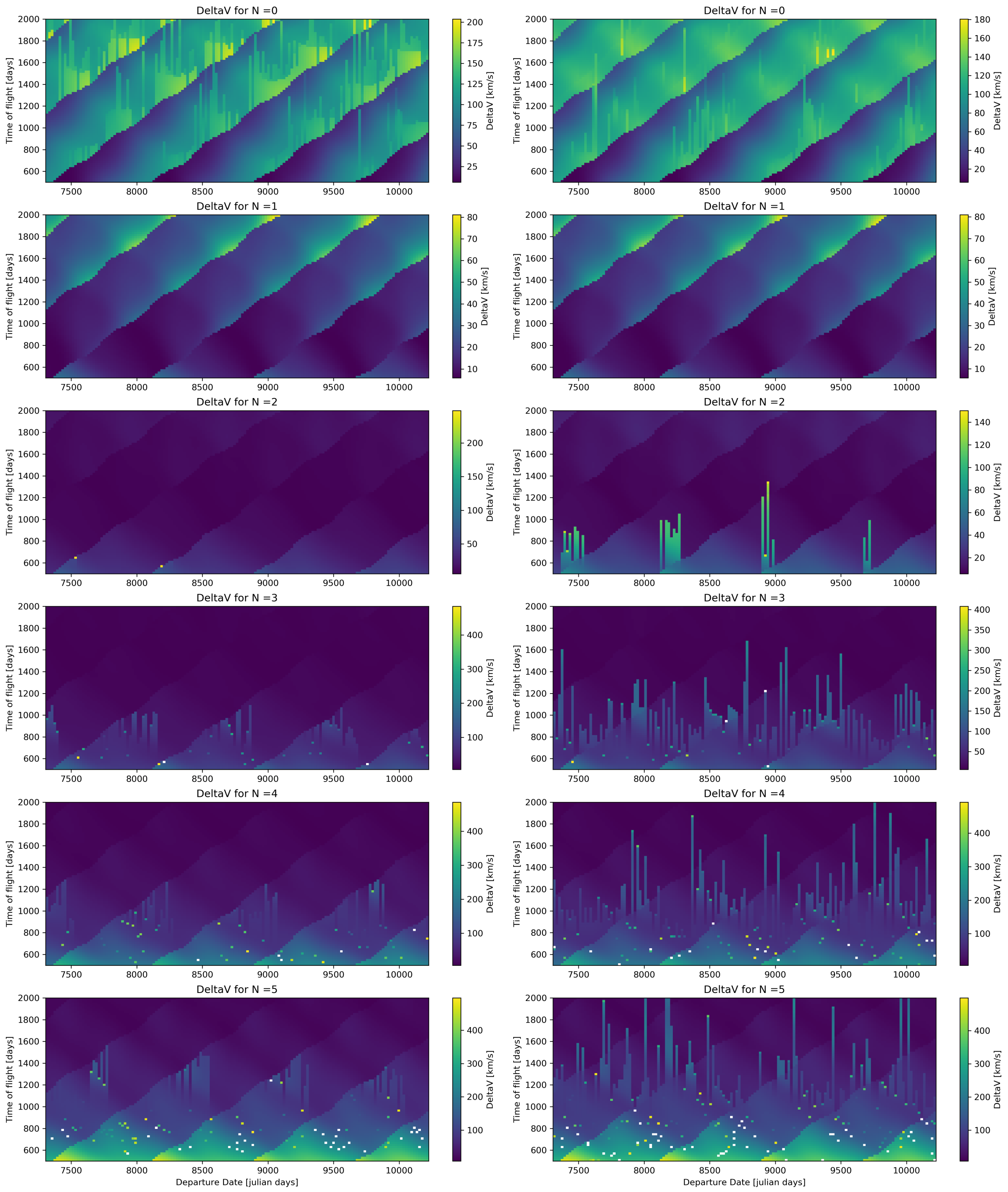


Figure B.9: Best transfer options to Mars in the chosen launch window for each N (higher-order solution), using the result of the previous computation as initial guess.
Left: Resetting the initial guess at the geometry change.
Right: Strictly using the previous result as initial guess.

angle discontinuities results in a combined result for the launch window without any artifacts caused by the optimizer, identical to Figure B.5b.

In general these techniques to using adapted initial guess are only applicable to grid searches in the launch window with a small step size. In this work the shaping method is used to model linked trajectories that are optimized using heuristic methods. As there is no set search direction, the initial guess was therefore uniformly set to $\mathbf{0}$, which is a robust choice as it represents a physically possible trajectory already satisfying the boundary conditions. The experience from the Earth-Mars grid search, as shown in Figure B.7, supports this choice.

B.2. The flyby model

In order to verify the flyby model, the example in the book by Curtis is reproduced [7, Example 8.6], which computes a flyby in the ecliptic plane at Venus. The velocity vectors of the spacecraft $\mathbf{V}_{\mathrm{sc}}$ and Venus $\mathbf{V}_{\mathrm{venus}}$ are given in cylindrical coordinates:

$$\mathbf{V}_{\mathrm{sc}} = \begin{pmatrix} -2.782\,\mathrm{km\,s^{-1}} \\ 37.51\,\mathrm{km\,s^{-1}} \\ 0 \end{pmatrix}, \quad \mathbf{V}_{\mathrm{venus}} = \begin{pmatrix} 0 \\ 35.02\,\mathrm{km\,s^{-1}} \\ 0 \end{pmatrix}. \tag{B.2}$$

The position vector is arbitrarily set to

$$\mathbf{P}_{\mathrm{sc}} = \begin{pmatrix} 108.2 \times 10^{6}\,\mathrm{m} \\ -30^{\circ} \\ 0 \end{pmatrix}, \tag{B.3}$$

following the given radius in the example.

Converting the state vectors $\mathbf{x} = [\mathbf{P}, \mathbf{V}]^T$ into Cartesian coordinates and plugging them into the `flyby()` function leads to the values listed in Table B.2 where they are compared to the corresponding values in [7]. As Curtis only considers the in-plane motion, the two cases of performing the flyby in front or behind the planet are differentiated by assigning a positive or negative direction to the turn angle (with respect to the z-axis). The flyby model here is implemented in 3D, meaning that the computed turn angle is always positive and the two cases correspond to plane angles β of 270° and 90°, respectively. The flyby is performed at a distance d of 300 km, leading to a flyby periapsis $r_3 = r_{♀} + d$ of 6352 km. The comparison lists the computed values for both cases, which are all within the margin of rounding errors. The flyby model is thus considered verified.

Table B.2: Flyby comparison to [7]

	Dark/Leading side approach		Sunlit/Trailing side approach	
Parameter	Here	Curtis	Here	Curtis
β	90°	—	270°	—
Direction of δ	—	positive	—	negative
$\tilde{V}_{\mathrm{in}}$	3.734 km s⁻¹	3.733 km s⁻¹	3.734 km s⁻¹	3.733 km s⁻¹
δ	103.59°	103.6°	103.59°	−103.6°
V_{out}	31.780 km s⁻¹	31.78 km s⁻¹	37.266 km s⁻¹	37.27 km s⁻¹
V_{out_r}	−1.766 km s⁻¹	−1.766 km s⁻¹	3.074 km s⁻¹	3.074 km s⁻¹
V_{out_θ}	31.731 km s⁻¹	31.73 km s⁻¹	37.139 km s⁻¹	37.14 km s⁻¹

B.3. Coordinate transformations

The transformations between cylindrical and Cartesian coordinate systems presented in Section A were implemented in custom functions and are verified here. For the position conversions there are corresponding functions in Matlab, providing reference values to compare to. The found differences for a set of 10 example coordinates are in the order of the machine precision for the double float data type and is therefore considered to be verified. For the velocity conversions there are no equivalent functions available, which is why a set of example values is transformed from cylindrical to Cartesian and back. Visually inspecting the data, see Figure B.10, and observing that the initial values are obtained after the reverse transformation to machine precision, provide a high confidence in a correct implementation. The corresponding numbers for both cases are listed in Table B.3.

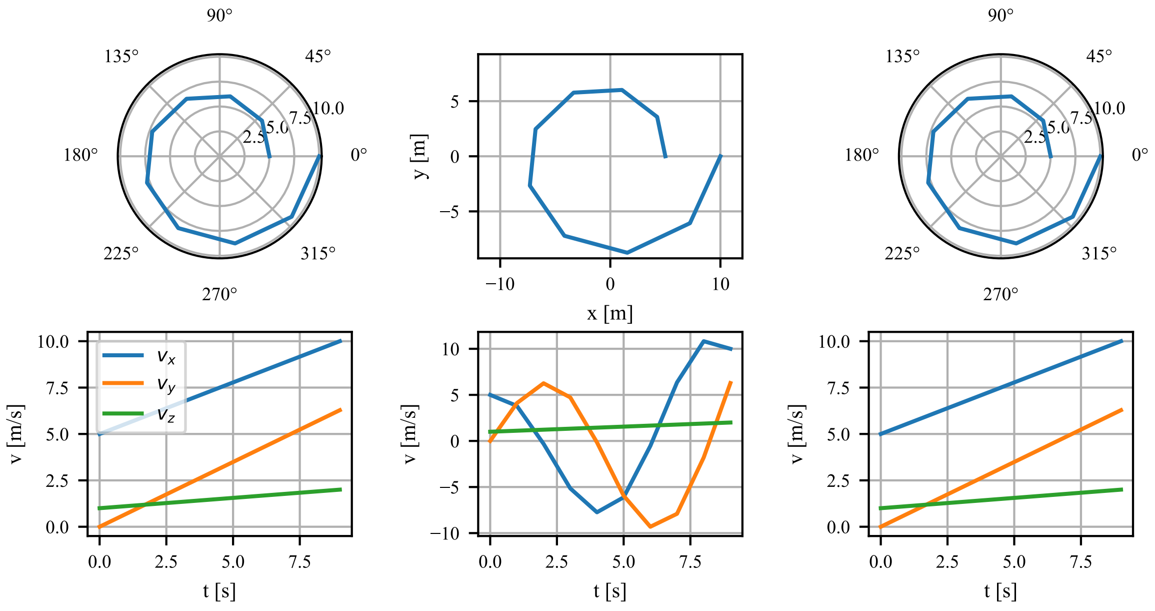


Figure B.10: Conversion of exemplary points between cylindrical and Cartesian coordinate systems

Table B.3: Data from the verification of coordinate transformations

Positions		Velocities	
Range of input values			
θ	[0, 2π]	v_θ	[0, 2π]
r	[5, 10]	v_r	[5, 10]
z	[1, 2]	v_z	[1, 2]
Max. difference to Matlab			
Cartesian	8.88×10^{-16}	Cartesian	—
Cylindrical	4.44×10^{-16}	Cylindrical	—
Max. difference after reverse conversions			
Cyl. → Cart.→ Cyl.	8.88×10^{-16}	Cyl. → Cart.→ Cyl.	1.78×10^{-15}

C Validation

Here, the dynamical model to compute low-thrust trajectories is validated to confirm that the results are suitable for preliminary optimization. The Genetic Algorithm and the ANN are checked as well to confirm that they perform their intended purpose.

C.1. Dynamical model

Trajectories computed in a two-body system consisting of a heavy central body and a massless spacecraft is one of the standard approaches to preliminary trajectories and often employed in literature. Examples include [7, 33, 46, 47]. Linking transfers by matching the position and computing a velocity change based on a physics model is also common. In comparison to the very common approach of patched conics the spacecraft's reference is not changed to the visited intermediate body but the velocity change is assumed to occur instantaneously. As the Spheres of Influence (SOIs) of solar system bodies are small in comparison to the Sun, this is a valid first-order approximation that is also frequently applied in literature [18, 25]. The trajectory model is therefore considered validated.

C.2. Genetic algorithm

The employed Genetic Algorithm (GA) was implemented by the Advanced Concepts Team (ACT) at European Space Agency (ESA) as part of their Pagmo/Pygmo optimization library [4]. Pagmo/Pygmo is used extensively in various research projects [5, 26, 28]. In combination with its observed convergence on the given example problems, which is much better than the random search conducted as a baseline, the GA is considered validated.

C.3. Artificial neural network

The Machine Learning part of the algorithm, the Artificial Neural Network, was implemented using the Keras [6] Application Programming Interface (API) and the Theano backend [45], both of which are widely used in ML research, development, and production [15]. Theano was developed at the University of Montreal while Keras is an open source community effort with funding from multiple companies and organizations. Combined with the implemented ANN's observed ability to learn and predict, the approach of using Keras and Theano is considered validated.

D Additional information

This chapter presents some additional information about the algorithm and the reference problems. First, the average conversion behavior of each optimization approach, classic and surrogate-assisted, is shown. Then, the thrust profiles of the best found trajectories for the Asteroid and Dawn problem are presented to show their level of feasibility. More information about the algorithms computational effort is given and the chapter is concluded by data from the unassisted GA runs on both problems, motivating the chosen parameter bounds.

D.1. Average convergence plots

The paper, see Chapter 2, shows plots of 5 individual runs for each of the 4 versions of the surrogate-assisted algorithm. Here, the corresponding averages are shown in one plot together with the classical GA without any surrogate assistance. Figure D.1 shows the case for the Dawn case. As described in the paper, there is, on average, little influence for all three versions of the online surrogate approach. The pretrained case, however, converges much faster, as the first surrogate search consistently leads to new champions and considerably improves the overall fitness of the population. As visible in the individual plots, the pretrained surrogate also keeps improving due to the performed incremental training. Beyond 500 fitness function evaluations all approaches behave similarly, slowly converging towards 12 km s^{-1}.

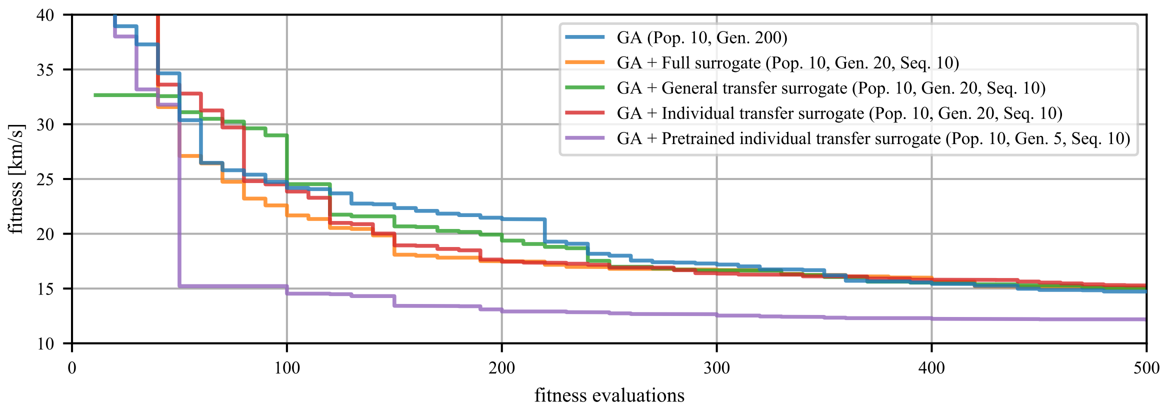


Figure D.1: Averaged convergence of the investigated algorithms on the Dawn problem

Figure D.2 shows the average convergence of the different approaches on the Asteroid problem. Here, both the general transfer surrogate and the individual transfer surrogate converge faster than the unassisted GA due to new solutions found utilizing the surrogate visible as steps in the plots at 200, 400 and 600 fitness evaluations. The individual transfer surrogate does, however, not converge to the same optimum within the observed 2000 fitness evaluations. The reasons for this behavior are unclear.

An interference from the surrogate can be ruled out due to the taken approach to evolution control: If no better solutions are found during surrogate utilization, the original GA evolution is not affected. The pretrained general transfer surrogate assisted GA converges the fastest, with a large step towards the minimum after the first sequence at 50 fitness evaluations. The surrogate's influence is less visible afterwards, but the plot of individual runs, see Figure 20, showed that the surrogate keeps supplying new candidate solutions until the third sequence, making this approach the most promising for future research.

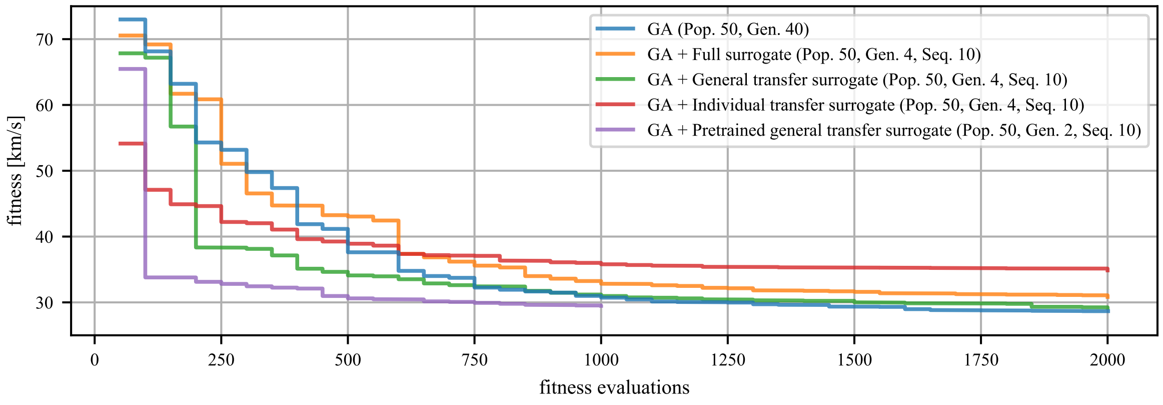


Figure D.2: Averaged convergence of the investigated algorithms on the Asteroid problem

D.2. Thrust profiles

The hodographic shaping method does not consider a limit on the maximum thrust acceleration. Here, it is checked if the resulting thrust levels are reasonable to further consider the found trajectories. Figure D.3 shows the best trajectory found using the unassisted GA ($\Delta V = 11.8\,\mathrm{km\,s^{-1}}$). As mentioned in the paper the maximum thrust acceleration of $2 \times 10^{-4}\,\mathrm{m\,s^{-2}}$ is feasible using current technology, considering a spacecraft like Dawn.

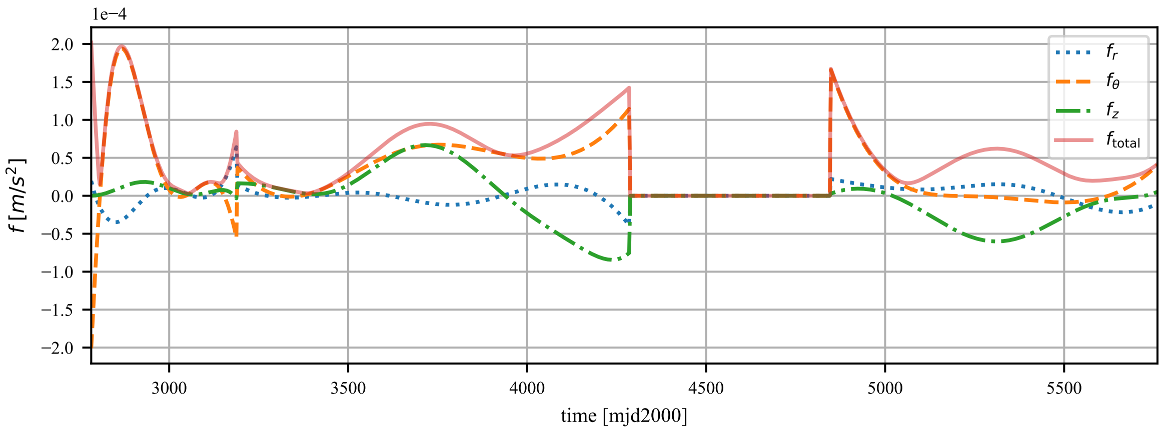


Figure D.3: Thrust acceleration profile of the optimized Dawn trajectory

The thrust profile for the best Asteroid trajectory ($\Delta V = 26.8\,\mathrm{km\,s^{-1}}$) is shown in Figure D.4. Here, the maximum thrust acceleration is necessary during the first transfer, reaching $20 \times 10^{-4}\,\mathrm{m\,s^{-2}}$, which is above the capabilities of current low-thrust engines. After the departure from the first asteroid the thrust acceleration stays below $6 \times 10^{-4}\,\mathrm{m\,s^{-2}}$. Further considering this trajectory would therefore need subsequent local optimization including a thrust constraint, for example using a direct Sims-Flanagan transcription [3, 43]. The shaping solution can then serve as an initial guess.

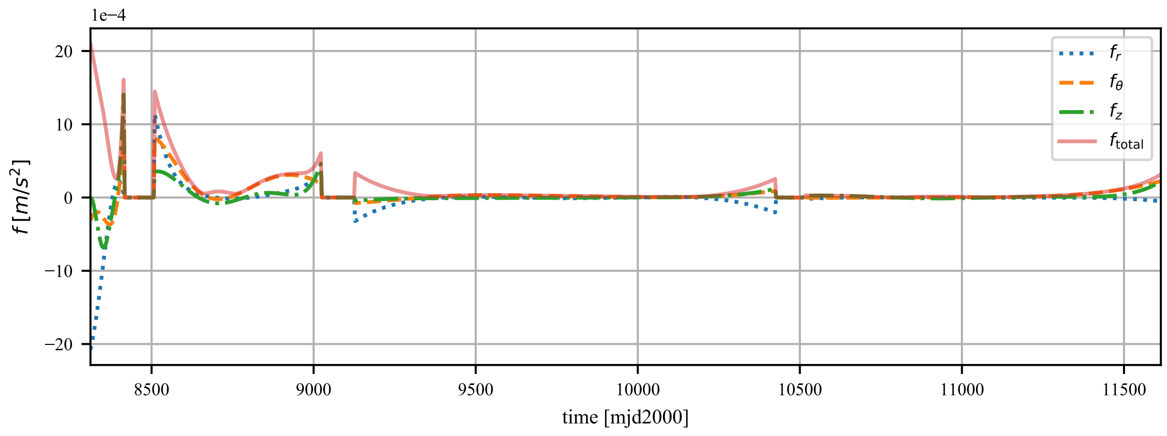


Figure D.4: Thrust acceleration profile of the optimized Asteroid trajectory

D.3. Details on the algorithm's computational effort

The computational effort between different algorithms are difficult to compare due to wildly different processor speeds, hardware acceleration using for example the GPU, and levels of algorithmic parallelization. Here, an overview is given of the computational effort of the algorithm's building blocks. The unit used is processor time as measured by Python's built-in `time.process_time()` function. Computational experiments were conducted on a laptop as well as a server provided by the Astrodynamics and Space Missions department. The laptop's Central Processing Unit (CPU) is an Intel Core i7-7700HQ@2.8 GHz (4 cores and 8 threads) and the GPU is an Nvidia Quadro M1200. Using the laptop it makes a large difference if a single or multiple cores are used. Intel processors are able to "turbo boost", which is a temporary increase in clock speed. This is used for high loads on single threads as otherwise the thermal protection mechanisms kick in and reduce clock speed again. The Core i7 is therefore able to run at 3.8 GHz for single thread processes but runs at 2.6 GHz to 2.8 GHz if all cores are being used. The faculty server is equipped with two Intel Xenon E5-2683@2.0 GHz (14 cores and 28 threads each). As it is constantly used by multiple users, turbo boosting does not happen. None of the computations are heavy on the RAM.

Hodographic shaping The biggest share of computational effort is taken by the numerical integration performed in the $\theta(t)$ computation. The initial implementation used scipy's `quad` function, providing an interface to the Fortran QUADPACK library which is focused on high-accuracy, automatic integration [34]. Computing the ΔV of a single transfer takes (33 ± 9) ms, when computing 10^5 trajectories on a single thread on the laptop. Changing the integration method to a vectorized trapezoidal method with 25 steps causes the results to differ less than 1%, as tested on 4 representative unit test cases, but decreased computation by an order of magnitude to (1.4 ± 0.2) ms. The computation time is increased when the maximum thrust acceleration is computed. The implemented method uses a grid search of 1000 equally spaced samples of the thrust acceleration profile and reports the maximum. This takes about 4 ms per evaluation.

Pygmo's GA Here, computation times for a realistic optimization case is shown. The Dawn problem included three inner optimization problems of finding the 6 free parameters for each shaped transfer. Computing this fitness function as the only running thread on the laptop takes on average 2.3 s. Running the same optimization in parallel and using all processor cores, which was done to generate the averaged convergence plots, increase the computation time to 3.3 s. Running the optimization on the server requires about 5.5 s per fitness computation.

ANN training The time needed for network training is the most difficult to quantify as it varies in a wide range. Small ANNs trained on small data sets are fully trained in a matter of seconds while there is no real upper limit for deep networks trained on huge data sets. For small datasets, i.e. the online surrogate creation, it was found to be more efficient to train on the CPU. The overhead of moving data to the GPU is not worth it in this case. Using the CPU also allows the running of surrogate assisted optimizations on the server, which is not equipped with a GPU. Training the 3/64 architecture on a small data set of 250 samples takes about 8.5 s using 4 processes on the laptop. Training on a larger data set of 1500 samples takes 77 s until early stopping. Both values are the average of 5 runs. Incremental training is slightly faster, taking 5 s to 60 s. Training on the server is proportionally slower. ANN evaluation is very fast with about 120 µs for an average network size of 3/64 on the laptop's CPU. This value depends on the architecture but not on the size of the data set. Pretraining the Dawn case was done using a data set of 14 900 samples resulting in CPU training times of 13 min (average of 3). Pretraining in the Asteroid case used a much larger data sets of 438 526 samples, making it efficient to switch to GPU training. Training then took 20 min.

D.4. Dawn optimization problem

Here, the results of a number of independent optimization runs are listed and compared to the set bounds of the Dawn problem, see Table D.1. It can be seen that the found optima are far away from the boundaries, showing that the bounds are chosen in a sensible way and do not the restrict the solution quality. The only parameters that get close to a boundary are the Times of Flight 2, 3, and 4. This behavior makes sense as longer flight times have a negative correlation to the necessary ΔV. However, the bounds were kept at the shown values in order to keep the total flight time to a reasonable level.

D.5. Asteroid optimization problem

The same data showing the found solutions and the problem bounds is given for the Asteroid problem, see Tables D.2 and D.3. The overview plot of the unassisted GA on the Asteroid problem was omitted in the paper due to space constraints and is included here, see Figure D.5. It can be seen that the GA converges especially well for a population size of 50 and a mutation probability of 10 %.

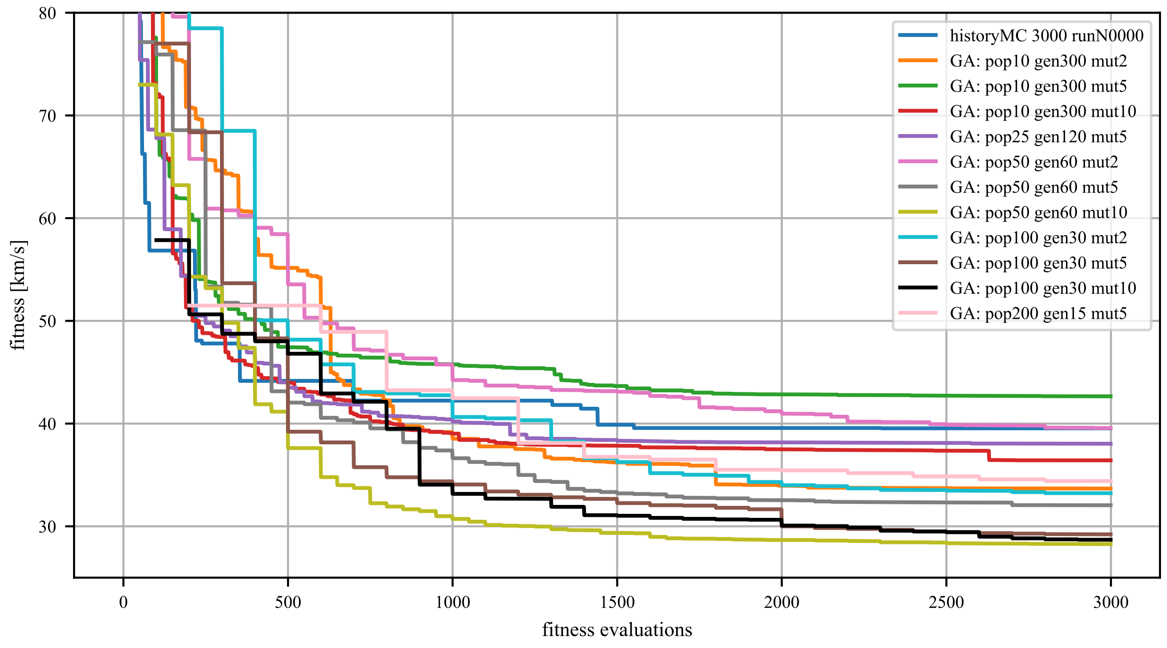


Figure D.5: Convergence of the classic GA with different settings on the Asteroid problem (average of 5 runs each)

Table D.1: GA Optimization results and bounds of the Dawn problem

Index	Date	ToF 1	ToF 2	ToF 3	ToF 4	V_∞	V_r	V_t	V_z	r_p	β	ΔV	run
	[mjd2000]	[days]	[days]	[days]	[days]	[days]	[m s^{-1}]	[m s^{-1}]	[m s^{-1}]	[m]	[deg]	[m s^{-1}]	
lower	2500.00	300.00	700.00	200.00	700.00	1500.00	-5000.00	10000.00	-5000.00	3.55e+06	0.00	—	bounds
upper	3500.00	700.00	1100.00	600.00	1100.00	5000.00	5000.00	50000.00	5000.00	1.34e+07	360.00	—	bounds
2935	2782.22	407.59	1096.42	559.19	912.69	2423.78	-2205.84	21678.38	-63.74	3.87e+06	155.26	1.2e+04	pop10 gen300 mut5 run2
3011	2790.72	408.31	1044.45	599.94	891.33	2904.92	-2187.15	22175.65	191.38	8.43e+06	195.78	1.2e+04	pop50 gen60 mut5 run3
3063	2791.69	421.21	1054.67	583.33	927.77	2962.10	-2315.45	22531.54	587.64	6.40e+06	189.02	1.2e+04	pop100 gen30 mut5 run2
3093	2800.88	415.52	1000.92	591.14	967.07	3648.19	-2927.64	22423.99	235.32	4.65e+06	160.19	1.2e+04	pop200 gen15 mut5 run4
3009	2737.80	469.41	1097.07	534.48	934.32	1611.51	-1809.89	22179.78	132.75	6.18e+06	192.68	1.3e+04	pop10 gen300 mut5 run5
2984	2746.26	456.61	1077.18	569.36	941.64	1621.30	-2027.47	21975.86	-588.07	4.78e+06	142.35	1.3e+04	pop50 gen60 mut5 run1
2908	2769.93	458.41	1039.03	552.46	937.80	2105.85	-2411.99	23273.59	459.39	6.61e+06	182.53	1.3e+04	pop100 gen30 mut5 run5
3030	2772.33	435.31	1078.18	479.48	1028.81	2192.08	-2099.85	22048.94	647.51	5.15e+06	183.15	1.3e+04	pop200 gen15 mut5 run2
3075	2764.74	427.96	1081.65	529.90	971.47	2001.84	-2184.62	21603.48	-813.99	4.18e+06	31.93	1.3e+04	pop200 gen15 mut5 run1
2945	2767.69	398.45	1076.18	596.46	923.13	1923.02	-1246.18	21502.82	-331.61	7.80e+06	212.40	1.3e+04	pop10 gen300 mut5 run1
3049	2731.87	496.98	1042.44	572.35	915.06	1592.76	-1605.47	23002.45	19.55	1.29e+07	192.71	1.3e+04	pop50 gen60 mut5 run4
2905	2804.16	416.63	1042.48	578.06	929.22	3394.95	-2332.91	23817.31	501.99	7.33e+06	195.14	1.3e+04	pop10 gen300 mut5 run4
2993	2804.44	407.86	1065.69	554.73	925.27	3429.00	-2315.91	23806.32	277.09	1.09e+07	207.16	1.3e+04	pop50 gen60 mut5 run5
3100	2777.80	443.87	978.85	583.77	761.01	2819.98	-1780.54	22921.38	-254.55	1.03e+07	208.72	1.3e+04	pop200 gen15 mut5 run5
2621	2800.66	408.98	1037.11	597.75	917.74	3418.10	-1397.98	24148.36	408.55	1.32e+07	176.24	1.3e+04	pop100 gen30 mut5 run1
3013	2796.64	429.23	1090.63	530.28	863.95	3197.28	-2070.55	24593.16	622.30	6.19e+06	196.35	1.4e+04	pop50 gen60 mut5 run2
3098	2814.33	434.21	982.63	599.06	816.53	3392.10	-3214.73	23772.79	-795.73	3.60e+06	3.38	1.4e+04	pop100 gen30 mut5 run3
3009	2773.90	519.08	906.16	597.90	938.15	2311.26	-2680.88	24723.49	-501.23	3.98e+06	28.18	1.4e+04	pop10 gen300 mut5 run3
3084	2798.84	458.81	1015.69	546.85	882.08	3623.97	-1124.30	25456.11	395.82	1.17e+07	168.72	1.4e+04	pop100 gen30 mut5 run4
3021	2745.59	496.97	1028.06	573.21	1085.49	2653.06	-1071.25	22816.57	-189.16	5.60e+06	192.87	1.4e+04	pop200 gen15 mut5 run3
501	2779.62	470.10	886.67	584.15	763.65	2392.45	-1446.54	21837.90	-1681.11	7.17e+06	313.71	1.7e+04	Random iter3000 run2
2963	2795.45	385.20	1092.72	218.80	1054.68	2635.95	-291.65	21774.05	-1588.32	1.29e+07	253.19	1.7e+04	Random iter3000 run1
2655	2793.55	420.26	1000.72	409.06	843.46	3645.56	-1515.06	25666.63	1891.51	8.21e+06	126.74	1.9e+04	Random iter3000 run3
2350	2730.07	510.87	1066.33	489.11	719.05	4494.21	839.67	25176.37	-1184.62	1.34e+07	350.81	1.9e+04	Random iter3000 run5
2761	2771.81	555.87	880.88	380.34	1030.58	3123.00	-1376.68	24700.38	1250.22	1.32e+07	234.23	1.9e+04	Random iter3000 run4

Table D.2: GA Optimization results and bounds of the Asteroid problem

Index	Date [mjd2000]	ToF 1 [days]	Stay 1 [days]	ToF 2 [days]	Stay 2 [days]	ToF 3 [days]	Stay 3 [days]	ToF 4 [days]	V_∞ [m s^{-1}]	φ [deg]	θ_1 [deg]	ΔV [m s^{-1}]	run
lower	5479.00	100.00	90.00	200.00	90.00	400.00	90.00	600.00	0.00	0.00	0.00	—	bounds
upper	9129.00	1700.00	110.00	1900.00	110.00	1900.00	110.00	2000.00	3500.00	180.00	360.00	—	bounds
3039	8311.52	106.85	90.02	517.15	102.09	1299.48	90.00	1096.61	3222.92	125.52	166.68	2.7e+04	pop50 gen60 mut10 run1
3050	8294.32	101.87	92.27	522.42	91.68	1197.60	96.94	1350.33	2136.77	127.00	157.67	2.7e+04	pop100 gen30 mut5 run1
3009	8283.72	105.16	90.21	554.68	107.44	1242.23	93.43	1162.65	1962.55	133.86	215.09	2.7e+04	pop10 gen300 mut10 run1
3007	8272.89	112.08	90.46	563.83	104.97	1300.21	90.33	1071.80	1465.95	146.24	175.29	2.8e+04	pop50 gen60 mut5 run1
3056	8271.49	114.11	93.32	555.72	101.84	1259.22	95.95	1112.17	1931.98	135.39	234.75	2.8e+04	pop100 gen30 mut10 run2
2871	8013.07	339.29	99.42	598.95	92.99	1263.87	90.03	1122.37	1960.42	115.06	129.48	2.8e+04	pop10 gen300 mut5 run2
3029	8244.46	135.25	90.40	535.11	109.36	1232.80	90.81	1216.97	646.03	136.93	133.35	2.8e+04	pop50 gen60 mut10 run2
2986	8131.21	266.43	90.03	516.38	109.70	1234.79	90.63	1231.77	2056.75	13.71	266.34	2.8e+04	pop10 gen300 mut10 run3
3007	8173.78	225.08	93.99	552.70	100.76	1199.00	90.40	1185.22	2330.72	13.02	273.92	2.9e+04	pop100 gen30 mut10 run3
2760	8208.73	185.65	92.22	510.20	108.68	1269.56	92.67	1133.73	1384.89	8.75	108.72	2.9e+04	pop50 gen60 mut10 run5
3033	8110.30	268.40	90.41	526.98	107.16	1269.55	90.38	1150.68	1418.56	22.93	111.37	2.9e+04	pop50 gen60 mut10 run3
3039	8231.36	153.16	90.26	558.36	108.05	1199.03	92.93	1239.56	422.50	75.68	119.81	2.9e+04	pop100 gen30 mut5 run3
3009	8198.56	202.80	90.84	561.97	105.39	1317.29	90.91	1060.05	1724.88	3.74	100.60	2.9e+04	pop10 gen300 mut2 run3
3024	8205.08	172.51	90.06	577.61	109.97	1220.79	91.03	1149.99	629.23	11.96	206.27	2.9e+04	pop25 gen120 mut5 run4
3025	8222.57	180.21	90.24	501.90	105.98	1295.14	90.36	1147.04	1422.19	50.56	263.22	2.9e+04	pop100 gen30 mut10 run1
2916	8280.69	119.79	98.00	564.52	96.64	1230.97	106.63	1057.76	1222.26	96.31	202.60	2.9e+04	pop100 gen30 mut10 run4
2910	8585.56	260.27	92.19	937.23	108.37	1180.25	103.66	688.93	2142.23	80.71	219.01	2.9e+04	pop100 gen30 mut10 run5
3067	8104.65	285.87	91.71	549.32	104.06	1296.31	94.25	1072.66	1176.28	9.50	198.48	2.9e+04	pop100 gen30 mut2 run4
2894	8571.76	239.24	109.75	877.62	108.81	1271.86	95.45	690.10	1782.27	45.06	150.38	2.9e+04	pop50 gen60 mut10 run4
2933	8534.94	288.83	105.29	917.95	109.47	1240.01	100.34	631.17	1595.83	35.75	202.78	2.9e+04	pop50 gen60 mut5 run3
3027	8203.98	179.83	90.09	604.92	90.95	1201.00	94.32	1144.42	446.34	13.64	139.22	2.9e+04	pop50 gen60 mut2 run4
2964	8128.78	262.35	90.20	620.55	91.10	1316.63	91.37	983.17	1729.24	17.43	278.80	2.9e+04	pop50 gen60 mut2 run1
3009	8074.23	288.96	90.19	591.68	102.11	1237.51	90.23	1156.58	2799.37	81.80	101.56	2.9e+04	pop10 gen300 mut10 run4
3025	8235.99	161.21	99.92	520.92	95.95	1301.25	106.47	1075.07	636.03	31.90	173.60	2.9e+04	pop100 gen30 mut5 run4
3005	8106.75	260.74	90.51	609.67	94.38	1317.92	94.90	1006.53	1534.00	53.92	106.06	2.9e+04	pop100 gen30 mut5 run5
3109	8184.24	178.44	94.80	522.63	91.23	1277.14	103.92	1184.68	414.71	64.93	98.52	3e+04	pop200 gen15 mut5 run3
3009	8647.71	192.02	95.40	858.87	109.01	1267.23	104.58	633.13	3354.29	104.65	179.94	3e+04	pop10 gen300 mut2 run1
2617	8597.48	235.42	108.87	921.14	101.06	1197.77	99.62	700.67	1188.41	91.11	224.85	3e+04	pop200 gen15 mut5 run1
3009	8075.46	287.53	90.42	636.36	91.07	1182.76	90.90	1195.77	1615.34	77.95	249.03	3e+04	pop10 gen300 mut5 run3
3039	8574.88	285.19	90.17	750.61	109.53	1282.76	105.37	702.75	2088.08	57.45	206.99	3e+04	pop100 gen30 mut2 run3
3099	8275.69	140.99	90.15	644.97	103.09	1297.13	109.98	980.73	1478.84	129.50	184.05	3e+04	pop100 gen30 mut2 run5
3040	8202.97	179.55	91.94	653.95	91.18	1162.77	92.70	1161.94	2123.14	47.72	85.35	3e+04	pop100 gen30 mut5 run2
3079	8168.56	180.91	105.01	524.81	109.48	1236.00	90.16	1223.59	1137.65	23.90	225.04	3e+04	pop100 gen30 mut2 run2

Table D.3: GA Optimization results and bounds of the Asteroid problem (continued)

Index	Date [mjd2000]	ToF 1 [days]	Stay 1 [days]	ToF 2 [days]	Stay 2 [days]	ToF 3 [days]	Stay 3 [days]	ToF 4 [days]	V_∞ [m s^{-1}]	φ [deg]	θ_1 [deg]	ΔV [m s^{-1}]	run
lower	5479.00	100.00	90.00	200.00	90.00	400.00	90.00	600.00	0.00	0.00	0.00	—	bounds
upper	9129.00	1700.00	110.00	1900.00	110.00	1900.00	110.00	2000.00	3500.00	180.00	360.00	—	bounds
3009	8162.03	243.51	109.13	1030.58	109.76	1291.47	109.68	675.07	2201.43	2.20	314.77	3.2e+04	pop10 gen300 mut2 run4
2961	8922.69	265.70	109.65	623.47	105.00	1179.26	104.11	626.69	2525.74	58.73	149.86	3.3e+04	pop50 gen60 mut5 run5
3019	8220.54	229.03	90.16	1115.92	109.81	1251.98	109.70	716.91	1308.08	11.39	242.92	3.3e+04	pop50 gen60 mut5 run4
2819	8182.49	296.37	90.19	1230.20	109.74	1247.77	109.94	658.13	1119.08	3.78	182.18	3.3e+04	pop25 gen120 mut5 run1
3125	7778.52	300.16	95.14	1435.42	109.93	1225.41	90.05	681.18	2109.23	41.60	331.70	3.5e+04	pop200 gen15 mut5 run2
3112	7684.29	361.16	95.12	1307.62	108.38	1313.06	98.58	716.42	552.31	100.81	248.59	3.6e+04	pop200 gen15 mut5 run5
3004	7852.06	186.56	97.19	1286.11	109.76	1263.66	107.48	763.57	3043.15	65.88	58.03	3.7e+04	pop50 gen60 mut5 run2
2904	7871.04	237.26	90.41	1511.62	109.74	1243.92	109.23	634.87	2840.82	70.98	69.82	3.8e+04	pop10 gen300 mut2 run2
3199	7864.31	234.97	90.30	1463.28	96.15	1267.73	97.40	694.78	2662.49	54.92	302.05	3.8e+04	pop200 gen15 mut5 run4
2865	7895.49	222.86	91.04	1543.63	109.75	1250.54	109.64	622.43	463.33	43.35	8.08	3.9e+04	pop25 gen120 mut5 run5
3044	7900.91	190.07	91.63	1457.77	109.01	1274.76	107.97	704.28	1002.66	93.12	14.41	3.9e+04	pop50 gen60 mut2 run2
2833	8734.22	647.46	109.78	1093.83	90.13	556.63	90.21	672.18	3399.33	96.46	8.74	4e+04	pop10 gen300 mut2 run5
3009	7826.83	307.17	90.08	1579.63	109.96	1244.02	109.90	606.70	964.63	32.96	10.22	4e+04	pop10 gen300 mut5 run5
3002	8744.77	313.81	104.31	1556.12	90.27	442.73	91.78	623.38	1829.55	106.59	179.10	4.4e+04	pop25 gen120 mut5 run3
3024	8812.42	263.62	90.08	1564.32	90.86	412.05	90.32	606.46	2147.38	56.60	211.15	4.5e+04	pop25 gen120 mut5 run2
3085	7465.10	207.34	107.66	1555.89	109.60	1324.37	103.15	800.05	3489.29	49.50	339.84	4.6e+04	pop100 gen30 mut2 run1
3043	8801.59	591.68	101.41	1035.81	90.05	565.68	96.87	635.53	3490.69	87.62	11.87	4.7e+04	pop50 gen60 mut2 run3
2945	8818.16	580.81	96.34	1112.85	90.14	502.65	93.41	602.51	3491.69	84.09	358.74	4.9e+04	pop10 gen300 mut10 run2
3009	7584.60	187.22	90.10	1881.78	109.44	1258.50	110.00	609.73	3066.79	122.79	286.85	4.9e+04	pop10 gen300 mut10 run5
3017	6835.56	273.47	90.31	1299.69	109.29	1014.32	107.91	1614.82	2188.35	110.60	8.75	5.3e+04	pop50 gen60 mut2 run5
2736	6570.98	422.71	91.10	395.62	90.53	743.79	90.46	1986.94	926.49	79.42	341.85	5.4e+04	pop10 gen300 mut5 run1
3009	5666.47	423.15	90.20	1019.81	109.87	738.05	90.03	1992.81	2969.45	90.32	340.61	6.1e+04	pop10 gen300 mut5 run4

Bibliography

[1] Ossama Abdelkhalik and Ehsan Taheri. Shape based approximation of constrained low-thrust space trajectories using fourier series. *Journal of Spacecraft and Rockets*, 49(3):535–546, 2012.

[2] Christos Ampatzis and Dario Izzo. Machine learning techniques for approximation of objective functions in trajectory optimisation. In *IJCAI-09 Workshop on Artificial Intelligence in Space*, pages 1–6, 2009.

[3] John T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[4] Francesco Biscani and Dario Izzo. Esa/pagmo2: Version 2.10, 2019. `https://doi.org/10.5281/zenodo.2529931`.

[5] Francesco Biscani, Dario Izzo, and Chit Hong Yam. A global optimisation toolbox for massively parallel engineering optimisation. *ICATT 2010: International Conference on Astrodynamics Tools and Techniques*, 2010.

[6] François Chollet et al. Keras. `https://keras.io`, 2015. Version 2.2.4.

[7] Howard D. Curtis. *Orbital Mechanics for Engineering Students*. Butterworth-Heinemann, 2005.

[8] Bram De Vogeleer. Automatic and fast generation of sub-optimal and feasible low-thrust trajectories using a boundary-value pseudo-spectral method. *M.Sc. Thesis, TU Delft*, 2008.

[9] William M. Folkner, James G. Williams, Dale H. Boggs, Ryan S. Park, and Petr Kuchynka. The planetary and lunar ephemerides de430 and de431. *Interplanetary Network Progress Report*, 196: 1–81, 2014.

[10] P. Gage, I. Kroo, and R. Braun. Interplanetary trajectory optimization using a genetic algorithm. In *Astrodynamics Conference*, page 3773, 1994.

[11] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[12] David J. Gondelach. A hodographic-shaping method for low-thrust trajectory design. *M.Sc. Thesis, TU Delft*, 2012.

[13] David J. Gondelach and Ron Noomen. Hodographic-shaping method for low-thrust interplanetary trajectory design. *Journal of Spacecraft and Rockets*, 52(3):728–738, 2015.

[14] N.S. Gopinath and K.N. Srinivasamuthy. Optimal low thrust orbit transfer from GTO to geosynchronous orbit and stationkeeping using electric propulsion system. In *54th International Astronautical Congress*, pages A–7, 2003.

[15] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd., 2017.

[16] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[18] Dario Izzo. Global optimization and space pruning for spacecraft trajectory design. In Bruce A. Conway, editor, *Spacecraft Trajectory Optimization*, chapter 7, pages 178–201. Cambridge University Press, 2010.

[19] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[20] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations*, 2014.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[24] Shuang Li, Xuxing Huang, and Bin Yang. Review of optimization methodologies in global and china trajectory optimization competitions. *Progress in Aerospace Sciences*, 2018.

[25] David Morante, Manuel Sanjurjo Rivo, and Manuel Soler. Multi-objective low-thrust interplanetary trajectory optimization based on generalized logarithmic spirals. *Journal of Guidance, Control, and Dynamics*, 42(3):476–490, 2018.

[26] Paul Musegaas. Optimization of space trajectories including multiple gravity assists and deep space maneuvers. *M.Sc. Thesis, TU Delft*, 2013.

[27] Michael A. Nielsen. *Neural networks and deep learning*. Determination press, 2015.

[28] Gustavo Montes Novaes et al. Combining polynomial chaos expansions and genetic algorithm for the coupling of electrophysiological models. In *International Conference on Computational Science*, pages 116–129. Springer, 2019.

[29] Daniel M. Novak and Massimiliano Vasile. Improved shaping approach to the preliminary design of low-thrust trajectories. *Journal of Guidance, Control, and Dynamics*, 34(1):128–147, 2011.

[30] Paolo De Pascale and Massimiliano Vasile. Preliminary design of low-thrust multiple gravity-assist trajectories. *Journal of Spacecraft and Rockets*, 43(5):1065–1076, 2006.

[31] Prashant R. Patel, Alec D. Gallimore, Thomas H. Zurbuchen, and Daniel J. Scheeres. An algorithm for generating feasible low thrust interplanetary trajectories. In *Proceedings of theInternational Electric Propulsion Conference, Michigan, USA*, 2009.

[32] Anastassios E. Petropoulos and James Longuski. Automated design of low-thrust gravity-assist trajectories. In *Astrodynamics Specialist Conference*, page 4033, 2000.

[33] Anastassios E. Petropoulos and James M. Longuski. Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories. *Journal of Spacecraft and Rockets*, 41(5):787–796, 2004.

[34] Robert Piessens, Elise de Doncker-Kapenga, Christoph W. Überhuber, and David K. Kahaner. *Quadpack: A subroutine package for automatic integration*, volume 1. Springer Science & Business Media, 2012.

[35] Marc D. Rayman, Philip Varghese, David H. Lehman, and Leslie L. Livesay. Results from the Deep Space 1 technology validation mission. *Acta Astronautica*, 47(2-9):475–487, 2000.

[36] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[37] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[38] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[39] Erik Saturnino. A piecewise shaping method for preliminary low-thrust trajectory design. *M.Sc. Thesis, TU Delft*, 2017.

[40] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[41] Abolfazl Shirazi, Josu Ceberio, and Jose A. Lozano. Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions. *Progress in Aerospace Sciences*, 102:76–98, 2018.

[42] Marsil A.C. Silva, Daduí C. Guerrieri, Angelo Cervone, and Eberhard Gill. A review of MEMS micropropulsion technologies for CubeSats and PocketQubes. *Acta Astronautica*, 143:234–243, 2018.

[43] Jon A. Sims and Steve N. Flanagan. Preliminary design of low-thrust interplanetary missions. In *SAAS/AIAA Astrodynamics Specialist Conference, Girdwood, Alaska*, pages 1–9, 1999.

[44] Leon Stubbig. AE4020 Literature Study: Optimization of Low-Thrust Trajectories using shaping, neural networks and Evolutionary Algorithms. *TU Delft*, 2019.

[45] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. `http://arxiv.org/abs/1605.02688`, 2016. Version 1.0.3.

[46] David A. Vallado. *Fundamentals of Astrodynamics and Applications*, volume 12. Springer Science & Business Media, 2001.

[47] Karel F. Wakker. *Fundamentals of astrodynamics*. TU Delft Library, 2015.

[48] Bradley J. Wall and Bruce A. Conway. Shape-based approach to low-thrust rendezvous trajectory design. *Journal of Guidance, Control, and Dynamics*, 32(1):95–101, 2009.

[49] Bradley J. Wall and Bruce A. Conway. Genetic algorithms applied to the solution of hybrid optimal control problems in astrodynamics. *Journal of Global Optimization*, 44(4):493, 2009.

[50] Abby Weeks. Interplanetary trajectory optimization using a genetic algorithm. *Journal of the Astronautical Sciences*, 43(1):59–75, 1995.

[51] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[52] Jun Zhang et al. Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine*, 6(4):68–75, 2011.