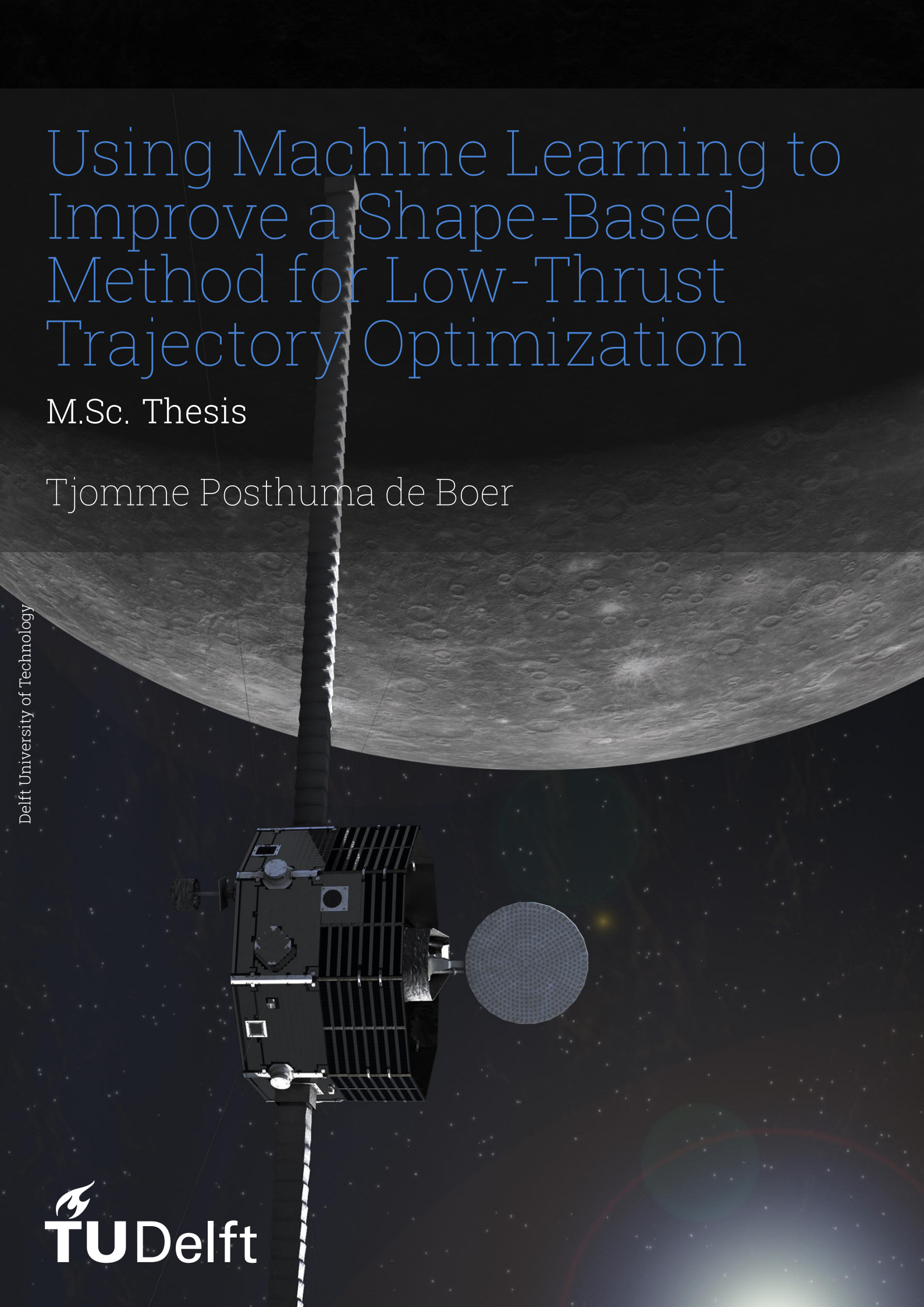
Using Machine Learning to Improve a Shape-Based Method for Low-Thrust Trajectory Optimization

M.Sc. Thesis

Tjomme Posthuma de Boer



Delft University of Technology

TU Delft

Using Machine Learning to Improve a Shape-Based Method for Low-Thrust Trajectory Optimization

M.Sc. Thesis

by

Tjomme Posthuma de Boer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on November 22, 2023

Student number: 4460227

Thesis committee:		
	Prof.dr.ir. P.N.A.M. Visser,	Chair
	Ir. K. J. Cowan MBA,	Supervisor
	Dr. A. Menicucci,	External examinator

Cover: BepiColombo over Mercury, credit: JAXA
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This thesis is submitted to obtain the degree of Master of Science at the Faculty of Aerospace Engineering of the Delft University of Technology. It marks the end of my time as a student and, therefore, concludes this chapter of my life. I want to thank everyone who supported me over the past year and a half when I sometimes felt the literature study and the thesis work were never-ending. All things, however, come to an end. A special thanks goes to my supervisor, Kevin Cowan, whose relentless enthusiasm and excellent guidance motivated me throughout the project and helped me reach this end. I can now proudly present this report describing my thesis research. Enjoy.

Tjomme Posthuma de Boer
The Hague, November 2023

Abstract

Shape-based methods are used in the preliminary optimization of low-thrust trajectories to rapidly search large design spaces and provide initial guesses for higher fidelity methods. The optimization process benefits from the shape-based methods providing initial guesses as quickly and as optimal as possible. This work aims to improve the hodographic shaping method by implementing machine learning (ML) to optimize its free parameters. The addition of free parameters enables a more optimal trajectory, while the ML model reduces the computational effort required to optimize this trajectory. ML-incorporated shaping models with 6 and 9 Degrees of Freedom (DoF) and models with different levels of mission generalization are built. The models are tested and compared to the shaping method without ML on different types of single transfers in a grid search and on more complex missions in a genetic algorithm. The ML 6-DoF models can consistently find trajectories that are more optimal compared to the 0-DoF models without ML, by up to 6 km/s for an Earth-Ceres transfer, and they can do so 50-100 times faster than the 6-DoF models without ML. The 9-DoF models produce inconsistent results and can only improve no-ML shaping in areas of low optimality. The ML models significantly accelerate the performance of the genetic algorithm when the relevant transfer bodies are included in the ML training data.

Contents

Nomenclature

List of Abbreviations

Abbreviation	Definition
2-D	2-Dimensional
3-D	3-Dimensional
AAS	American Astronautical Society
Adam	Adaptive Moment Estimation
AU	Astronomical Unit
BOBYQA	Bound Optimization BY Quadratic Approximation
cobyla	Constrained optimization by linear approximation
CPU	Central Processing Unit
DE	Differential Evolution
DoF	Degrees of Freedom
ECLIPJ2000	Ecliptic coordinates based on the J2000 frame
ESA	European Space Agency
GA	Genetic Algorithm
GPU	Graphics Processing Unit
GTOC	Global Trajectory Optimization Competition
JAXA	Japan Aerospace Exploration Agency
LB	Lower Bound
LR	Learning Rate
LTP	Low-Thrust Propulsion
MJD2000	Modified Julian Date 2000
ML	Machine Learning
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NN	Neural Network
N-M	Nelder-Mead
PaGMO	Parallel Global Multiobjective Optimization
ReLU	Rectified Linear Unit
sbplx	Subplex
SGA	Simple Genetic Algorithm
Sklearn	Scikit-learn
SPICE	Spacecraft Planet Instrument Camera-matrix Events
ToF	Time of Flight
TU	Technical University
Tudat	TU Delft Astrodynamics Toolbox
UB	Upper Bound

List of Symbols

Symbol	Definition	Unit
a	Neuron activation matrix	[-]
a_i	Actual values loss function	[-]
a_1	Departure body semi-major axis	[AU]
a_2	Arrival body semi-major axis	[AU]
b	Biases matrix	[-]
c_i	Hodographic shape coefficient	[-]
d_0	Departure date	[mjd2000]
e	Coordinate system axis	[-]
f	Thrust acceleration	[m/s^2]
f_i	Predicted values loss function	[-]
i, j, k	Counters	[-]
l	Neural network layer	[-]
N	Number of revolutions	[-]
n	Number of samples/functions	[-]
r	Radial direction	[-]
r, θ, z	Cylindrical coordinates	[-]
t	Time	[s]
V	Velocity	[m/s]
V_∞	Injection velocity	[m/s]
w	Weights matrix	[-]
x, y, z	Cartesian coordinates	[-]
z	Axial direction	[-]
ΔV	Velocity increment	[m/s]
θ	Tangential direction	[-]
θ_1	Spherical polar angle	[rad]
μ	Gravitational parameter	[m^3/s^2]
σ	Activation function	[-]
ϕ	Phase angle	[rad]
ϕ_1	Spherical azimuthal angle	[rad]
$\square_0$	Initial value	[-]
$\square_f$	Final value	[-]
$\dot{\square}$	First derivative with respect to time	[-]
$\ddot{\square}$	Second derivative with respect to time	[-]
$\tilde{\square}$	First antiderivative with respect to time	[-]

1 Introduction

To increase the scientific return of space exploration, more complex missions are undertaken, which can involve orbiters, landers, or visits to multiple bodies. Not only do these missions require larger and heavier spacecraft to harbor more scientific instrumentation, but they typically also need to meet more challenging propulsion requirements [1]. To reconcile these requirements with the ever-present need to minimize cost, using highly efficient Low-Thrust Propulsion (LTP) systems is a possible solution. Electric propulsion is the most commonly used LTP system. It has successfully been used in NASA's Deep Space 1 [2] and Dawn [3] missions, and it is currently used in ESA's and JAXA's BepiColombo mission on its way to Mercury [4]. The efficiency of LTP could be used to make missions achievable that would otherwise not be possible. Because of its low thrust, however, the engine must operate for extended periods of time to achieve the required velocity increments. Thrusting by a chemical propulsion system is typically modeled as an impulsive shot during trajectory optimization, meaning that only a limited number of variables must be optimized. Low-thrust optimization requires continuous steering and thus continuous optimization throughout the entire trajectory and is, therefore, a much more complex and challenging problem [5].

Low-thrust trajectory optimization can be modeled as a continuous optimal control problem. For this problem, the control law and its corresponding trajectory must be determined while minimizing a cost function. Furthermore, the trajectory must satisfy boundary conditions, such as the departure and arrival states, and constraints, such as maximum available thrust [6]. Several numerical approaches exist to solve this problem, but they are computationally expensive and need a good initial guess to converge [7]. Shape-based methods describe the trajectory analytically by satisfying the boundary conditions, after which the spacecraft dynamics are obtained from assessing this shape. This avoids numerical iteration and makes these methods relatively fast. However, to make it possible to describe the problem analytically, the accuracy of the trajectory is significantly reduced. As the shape-based methods are able to quickly scan large design spaces in the preliminary design phase, their solution is often used as an initial guess for the numerical methods[8]. The time it takes for the numerical methods to converge will be shorter when the solution of the shape-based method is closer to the true optimum [9]. The overall trajectory design therefore benefits from the solution of the shape-based method being as optimal as possible while remaining quick.

A separate method that has made recent advancements in many fields is that of Machine Learning (ML). ML can provide the best solution already in some fields of research, such as image detection and speech recognition [10]. Its application to trajectory optimization is still relatively new but brings many promising opportunities [11]. Trajectory optimization processes can have elaborate and computationally expensive cost functions. The powerful regression capabilities of ML models can approximate these functions, significantly reducing the computational cost of these procedures [12]. A field within trajectory optimization where ML has yet to be applied directly is that of shape-based methods. The goal of this research is to examine if the performance of a shape-based method can improve through the implementation of machine learning. This is examined by implementing an artificial Neural Network (NN) in the optimization of trajectory shape parameters.

1.1. Research Questions

The following research goal is pursued in this report:

To improve a shape-based method for low-thrust trajectory optimization by incorporating machine learning.

To help achieve this goal, the following main research question is formulated:

How can machine learning improve a shape-based method for low-thrust trajectory optimization?

To help answer the main research question, the following sub-questions are formulated:

- What does it mean for a shape-based method to improve?
- Is the use of the shaping method less computationally expensive when the ML model is incorporated?
- Does the shaping method lead to more optimal results when the ML model is incorporated?
- Can a single ML-incorporated shaping method be used for different transfer types?
- How does the ML-incorporated shaping method perform when implemented in a more complex preliminary mission design?

1.2. Report Structure

The report is structured as follows. A scientific paper in AAS format is presented in Chapter 2. This paper contains the methodology and the main results of this research. Chapter 3 uses these results to answer the research questions and draw conclusions. Also, the most promising steps for future research are given as recommendations. The appendices are used to supply background information on the used methods and to provide additional results. In Appendix A, the dynamical model is presented, with an emphasis on the shaping method used. Appendix B and Appendix C give background information on the optimization method and the neural network used in this research. Furthermore, Appendix D contains verification and validation of the presented models. Finally, additional results are shown in Appendix E.

2

Scientific Paper

(Preprint) AAS XX-XXX

USING MACHINE LEARNING TO IMPROVE A SHAPE-BASED METHOD FOR PRELIMINARY LOW-THRUST TRAJECTORY OPTIMIZATION

Tjomme Posthuma de Boer* **and Kevin Cowan†**

Shape-based methods are used in the preliminary optimization of low-thrust trajectories to rapidly search large design spaces and provide initial guesses for higher fidelity methods. The optimization process benefits from the shape-based methods providing initial guesses as quickly and as optimal as possible. This work aims to improve the hodographic shaping method by implementing machine learning (ML) to optimize its free parameters. The addition of free parameters enables a more optimal trajectory, while the ML model reduces the computational effort required to optimize this trajectory. ML-incorporated shaping models with 6 and 9 Degrees of Freedom (DoF) and models with different levels of mission generalization are built. The models are tested and compared to the shaping method without ML on different types of single transfers in a grid search and on more complex missions in a genetic algorithm. The ML 6-DoF models can consistently find trajectories that are more optimal compared to the 0-DoF models without ML, by up to 6 km/s for an Earth-Ceres transfer, and they can do so 50-100 times faster than the 6-DoF models without ML. The 9-DoF models produce inconsistent results and can only improve no-ML shaping in areas of low optimality. The ML models significantly accelerate the performance of the genetic algorithm when the relevant transfer bodies are included in the ML training data.

INTRODUCTION

There is a rising demand for space expeditions of increasing complexity as the space industry is rapidly expanding. In order to facilitate missions to far-away planets or missions that visit multiple bodies, an efficient propulsion system is required. Recent missions like DAWN and BepiColombo show that low-thrust propulsion can provide the desired efficiency.[1,2] Low-thrust propulsion allows for a higher velocity increment, ΔV, for the same amount of propellant mass and thus could realize missions with a higher scientific payload mass or missions that were previously not feasible.[3] The design of optimal low-thrust trajectories is a very challenging task, however. To achieve the required ΔV, the propulsion system has to run for extended periods of time, changing the optimization from a discrete problem to a continuous problem. Several numerical approaches exist to solve this problem, but they are computationally expensive and therefore time-consuming.[4]

Shape-based methods describe the trajectory analytically by satisfying the boundary conditions. After this, the dynamics of the spacecraft are assessed, avoiding numerical iteration and making these methods relatively fast. However, to make it possible to describe the problem analytically, the

*MSc. student, Faculty of Aerospace Engineering, Delft University of Technology, the Netherlands, T.K.PosthumadeBoer@student.tudelft.nl.

†Education Fellow and Lecturer, Faculty of Aerospace Engineering, Delft University of Technology, the Netherlands, K.J.Cowan@tudelft.nl

accuracy of the trajectory will be reduced. For this reason, shape-based methods are often used to scan large design spaces in the preliminary design phase, after which their solutions can be used as an initial guess for the numerical methods.[5] The time it takes for the numerical methods to converge will be shorter when the solution of the shape-based method is closer to the true optimum.[6] The overall trajectory design therefore benefits from the solution of the shape-based method being as optimal as possible while remaining quick.

In recent years, Machine Learning (ML) models have been successfully implemented in many areas, among which is trajectory optimization.[7] Some optimization processes have elaborate and computationally expensive cost functions. The powerful regression capabilities of ML models can approximate these functions, significantly improving the optimization procedure.[8] A field within trajectory optimization where ML has not been applied directly yet is that of low-thrust shape-based methods. The goal of this research is to examine if the performance of a shape-based method can improve through the implementation of machine learning. Here, an artificial Neural Network (NN) is used in the optimization of shape parameters of a trajectory using the hodographic shaping method. The addition of these parameters allows for a more optimal trajectory while the NN minimizes the computational effort that is required to calculate this trajectory. The implementation of ML will be deemed to have improved the shaping method when a combination of optimality and computational speed is achieved, which would have been unachievable without ML.

This paper is structured in the following way. First, the hodographic shaping method is discussed in order to identify the best way to implement the NN. After that, the process of generating the training data for the NN is described. Then, the machine learning model and its tuning, the feature selection, and the selection of the optimizer are shown. When the complete model is described, the test cases used to measure the performance of the model are presented. Finally, the results of the research are shown, after which the conclusion follows.

HODOGRAPHIC SHAPING

The shaping method that was selected for this research is the hodographic shaping method, created by Gondelach and Noomen.[9] This method is used to describe low-thrust trajectories between celestial bodies in cylindrical coordinates. It is capable of 3-dimensional, rendezvous transfers, and no tangential thrust is assumed, meaning a realistic thrust profile can be acquired. It was found that this method performs well with respect to other methods, such as spherical and pseudo-equinoctial shaping.[10,11] The time-driven version of hodographic shaping is implemented here using TudatPy, an astrodynamical toolbox that allows for simple integration of the shaping method with, for instance, linked trajectories or optimization algorithms.[12]

Hodographic shaping describes the shape of the trajectory in the velocity domain rather than the position domain by connecting the velocity hodographs of the departure and arrival body. For each of the three directions in cylindrical space r, θ and z, a velocity function of time $V_k(t)$ is created, which is the summation of coefficients c_i times base functions $v_i(t)$ as shown in Equation 1.

$$V_k(t) = \sum_{i=1}^{n} c_i v_i(t) \text{ where } k \in \{r, \theta, z\} \tag{1}$$

Here, t is time, and n is the number of base functions. The base functions are chosen to be simple functions that can be integrated and differentiated analytically. The position coordinates of the trajectory can then be obtained through integration, while differentiation gives the accelerations. From the accelerations, the thrust profile and the ΔV can be determined. A minimum of three base functions per direction are needed to satisfy the boundary conditions: one for the initial velocity, one for the final velocity, and one for the difference in position. The boundary condition on the initial position can then be satisfied with the constant obtained through integration.

Next to the three base functions per direction required to satisfy the boundary conditions, additional functions with coefficients can be added. These functions add Degrees of Freedom (DoF) which increase the flexibility and thus the optimality of the shape. The corresponding 'free' coefficients or parameters need to be optimized to minimize the cost function: the ΔV of the trajectory acquired through the integration of the thrust profile. This process is relatively time-consuming. Gondelach finds that a trajectory with two free parameters per direction, so 6 in total, is generally near-optimal but takes about 2 seconds to generate, while a trajectory without free parameters is not as optimal but only takes in the order of milliseconds to generate.[9]

This research assesses 0-DoF, 6-DoF, and 9-DoF trajectories. The recommended shaping functions are used for the first five base functions.[9] The sixth base function is arbitrarily chosen as no previous work on function selection exists. The radial and tangential velocities $V_{(r/\theta)}$ are given by Equation 2 and axial velocity V_z is given by Equation 3.

$$V_{(r/\theta)} = c_1 + c_2 t + c_3 t^2 + c_4 t \sin(0.5t\pi) + c_5 t \cos(0.5t\pi) + c_6 e^t \sin(0.5t\pi) \tag{2}$$

$$\begin{aligned} V_z = c_1 \cos(2\pi t(N+0.5)) + c_2 t^3 \cos(2\pi t(N+0.5)) + c_3 t^3 \sin(2\pi t(N+0.5)) \\ + c_4 t^4 \cos(2\pi t(N+0.5)) + c_5 t^4 \sin(2\pi t(N+0.5)) + c_6 e^t \sin(0.5t\pi) \end{aligned} \tag{3}$$

Where N is the number of revolutions of the trajectory. In the trigonometric functions, the time t is scaled by a factor 2π over the Time of Flight (ToF). c_4,c_5 and c_6 equal zero for a 0-DoF trajectory, c_6 equals zero for a 6-DoF trajectory. In this research, we assume the hodographic shaping method is capable of providing close-to-optimal solutions to be used as initial guesses by higher-fidelity methods, where a lower ΔV trajectory means a more optimal solution.

MODEL ARCHITECTURE

The approach of this research is as follows. The cost function of the optimization of the free parameters will be replaced by an ML algorithm, allowing for rapid optimization. The found free parameters, which are optimal according to the machine learning algorithm, will be used to find the corresponding hodographic shape. As no optimization is now required, the hodographic shaping function will almost instantly return the resulting trajectory and its corresponding ΔV. The hodographic shaping model that incorporates the ML algorithm will be called the ML model, while the model developed by Gondelach will be called the traditional or no-ML model. A 6-DoF ML model is built with the goal of finding trajectories with combinations of optimality and computational speed that would be unachievable with the traditional 0-DoF and 6-DoF models. A 9-DoF ML model is built to see if an ML model can find more optimal trajectories than the traditional 6-DoF method at a lower computational cost. An overview of the approach of the ML models is given in Figure 1.

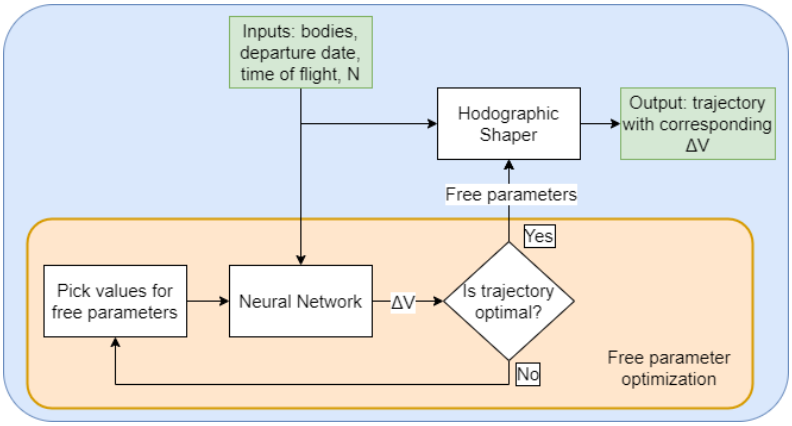


Figure 1: Overview of hodographic shaping model with incorporated ML algorithm (neural network).

TRAINING DATA GENERATION

For the ML algorithm to optimize the free parameters, it has to know which set of free parameters leads to the most optimal trajectory for a given mission. Therefore, the ML algorithm has to be trained, and training data has to be generated. The traditional hodographic shaping optimizer takes the free parameters as input, after which the ephemerides of the transfer bodies are used to set up the boundary conditions and construct the trajectory. Furthermore, the time of flight and number of revolutions of the transfer are used in the velocity functions. The corresponding ΔV is then calculated, after which the free parameters are changed, and the process repeats until the ΔV has converged to a minimum. This leads to the conclusion that the parameters affecting the optimal free parameters definitely include the ephemerides of the transfer bodies, the time of flight, and the number of revolutions. These parameters and the free coefficients should thus be used as inputs for the ML algorithm, while ΔV will be used as an output.

It was selected to encode the departure and arrival ephemerides as departure and arrival dates to limit the number of inputs and thus decrease the required complexity of the model. The departure date is used as an input and the arrival date can be obtained by the algorithm by adding the departure date and the time of flight. TudatPy's hodographic functions also use these inputs, enabling rapid switching between approaches with and without ML. The ephemerides of the bodies are extracted using SPICE kernels*. The range of the free parameters is taken to be between -10000 and 10000, as the majority of optimal trajectories are found to lie in this range.[9] The number of revolutions that are considered are 0, 1, 2, and 3, the departure date range is between 8775 and 10275 MJD2000, and the ToF range is between 500 and 2000 days for all transfers. These ranges coincide with other work to make verification of our model easier.[13] The initial training data set is created by distributing the input data uniformly and calculating the corresponding ΔV. Generating a single trajectory for the training data is rapid, as the free parameters do not have to be optimized. Generation of 1000 training samples takes around 11 seconds on average†.

*https://naif.jpl.nasa.gov/naif/spiceconcept.html (retrieved in Sept. 2023)

†All mentioned computation times are from a current laptop CPU: Intel Core i7-9750H CPU @ 2.60GHz, GPU: NVIDIA Quadro P1000

Generalization

Generating training data and training an ML model is a time-consuming process, making it undesirable to create an ML model for each different transfer. In the ideal case, we want to create a single general model that applies to every type of transfer. To test the possibilities of creating such a model and to study the effect of generalizing, three different models are developed in this study, with three different levels of mission generalization. Model 1 will only be trained on a single outward transfer, with two revolutions around the Sun. Model 2 will also only be trained on a single outward transfer but will consider multiple numbers of revolutions. Model 3 will be trained on multiple inward and outward transfers with multiple possible numbers of revolutions. The goal is to generalize this model for all single rendezvous transfers between two bodies.

While the states of the bodies for a single transfer can be represented with the departure date and time of flight, this is not true anymore for a general model. It was selected to still encode the states of the bodies with departure date and time of flight for the general model but to generalize these inputs using the synodic period between the two considered bodies. The departure date is taken as the fraction of the synodic period that has passed since the last opposition, and the time of flight is divided by the synodic period. The training data considers only one synodic period of every transfer. A consequence is that the model loses accuracy as it assumes every synodic period between two bodies is the same, while there are variations caused by inclination and eccentricity differences. Other inputs that were taken into account for generalization are discussed in feature selection.

Models 1 and 2 are only trained on a single transfer and thus do not use the generalized inputs for departure date and time of flight; Model 3 does. All models are tested with 6-DoF and 9-DoF shaping. The models performed best at 125,000 training samples for model 1, 200,000 samples for model 2, and 300,000 samples for model 3. An overview of the differences between the models is given in Table 1.

Model	Bodies in training data	N	Direction	Inputs
1	Earth, Mars or Earth, Ceres	2	Outward only	Not generalized
2	Earth, Mars or Earth, Ceres	0, 1, 2, 3	Outward only	Not generalized
3	Earth, Mars, Vesta, Ceres, Jupiter	0, 1, 2, 3	Inward & outward	Generalized

Table 1: Differences between the proposed models.

MACHINE LEARNING ALGORITHM

This research selected an artificial Neural Network (NN) as the ML algorithm for several reasons. Firstly, we are dealing with a non-linear problem, rendering a regular least-squares fit obsolete. Neural networks are known for being excellent at solving non-linear problems.[14] Secondly, neural networks are good at complex pattern recognition.[15] This is useful, as the data assessed in this research has underlying patterns based on physical and mathematical phenomena. These patterns are hard to recognize for humans but could be detected by an NN. Finally, NNs have widely available documentation and are relatively easy to implement. They are flexible and can be adapted to a wide range of problems with different shapes and sizes.[8] In this work, the neural network is implemented using Scikit-learn.[16]

A typical neural network consists of an input layer, one or more hidden layers, and an output layer. Every layer consists of neurons, where every neuron in a layer is connected to every neuron in the next layer. The input of a neuron is the weighted sum of the outputs from the previous layer's neurons and an added bias. This input is passed through an activation function in order to introduce non-linearity to the model. For a single layer, this process can be denoted in matrix form as in Equation 4.[17]

$$\mathbf{a}^l = \sigma(\mathbf{w}^l \mathbf{a}^{l-1} + \mathbf{b}^l) \tag{4}$$

Where $\mathbf{a}$ is the activation of the neurons in layer l, σ is the activation function, $\mathbf{w}$ is the weights matrix, and $\mathbf{b}$ is the biases matrix. The neural network needs to be trained to obtain its weights and biases. The difference between the predicted output and the actual outputs of the training set is calculated with a loss function. The derivative of this loss function with respect to each weight and bias can be calculated, after which they can be updated accordingly in order to minimize the loss function.[18]

Performance Metrics

Four performance metrics are monitored to see how the model performs when we change its parameters. These metrics are selected as they capture the performance of the entire model and not just the performance of its separate components. The first one is the loss function of the neural network, which in this research is the Mean Squared Error (MSE). This is a measure of the ability of the NN to converge on the dataset. The MSE is calculated as shown in Equation 5.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (a_i - f_i)^2 \tag{5}$$

Where n is the number of samples, a_i are the actual ΔV values, and f_i are the predicted ΔV values. The second metric is the sorting accuracy. It determines how many training samples the model correctly places in the top 25% of a set of 100 random samples. This percentage is then subtracted from 100% to ensure that all performance metrics are best when they are lowest. This metric is chosen because it measures the neural network's ability to distinguish good sets of coefficients from bad sets of coefficients within an optimization.

Furthermore, it is important for the model to converge to the correct minima such that it can find the best combination of input parameters of the overall design space. This is useful in the preliminary design phase for actually locating the regions of optimality. This third metric is measured by looking at a grid search where the departure date and time of flight are varied, and the optimal ΔV is calculated with the ML model for each of them. The pruning accuracy is then calculated by looking at how many minima in a grid search the model correctly places in the top 25%. This percentage is also subtracted from 100%. The final metric is the optimization time. We want the ML model to be significantly faster computationally than the traditional method and as fast as possible, so it is essential to keep track of this. It is defined as the average wall-clock time the optimization of the free parameters of a single trajectory takes.

Tuning

Using the metrics mentioned above, different neural network parameters were tuned to obtain the best possible model. The first thing that was clear was that the data had to be scaled. The data deals with different types of input data, such as Julian dates, velocity difference, and free parameter

values, which can differ in many orders of magnitude. Scaling is used to ensure these data types are in the same range, which enables the NN to learn faster.[19] A standard scaler*, which scales data by removing the mean and scaling to unit variance, worked best in this research.

It was decided that this research would only consider feed-forward neural networks to restrict the number of possible architectures to a reasonable size. The architecture of the NN was determined by keeping the other parameters of the network to their default settings and performing a grid search. During the search, it was found that shallow networks, so networks with a low amount of hidden layers (1-3), were unable to converge during training. This led to very long training times and poor performance. Thus, only a higher number of hidden layers was considered in this grid search, making the network deeper. Deeper networks generally need more training data,[20] but seeing as training data generation is rapid, this is not deemed a problem. Every network architecture was trained three times with random weight initialization, after which the averages of the performance metrics were taken. The effect of the grid search on the performance metrics for model 1 is shown in Figure 2.

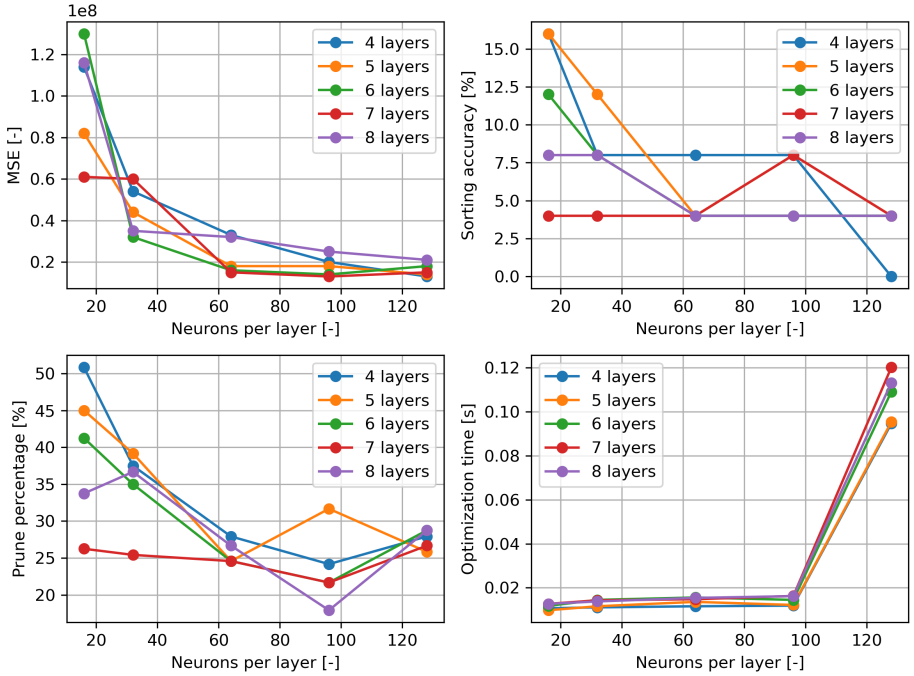


Figure 2: Performance metrics behavior for model 1 in NN architecture grid search for 125,000 training samples. Low scores indicate a better performance for all metrics.

The tuning of the NN mostly behaves as expected: increasing the complexity of the network by increasing the number of layers or neurons per layer generally leads to better performance but higher optimization times.[15] Increasing the number of neurons per layer from 96 to 128 leads to optimization times that are 10 times longer, decreasing the model's effectiveness. Furthermore, the performance of networks with 64 or 96 neurons is similar, but decreasing the number of neurons per layer further leads to worse results, especially considering MSE and prune percentage. We see that

*https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (retrieved in Oct. 2023)

sorting accuracy performance is relatively constant for higher numbers of hidden layers. Finally, for 64 or 96 neurons per layer, a network with 7 or 8 hidden layers performs consistently best or second best in all metrics except optimization time, where the results are very close. The same results were found in the tuning of the networks for the other two models. To keep the approach consistent, the least complex one out of the best-performing architectures for all models was chosen. This means that all models were equipped with a 7x64 neural network.

The Rectified Linear Unit (ReLU) function is used as the activation function in the hidden layers. This function was selected because of its computational efficiency and because it mitigates the vanishing gradient problem.[21] The Adaptive Moment Estimation (Adam) optimizer is used as the solver for weight optimization. A batch size of 200 and a learning rate of 1e-3 are used. The weights are randomly initialized between -1 and 1, and the biases are initialized to zeros. To prevent overfitting of the network, the training data is split into a training set and a test set with an 80/20 split. Other measures to prevent overfitting were not taken, as overfitting did not seem to be an issue in this research.

Feature Selection

The optimal set of free parameters of the hodographic functions changes when the number of revolutions is varied or when different transfer bodies are selected. It is therefore likely the models will benefit from having different input sets depending on their level of generalization. For this reason, a quick investigation of feature selection was performed for each model. Even though the selected features have a significant impact on the model performance, it should be noted that it is possible these features are not able to fully encode a generalized trajectory, and a full review of all possible features could be conducted entirely as its own research. The results of the feature selection performed in this research are shown together in Figure 3 but the results of each separate model should be regarded independently.

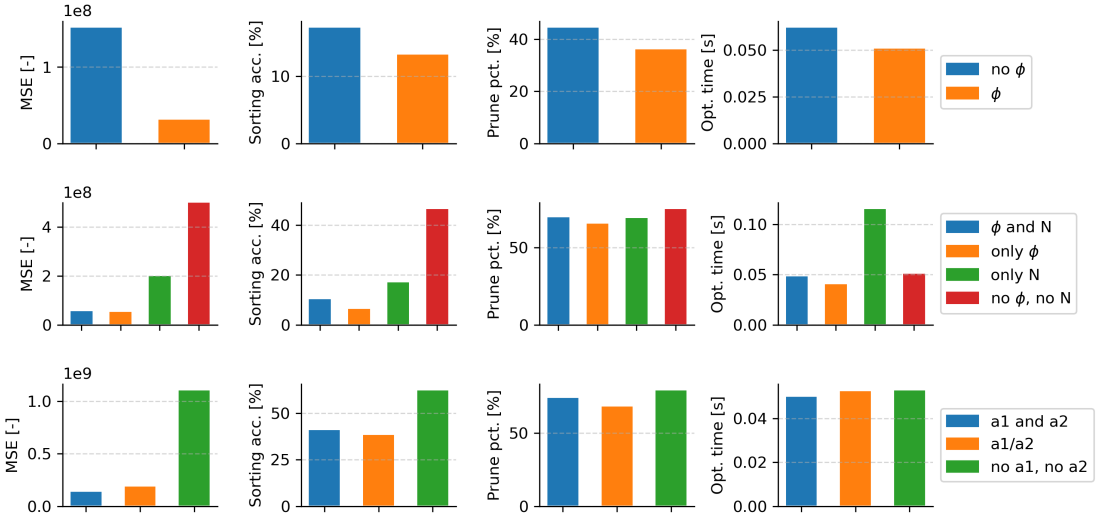


Figure 3: Effect of feature selection on the performance metrics for model 1 (top row), model 2 (middle row), and model 3 (bottom row), average of five runs. Results of each model should be regarded independently. Low scores indicate a better performance for all metrics.

The difference between model 1 and model 2 is the possible number of revolutions considered. It was clear that a feature had to be included that takes this into account. N, the number of revolutions, an integer, and ϕ, the phase angle, a continuous input, were considered for this. Including only the phase angle works best for model 2 across all performance metrics. Retroactively, this feature was also analyzed for model 1, where it was found that the addition of this input also improved this model's performance. Naturally, ϕ was also included as a feature in model 3.

The main difference between the third model and the other two models is that different sets of transfer bodies can be considered using this model, affecting the trajectory's size. It was therefore decided to include a feature that encodes the trajectory size, using the semi-major axes of the departure and arrival bodies, a_1 and a_2. These were tested as separate inputs and as a single input, the ratio of the two axes, and compared to a model without this feature included. It was found that incorporating an input that encodes the size of the trajectory was desirable. It was determined that the ratio should be included as an input for Model 3 because most transfers have a ratio of two semi-major axes that fall within the range in the training data, but they do not necessarily have semi-major axes that are present in the training data. This means that the complete input list for models 1 and 2 is departure date, time of flight, phase angle, and free parameters. The ratio of the semi-major axes is included as input for model 3.

OPTIMIZATION APPROACH

Two optimization methods are used in this research. The first one is the shape optimization, which optimizes the free parameters. This is an integral part of the model that is developed. The second one is a global optimizer that is used to test the application of the model in a global preliminary trajectory optimization.

Shape optimization

The method used to optimize the free coefficients can have a significant effect on the model performance. When Gondelach developed the hodographic shaping method, he tested different optimizers and found that the Nelder-Mead (N-M) simplex method[22] performed best.[9] This finding was later confirmed by Stubbig when he re-assessed the shaping method in the context of his research work.[13] However, by using a neural network to estimate the ΔV instead of the original cost function, the optimization process fundamentally changes, and it is no longer correct to assume that the Nelder-Mead optimization method performs best. For that reason, multiple different optimization methods have been selected and compared for this research. Four local optimizers, which were promising in Stubbig's research,[13] are compared. Differential Evolution (DE) is also tested to see how a global optimizer compares. All optimization algorithms are implemented using PyGMO.[23] The effect of optimizer selection on the performance metrics is shown in Figure 4. The NN MSE is not displayed here because it is independent of the selected optimizer.

The cobyla and sbplx algorithms perform unfavorably, and the DE performs well in terms of sorting accuracy, but the DE is much slower than the other algorithms, indicating that a local optimizer is preferred for this routine. The BOBYQA (Bound Optimization BY Quadratic Approximation) local optimization algorithm[24] performs slightly better than Nelder-Mead in terms of sorting and prune accuracy while being about five times as quick. Therefore it was selected as the optimizer for the ML models in this research.

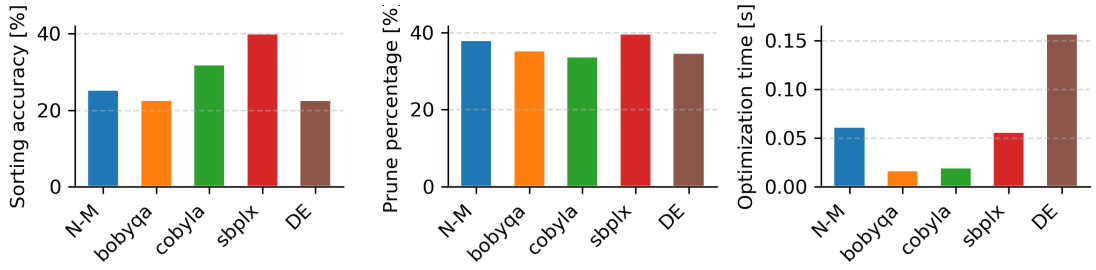


Figure 4: Effect of optimizer selection on performance metrics, average of five runs. Low scores indicate a better performance for all metrics.

The difference in speed is examined more thoroughly in Figure 5, where the average convergence of 10 runs is shown. It is clear that the BOBYQA algorithm needs fewer function evaluations than N-M to converge. Furthermore, the coybla and sbplx algorithms are unable to find the minimum every time, while BOBYQA and N-M are.

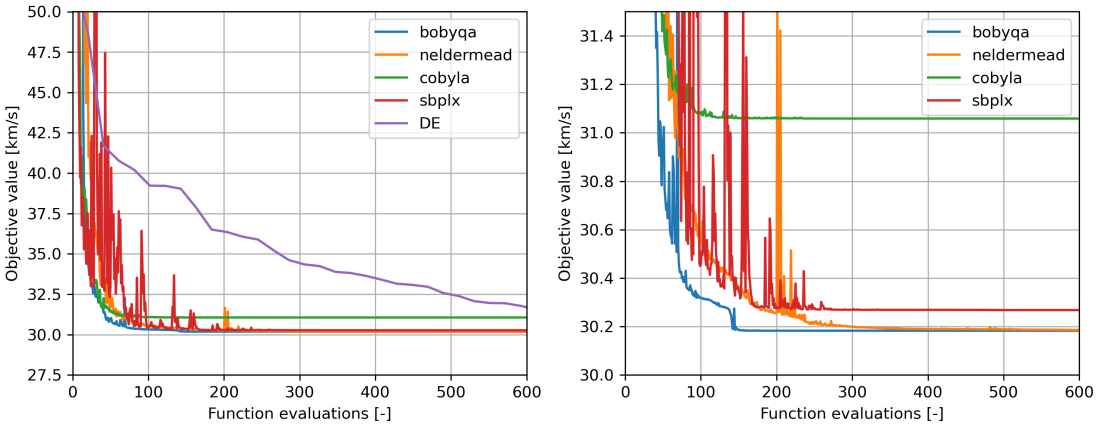


Figure 5: Convergence behavior of different optimizers for ML-implemented hodographic shaping, with zoom-in on the right. Average of 10 runs each.

The optimizer will stop its optimization when one of two stopping criteria is met. A tolerance criterion was set at 1e-3 and a maximum amount of function evaluation criterion was set at 1000. In practice, this amount of function evaluations was never reached, and the stopping was always triggered by the tolerance criterion. All free parameters were set to 0 as an initial guess, which is equal to a hodographic trajectory with zero degrees of freedom, thus constituting a valid trajectory. Optimization of a single transfer takes 0.017 seconds on average. This is up to 100 times as fast as the traditional method with 6 degrees of freedom and close to equally as fast as the traditional method with 0 degrees of freedom.

Global optimizer

In order to test our model in a more realistic preliminary trajectory design context, a global optimizer is used in which candidate solutions for a full mission are evaluated. These candidate solutions consist of full mission trajectories, of which some or all of the legs can be estimated with the ML-incorporated hodographic shaping tool that we developed. It was decided not to tune this global optimizer. This was done because the goal of this test is to see how our developed tool affects a global optimization routine and not necessarily to optimize this routine. Our results show that our ML model can compete with the no-ML model in a global optimization routine even when the global optimizer is not tuned.

The global optimizer used is a Simple Genetic Algorithm (SGA),[25] implemented using PyGMO.[23] This method was also implemented by Stubbig for his evaluation of the DAWN mission[1] and a simplified Global Trajectory Optimization Competition (GTOC) 2 mission.[26] As we will evaluate the same missions, his recommended SGA settings are used in this research.[13] This means that the default PyGMO settings were used for most parameters. The mutation probability was set to 10%, and the mutation scheme was changed to 'uniform'. The population size for both problems was set to 10.

TEST CASES

The presented models are tested in a variety of ways to assess their capabilities and robustness. Firstly, all three models are used in grid searches, where a trajectory and its corresponding ΔV are optimized for sets of departure date and time of flight. This is done for different types of transfers. The results will be compared to the results given by the traditional hodographic shaping methods in terms of computational speed and optimality. Optimality is compared by assessing the lowest ΔV trajectory each method can find. The CPU time is deliberately given in seconds and not in function evaluations, as function evaluations are not equivalent for different models. All results are generated on the same hardware in the same circumstances to give a fair comparison.

Our developed ML models are assessed with 6- and 9-DoF's. The results of the traditional approach will be presented in 6-DoF and 0-DoF. An Earth-Mars and an Earth-Ceres transfer will be evaluated with all three models. Model 1 means that only N = 2 will be analyzed in the grid search. For models 2 and 3, the grid search will be performed for N = [0,1,2,3]. This analysis is done to inspect the robustness of the model and to look at the effect of generalization on model performance. An inward transfer from Mars to Earth is also analyzed with model 3. This is done to see if the performance of the model changes when changing from outward to inward transfers. Finally, a transfer from Vesta-Jupiter is assessed while this transfer is left out of the training data.

As mentioned, the presented model will also be implemented in a global optimization to test its performance in a more realistic preliminary mission design context. This will be done by using a Genetic Algorithm (GA) for the optimization of two different missions: the DAWN mission and the simplified GTOC 2 asteroid mission. A short description of these two missions is given. For each of these missions, a specific 6-DoF ML model is built and used in a global optimization process, as well as the general model 3 and traditional 6-DoF and 0-DoF methods. The performance of the GA will then be compared in terms of computational speed and optimality.

RESULTS

Earth-Mars

The ΔV of the found optimal trajectories and the time it took to complete the grid search of an Earth-Mars transfer for the different discussed models is shown in Figure 6. The lines drawn between the no-ML 6-DoF trajectory and the no-ML 0-DoF trajectory represent a Pareto front for their respective model. Any found optimal trajectory, which is below and to the left of this line, represents a point that improves upon the traditional hodographic shape-based method in either CPU time, optimality, or both. The orange and green no-ML points are in the same location but separated here for visual clarity.

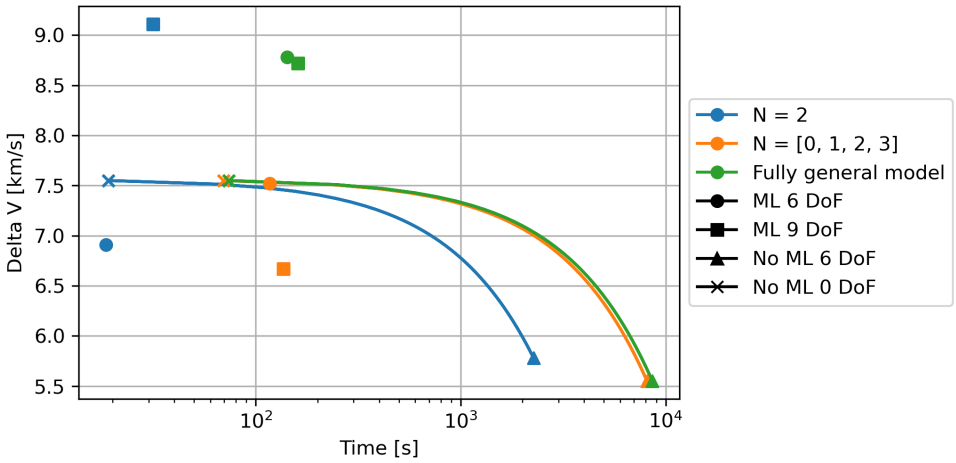


Figure 6: Trajectory optimality and grid search time of different models for an Earth-Mars transfer.

When considering model 1 (N=2), the ML 6-DoF method is as fast as the traditional 0-DoF method and approximately 100 times faster than the traditional 6-DoF method. It is able to find a trajectory that is optimal by more than 0.6 km/s compared to the 0-DoF method. The 9-DoF method is slower, which is expected as it has a larger number of free parameters to optimize. In regards to finding an optimum, the 9-DoF method performs worse than the 6-DoF method, and it is unable to improve the traditional models. For model 2 (N = [0, 1, 2, 3]), the ML 6-DoF method finds a trajectory that is only slightly more optimal than the 0-DoF method. The 9-DoF method does perform well here. Model 3 (fully general) performs significantly worse. Both the 6- and 9-DoF ML methods cannot find a trajectory that improves the traditional methods. The general model is about twice as slow as the traditional 0-DoF method but still about 60 times faster than the traditional 6-DoF method.

From this result, we can conclude that the ML-incorporated methods are very rapid. They are capable of generating results 50-100 times faster than the traditional 6-DoF method and just as fast to twice as slow as the traditional 0-DoF method. These CPU times only include the grid search and thus neglect training data generation time and network training time. This is really only acceptable for a completely generalized model, as then the network has to be trained only once, and these times become genuinely negligible. When generalizing the model, however, the performance of the ML 6-DoF significantly worsens to unacceptable levels. Furthermore, the ML methods become slightly

slower when generalizing, albeit only in the order of seconds for a full grid search. Finally, the ML 9-DoF method has an inconsistent performance over all three models, providing no clear advantage to use it.

Earth - Ceres

The three models and the effect of their generalization are again tested for an Earth-Ceres transfer. The results can be seen in Figure 7. The purple and yellow no-ML points are in the same place but separated here for visual clarity.

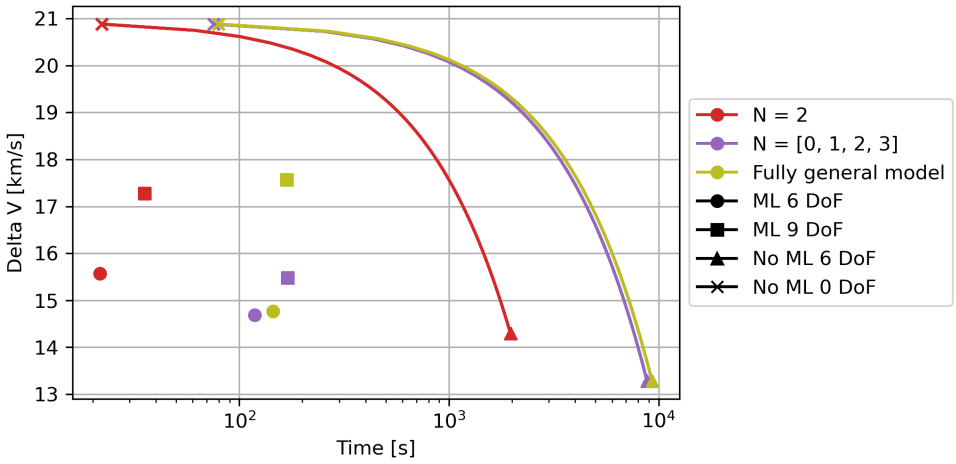


Figure 7: Trajectory optimality and grid search time of different models for an Earth-Ceres transfer.

All three models perform consistently better for an Earth-Ceres transfer than for an Earth-Mars transfer. The 6-DoF method of Model 1 can find a more optimal trajectory by more than 5 km/s compared to the no-ML 0-DoF method in approximately the same time. The 6-DoF methods of models 2 and 3 provide even better trajectories, although they are slightly slower. Here, the models are capable of providing acceptable results even when they get more generalized. The fully general model 3 finds an optimal trajectory that has a lower ΔV by more than 6 km/s in only twice the time compared to the traditional 0-DoF method. Meanwhile, it is approximately 64 times faster than the traditional 6-DoF method, which provides a trajectory that requires a 1.5 km/s lower ΔV. The 9-DoF methods perform more consistently for this transfer as well. All three models provide improvements upon the traditional methods, but they never perform better than the 6-DoF methods.

General Model

To inspect the performance of the general model in different mission types, it is tested on two new transfers and compared to the transfers that are already evaluated. These are an inward transfer, Mars-Earth, and a transfer that has been left out of the training data of the NN specifically for this analysis, Vesta-Jupiter. The performance of model 3 for all transfers it has been assessed for is shown in Figure 8.

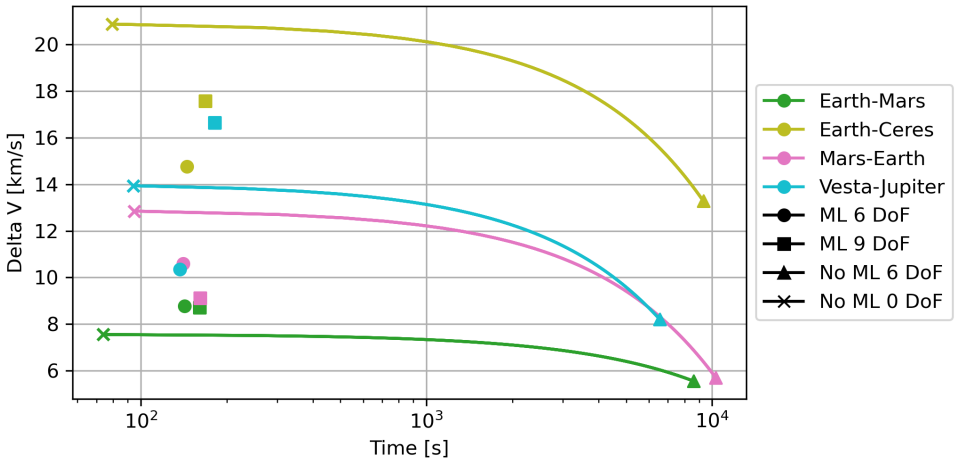


Figure 8: Trajectory optimality and grid search time of different transfers using the fully general model 3.

For the Mars-Earth inward transfer, both the ML methods perform well, the 9-DoF method even finding a more optimal trajectory than the 6-DoF method. This indicates that the developed model is capable of analyzing inward transfers, and the performance does not become worse when assessing inward transfers. For the Vesta-Jupiter transfer, the 9-DoF method does not perform as well, but the 6-DoF method does give an acceptable result. This indicates that the 6-DoF method is capable of analyzing transfers it has not been trained on but which have transfer bodies that are included in the training data.

A couple of conclusions can now be drawn regarding model performance. Firstly, the ML 6-DoF method finds a trajectory that comes close to the no-ML 0-DoF method in terms of computational speed but which is more optimal regarding ΔV in almost all cases. Furthermore, while not reaching the optimality of the traditional 6-DoF methods, the ML models provide a much faster solution, significantly reducing computation time by 50-100 times. In the reported times, the time required for training data generation and NN training is not taken into account, an assumption that is really only fair to make for a full general model. Even though we see that the model loses accuracy and becomes slightly slower when it becomes more generalized, acceptable results for different transfer types, and even for a transfer that was not in the training data, are still produced.

The notable exception to these results is the general model 3 Earth-Mars transfer. The reason for this is suspected to be in the generalization procedure. In support of this, it was found that the general model also performed unsatisfactorily on transfers with a body outside of the training range (for instance Venus-Earth), despite the inputs of these transfers being within the generalized input ranges. This research assumed that generalizing the model using the synodic period, phase angle, and ratio of semi-major axes would be sufficient, but these results suggest there must be other factors influencing the hodographic shape parameters. More extensive research into the features that affect the generalized free parameters of the hodographic shape, and including these as generalized inputs into the model might therefore improve the model significantly. Finally, the 9-DoF methods performed inconsistently across all transfers and all models. To gain more insight into the 9-DoF results, they are discussed in the next section.

9 Degrees of Freedom

The goal of including the 9-DoF method was to see if an ML model could find a more optimal trajectory than the traditional 6-DoF method. This would be done by providing a way to make the trajectory more flexible with more degrees of freedom while not significantly increasing computational cost because it is calculated with an ML model. From the previous results, we can conclude that, despite being much faster, the 9-DoF methods never improve upon the optimal trajectory found by the traditional 6-DoF methods. To gain more insight into these results, Figure 9 shows how the grid search for the 9-DoF general model 3 Earth-Mars transfer compares to the traditional, no-ML, 6-DoF method.

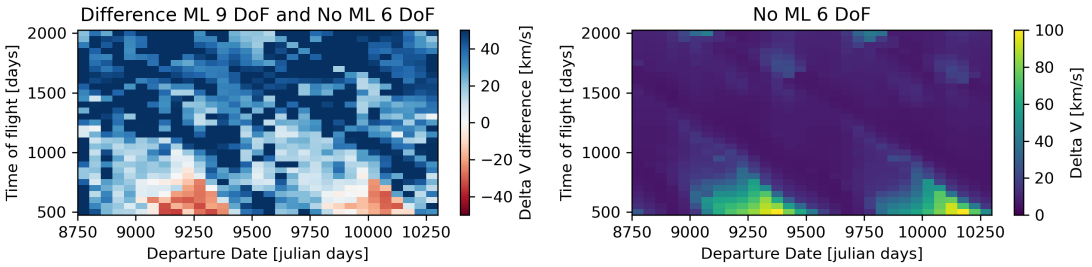


Figure 9: Results of grid search of 9-DoF model 3 compared to traditional 6-DoF method for Earth-Mars transfer. Red color indicates a better performance by the 9-DoF ML model.

This was one of the worst-performing test cases, with the overall optimum found by the 9-DoF model being more than 3 km/s higher than the optimum found by the traditional 6-DoF method. However, it is apparent that there are some regions in the grid search where the 9-DoF model is indeed able to improve upon the traditional method quite significantly by more than 20 km/s. This happens in the regions where the traditional method has the highest ΔV. The same pattern is seen when repeating the analysis on the grid search for the 9-DoF general model 3 Earth-Ceres transfer as shown in Figure 10.

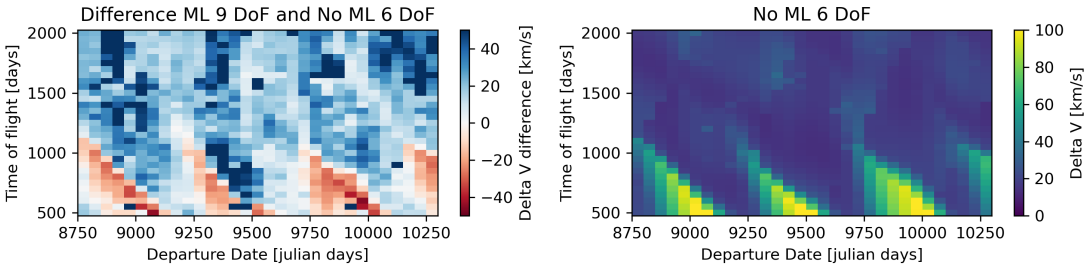


Figure 10: Results of grid search of 9-DoF model 3 compared to traditional 6-DoF method for Earth-Ceres transfer. Red color indicates a better performance by the 9-DoF ML model.

Again, the 9 DoF method is able to significantly improve some trajectories but only in regions of high ΔV. This means that the 9-DoF ML model is indeed capable of improving the 6-DoF traditional method, but in the way it has been implemented now, it is not yet applicable. However, there are clear ways this method can be improved. Firstly, in this research, the model architec-

ture was tuned for the 6-DoF method and not for the 9-DoF method. With the three extra inputs, the model's complexity increases, which might make different model settings favorable. Secondly, a more thorough investigation into shaping function selection is required. The recommended 6-DoF functions for an Earth-Mars transfer were used here for all transfers, and the additional 9-DoF functions were arbitrarily chosen, meaning there probably are more suitable functions for all other transfers. This investigation will not only improve the proposed ML model but also the hodographic shaping method in general. Upon implementing these improvements, it is expected that the 9-DoF ML model could be able to find better optimal trajectories than the 6-DoF model in only a fraction of the time.

DAWN mission

The performance of the ML model in a GA is now discussed. The DAWN test case is a slight simplification of NASA's DAWN mission, which set out to explore Vesta and Ceres in 2007.[1] DAWN was a low-thrust propulsion spacecraft that first reached Mars after an orbit injection at Earth. At Mars, it performed an un-powered fly-by towards Vesta, where it performed a rendezvous. After a scientific mission phase where the spacecraft orbited Vesta, a low-thrust rendezvous transfer towards Ceres was performed, where DAWN again executed a scientific phase in orbit until its end of life. The total mission ΔV came to about 11 km/s. The trajectory of the DAWN mission is shown in Figure 11.

The genetic algorithm optimizes this transfer with respect to ΔV. The ML model is implemented in this mission only in the third transfer from Vesta to Ceres. The traditional 6-DoF method is used for the other two transfers, as our model can not yet deal with orbit injections or fly-bys. A specific model, trained only on the Vesta-Ceres transfer, and the general model 3 are both implemented. The performance of the genetic algorithm is compared to its performance when the traditional 6-DoF and 0-DoF methods are used for this transfer. The number of revolutions of every transfer stays fixed at 0, 1, and 0 respectively. The parameters that are optimized by the GA, their Lower Bounds (LB), and Upper Bounds (UB) are shown in Table 2.

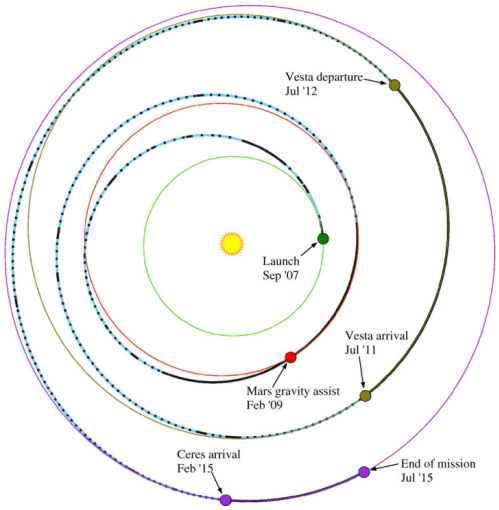


Figure 11: DAWN mission trajectory.[27]

Parameter	**LB**	**UB**
d_0 [MJD2000]	2500	3500
ToF 1 [days]	300	700
ToF 2 [days]	700	1100
Stay 1 [days]	200	600
ToF 3 [days]	700	1100
Injection V_∞ [m/s]	1.5×10^3	5×10^3
Mars arrival V_r [m/s]	-5×10^3	5×10^3
Mars arrival V_θ [m/s]	1×10^4	5×10^4
Mars arrival V_z [m/s]	-5×10^3	5×10^3
Flyby altitude [m]	1.5×10^5	1×10^7
Flyby plane angle [rad]	0	2π

Table 2: Optimization bounds for DAWN problem.[13]

The genetic algorithm was run for 30 generations for each of the models. The average of 5 different optimization runs was taken to rule out any unexpected irregularities. For these optimization runs, the best individuals over generations and over time are shown in Figure 12.

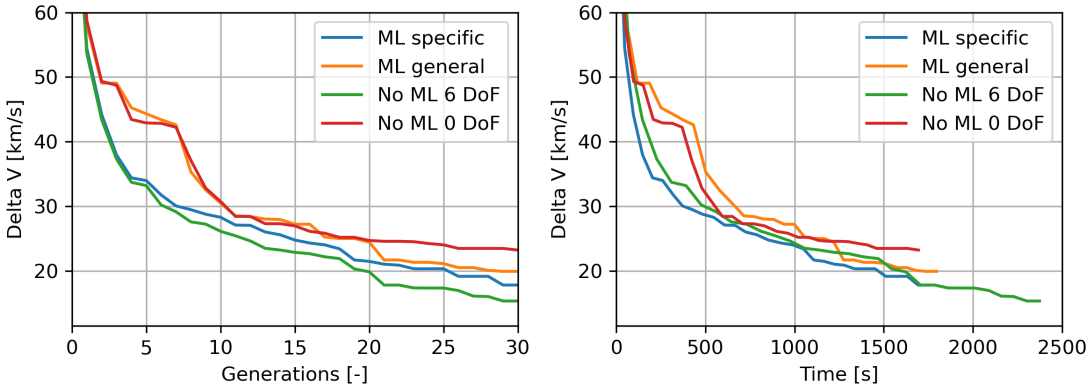


Figure 12: Genetic algorithm performance on DAWN problem with different models used for the Vesta-Ceres transfer, average of 5 runs each.

Considering ΔV over generations, the following can be concluded: the traditional 6-DoF method performs best, converging to the lowest ΔV. Then, the ML model trained specifically for this transfer and the general ML model follow. With the traditional 0-DoF method performing worst, the different methods compare as expected from the previous results in terms of optimality. From this analysis only, it seems not beneficial to implement the ML models. However, these models are much faster than the traditional 6-DoF method.

Considering ΔV over time, the GA with the specific ML model has a better candidate solution than any of the other methods for any given time before approximately 1700 seconds. If the optimization were stopped after approximately 1400 seconds for all models, both ML models would have a better candidate solution than the traditional method. This shows that in a time-driven optimization, it can be beneficial to incorporate these models. However, when the optimization routine can be completed for all models, the traditional method will have a better candidate solution. Analyzing these results shows that it is more beneficial to implement an ML model when there is a larger time gain to be made. In the DAWN problem, only one out of three legs is optimized with this model. To demonstrate the time difference the ML model can make, the simplified GTOC 2 mission is assessed, where more legs can be optimized using ML.

Simplified GTOC 2 mission

The goal of GTOC 2 was to find a trajectory that visits one asteroid out of four different groups in the most efficient manner.[26] After orbit injection at Earth, there is a scientific orbit stay at every asteroid, and every subsequent transfer is a rendezvous transfer. The minimum ΔV found in the competition was about 20 km/s. The different asteroid groups are shown in Figure 13 to show the geometry of the problem. To simplify this problem, we assume a predetermined asteroid sequence* and optimize this trajectory for low ΔV with the GA. The ML model is used to optimize three transfers: every transfer except for the first one because of its orbit injection. The number of

*Pykep[28] indices: 815, 300, 110, 47

revolutions is fixed for all transfers at 0. The general model used in this analysis is model 3. The specific model for this test case is also a generalized model, but it is trained only on all possible trajectories between Earth and the four asteroids. The parameters and their bounds that are used in the optimization are shown in Table 3.

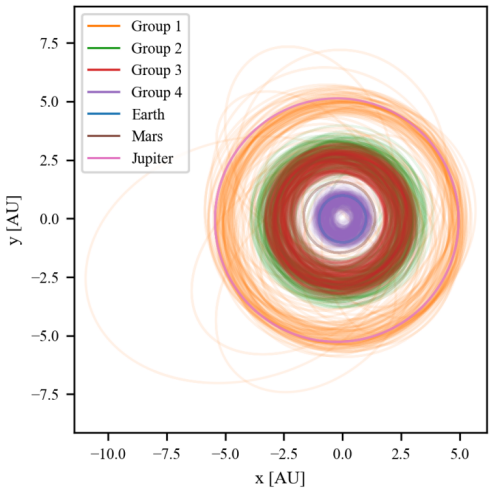


Figure 13: GTOC 2 asteroid groups.[13]

Parameter	**LB**	**UB**
d_0 [MJD2000]	5479	9129
ToF 1 [days]	100	1700
Stay 1 [days]	90	110
ToF 2 [days]	200	1900
Stay 2 [days]	90	110
ToF 3 [days]	400	1900
Stay 3 [days]	90	110
ToF 4 [days]	600	2000
Injection V_∞ [m/s]	0	3.5×10^3
Injection angle ϕ_1 [rad]	0	π
Injection angle θ_1 [rad]	0	2π

Table 3: Optimization bounds for GTOC 2 problem.[13]

Due to the time difference the ML models can make when they are implemented in multiple legs of a mission, the GA can complete 50 generations in the time it took the GA to complete 30 generations for the DAWN mission. Therefore it was decided to run the genetic algorithm for 50 generations for each of the models. Again the average of 5 different optimization runs was taken to rule out any unexpected irregularities. The results are shown in Figure 14.

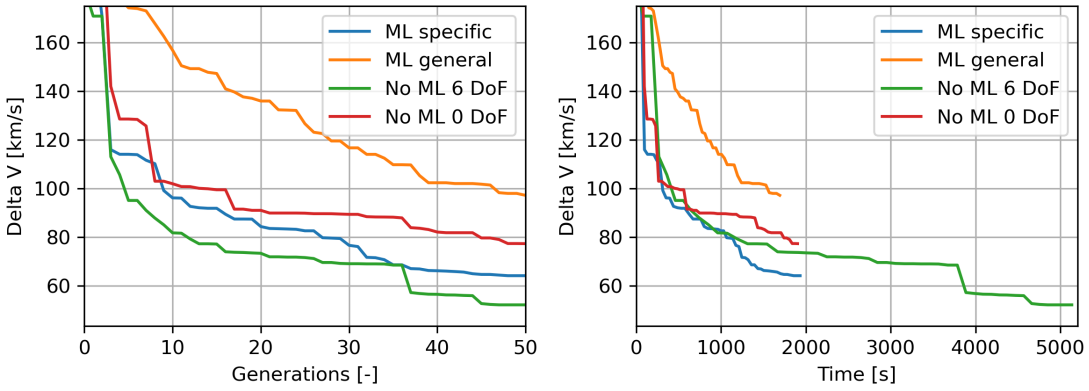


Figure 14: Genetic algorithm performance on GTOC2 problem with different models used for all transfers between two asteroids, average of 5 runs each.

The general ML model performs poorly in this optimization process. This is not a surprise as none of the asteroid bodies are included in the training data, reaffirming our conclusion that the

generalization process might need to be completed. The specific ML model does perform well and converges rapidly time-wise. After 50 generations and 1900 seconds, the best solution found by the GA with the general ML model is approximately 65 km/s. It takes the GA with the no-ML 6-DoF method more than twice as long to find a better candidate solution. It can be concluded that ML-incorporated hodographic shaping could significantly speed up a preliminary mission design process when used in a global optimization tool. However, improvements must be made before a single general model can be used for different missions. Two possible improvements tie back into the previous two main recommendations. Firstly, a single general model can only be used if the generalization process can be made more consistent and reliable across all kinds of different transfers. Furthermore, if the 9-DoF model can be improved and used in the GA, it could converge to ΔV's lower than the traditional 6-DoF model can now provide.

CONCLUSION

In this paper, a model has been developed that implements machine learning into a shaping method for low-thrust trajectory optimization. A neural network is used to optimize the free parameters of the hodographic shaping method, with the goal of improving this method in terms of either computational speed, optimality, or both. It is found that for single transfers, an ML model generally finds more optimal trajectories than the no-ML 0-DoF hodographic method while only slightly increasing the required computational effort. It is also a factor of 50-100 times faster than the no-ML 6-DoF method but is not able to equal this method's optimality. It was found that the realization of a completely general model is possible as it was seen that this model could work for outward and inward transfers and for trajectories where the transfer itself is not in the training data, but its transfer bodies are. There were, however, unacceptable results for certain test cases suggesting that there are more factors that should be considered for the generalization procedure.

Furthermore, a 9-DoF ML model was developed which performed inconsistently but showed promise as it could improve the no-ML model in certain regions. Finally, It was found that implementation of the ML model can significantly speed up the convergence of a genetic algorithm in the preliminary optimization of complex missions. It can be concluded that by providing a combination of optimality and computational speed that was previously unfeasible, the implementation of machine learning can improve a shape-based method. When the generalization procedure and more degrees of freedom model are further developed, this could be a valuable method for the preliminary trajectory design of space missions.

REFERENCES

[1] C. Russell and C. Raymond, "The Dawn mission to Vesta and Ceres," *The dawn mission to minor planets 4 vesta and 1 ceres*, 2012, pp. 3–23.

[2] J. Benkhoff, J. Van Casteren, H. Hayakawa, M. Fujimoto, H. Laakso, M. Novara, P. Ferri, H. R. Middleton, and R. Ziethe, "BepiColombo—Comprehensive exploration of Mercury: Mission overview and science goals," *Planetary and Space Science*, Vol. 58, No. 1-2, 2010, pp. 2–20.

[3] S. Williams and V. Coverstone-Caroll, "Benefits of Solar Electric Propulsion for the Next Generation of Planetary Exploration Missions," *The Journal of the Astronautical Sciences*, Vol. 45, No. 2, 1997, pp. 143–159.

[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] M. Vijayakumar and O. Abdelkhalik, "Shape-Based Approach for Low-Thrust Earth–Moon Trajectories Initial Design," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 1, 2022, pp. 103–120.

[6] B. Dachwald, "Optimization of very-low-thrust trajectories using evolutionary neurocontrol," *Acta Astronautica*, Vol. 57, No. 2-8, 2005, pp. 175–185.

[7] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodynamics*, Vol. 3, 2019, pp. 287–299.

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

[9] 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.

[10] B. J. Wall and B. A. Conway, "Shape-based approach to low-thrust rendezvous trajectory design," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 95–101.

[11] P. De Pascale and M. Vasile, "Preliminary design of low-thrust multiple gravity-assist trajectories," *Journal of Spacecraft and Rockets*, Vol. 43, No. 5, 2006, pp. 1065–1076.

[12] D. Dominic, F. Marie, G. Geoffrey, A. Miguel, C. Kevin, C. Sean, E. Joao, C. F. Lombrana, G. Jérémie, H. Jonas, *et al.*, "The open-source astrodynamics Tudatpy software. Overview for planetary mission design and science analysis," *Europlanet Science Congress 2022*, 2022.

[13] L. Stubbig and K. Cowan, "Improving the Evolutionary Optimization of Interplanetary Low-Thrust Trajectories Using a Neural Network Surrogate Model," *Advances in the Astronautical Sciences*, Vol. 175, 2021.

[14] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, Vol. 521, No. 7553, 2015, pp. 436–444.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, Vol. 12, 2011, pp. 2825–2830.

[17] M. A. Nielsen, *Neural networks and deep learning*, Vol. 25. Determination press San Francisco, CA, USA, 2015.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, Vol. 323, No. 6088, 1986, pp. 533–536.

[19] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Neural Networks: Tricks of the Trade: Second Edition*, pp. 437–478, Springer, 2012.

[20] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital signal processing*, Vol. 73, 2018, pp. 1–15.

[21] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, Vol. 6, No. 12, 2017, pp. 310–316.

[22] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions," *SIAM Journal on optimization*, Vol. 9, No. 1, 1998, pp. 112–147.

[23] F. Biscani, D. Izzo, and C. H. Yam, "A global optimisation toolbox for massively parallel engineering optimisation," *arXiv preprint arXiv:1004.3824*, 2010.

[24] M. J. Powell *et al.*, "The BOBYQA algorithm for bound constrained optimization without derivatives," *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, Vol. 26, 2009.

[25] S. Mirjalili, "Genetic algorithm," *Evolutionary algorithms and neural networks*, pp. 43–55, Springer, 2019.

[26] A. E. Petropoulos, "Problem description for the 2nd global trajectory optimisation competition," *Online at http://www.esa.int/gsp/ACT/doc/MAD/ACTPRE-MAD-GTOC2-problem_description. pdf*, 2006.

[27] 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.

[28] D. Izzo, "Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization)," *Proceedings of the Fifth International Conference on Astrodynamics Tools and Techniques, ICATT*, sn, 2012.

3

Conclusions & Recommendations

A model has been developed that implements machine learning into a shaping method for low-thrust trajectory optimization. This chapter contains the conclusions of this research and recommendations for future work.

3.1. Conclusions

The obtained results can now be used to answer the research questions.

- *What does it mean for a shape-based method to improve?*
  A shape-based method is used as a preliminary design tool. Its (semi-)analytical nature is used to rapidly search the initial design space for optimal regions. In this research, we assume the hodographic shaping method is capable of providing close-to-optimal solutions to be used as an initial guess by numerical methods, where a lower ΔV trajectory means a more optimal solution. This means the shaping method can be improved in two ways. Firstly, the shaping method can be made less computationally expensive because the entire optimization process will then be quicker. Secondly, if the shaping method can provide lower ΔV trajectories, it can supply better initial guesses to numerical methods.

- *Is the use of the shaping method less computationally expensive when the ML model is incorporated?*
  In this work, machine learning is incorporated into the optimization of free parameters of the shaping method. It is found that the shaping method is considerably faster with this incorporation. The addition of degrees of freedom of the trajectory shape can be performed at a low computational cost. The ML 6-DoF method is 50-100 times faster than the no-ML 6-DoF method and has a computational speed close to the no-ML 0-DoF method. The ML 9-DoF method is slightly slower than the ML 6-DoF method but is still two orders of magnitude faster than the no-ML 6 DoF method.

- *Does the shaping method lead to more optimal results when the ML model is incorporated?*
  The ML 6-DoF method can consistently find more optimal trajectories than those found by the no-ML 0-DoF method, with a maximum found improvement in ΔV of more than 6 km/s for an Earth-Ceres trajectory. However, the no-ML 6-DoF method is able to find the most optimal trajectory in every test case. The 9-DoF ML model was built to see if an ML model could find a more optimal trajectory than the no-ML 6-DoF method. This model performed inconsistently and was often unable to find a trajectory more optimal than the 6-DoF models. However, the 9-DoF ML model showed the capability of improving the no-ML 6-DoF model in the areas of lowest optimality for the no-ML model.

- *Can a single ML-incorporated shaping method be used for different transfer types?*
  Several ML models have been built with varying levels of generalization, with the goal of finding out the effect of generalization on the model's performance. It was found that generalizing the

model decreases its computational efficiency slightly and makes it produce less optimal trajectories. Nevertheless, the results were still deemed acceptable in terms of computational speed and optimality for most test cases within the training range. It was seen that this model could work for outward and inward transfers and for trajectories where the transfer itself is not in the training data but its transfer bodies are. However, there were unacceptable results for an Earth-Mars trajectory and transfers with bodies not included in the training data but which do fall into the generalized training data ranges. These results suggest there must be other factors influencing the hodographic shape parameters than the ones taken into account in the generalization process.

- *How does the ML-incorporated shaping method perform when implemented in a more complex preliminary mission design?*
  The performance of a genetic algorithm optimizing a complex space mission with ML-incorporated shaping and with traditional hodographic shaping was compared. It was found that implementing the ML model could significantly speed up a preliminary mission design process when used in global optimization. Generally, the traditional method will find a better candidate solution per generation, but the ML models can perform many more generations in the same time because they are much quicker. However, generalization improvements must be made before a single model can be used for the optimization of different types of missions.

By combining the answers to these questions, the main research question can be answered: by providing a combination of optimality and computational speed that was previously unfeasible, the implementation of machine learning can improve a shape-based method. It is shown that this could be a valuable method for the preliminary trajectory design of space missions, however further improvements are still required.

3.2. Recommendations

Several recommendations for future work can be made based on these conclusions. The three listed below are the most promising options. Implementing these recommendations will significantly improve the model and will make it usable for implementation in preliminary mission design.

- *Extensive feature engineering for generalization procedure.*
  It was found that the general model performed inconsistently, and therefore, extensive research on the features that affect the generalized free parameters of the hodographic shape is recommended. This study has shown that the inclusion of features regarding the relative position and distance of the transfer bodies, as well as the phase angle of the transfer, visibly improves the model. Still, there will be other features that were not considered that can have a considerable effect. A good starting point would be to consider more features describing the characteristics of the two transfer bodies' orbits in relation to each other. Also, an investigation of features using transfer bodies outside of the ones that were considered in this study would be useful. Including the found features as generalized inputs into the model will improve the model significantly.

- *Investigate 9 degrees of freedom tuning and shaping functions.*
  The 9-DoF ML models showed that they were capable of finding better trajectories than the traditional 6-DoF shaping method. However, these trajectories have not yet been found in valuable regions. By improving this model, the ML models could find more optimal trajectories than the traditional method in just a fraction of the time. Two things could be done to improve the method significantly. Firstly, the model should be properly tuned for the 9-DoF model. This research assumed the same model architecture would be sufficient for both 6-DoF and 9-DoF models. However, the model's complexity increases, as it now has three extra inputs, which might make different model settings favorable. Secondly, a more thorough investigation into shaping function selection is required. The recommended 6-DoF functions for an Earth-Mars transfer were used here for all transfers, and the additional 9-DoF functions were arbitrarily chosen. Ideally, the hodographic shaping should automatically recognize which functions are best to use from the assessed transfer bodies. Not only the hodographic ML model will improve from the selection of better functions, but also the hodographic shaping method in general will improve.

- *Incorporation of non-rendezvous trajectories.*
  The models built in this research are only capable of analyzing rendez-vous trajectories, but preferably, they should also be able to deal with orbit injections and fly-bys. This would enable the ML models to be employed in every leg of complex missions instead of just the rendez-vous legs. This could be seen as an even further generalization of the model. It is suspected that including orbit injections and fly-bys would require many more inputs, as parameters like injection angles, injection magnitude, fly-by radius, and fly-by plane angle, to name a few, all affect the shaping parameters of these legs. It is, therefore, unclear if the model architecture proposed in this research would still be the best. This recommendation would require the most work, but it will also vastly improve the model if properly implemented. When every leg of a complex mission can be analyzed with the ML model, its optimization procedure will be sped up immensely.

A

Dynamical Model

The paper presented an approach that could be used for low-thrust trajectory optimization. This appendix presents the dynamical model that is used to describe these trajectories. The reference frame and coordinate system are defined and the hodographic shaping method used to shape the trajectories is explained.

A.1. Reference Frame & Coordinate System

This work uses an inertial reference frame centered at the Sun to express positions and velocities within the solar system: the ECLIPJ2000 frame [13]. The x-axis of this reference frame points to the interception of the equatorial and ecliptic planes, the vernal equinox, at the J2000 epoch. The z-axis points perpendicular to the ecliptic, at the J2000 epoch. The y-axis completes the right-hand frame. This means that the x-y plane of this frame coincides with the ecliptic plane.

The hodographic shaping method makes use of a cylindrical coordinate system. Cylindrical coordinate systems use radial distance r, polar angle θ, and height z to describe something in 3 dimensions. The cylindrical coordinates can be calculated from Cartesian coordinates x,y, and z with the following equations.

$$r = \sqrt{x^2 + y^2} \tag{A.1}$$

$$\theta = arctan(\frac{y}{x}) \tag{A.2}$$

$$z = z \tag{A.3}$$

A low-thrust trajectory can have multiple revolutions around the Sun. The polar angle enables this coordinate system to show smaller variations during these revolutions, making it suitable for shape-based methods. The cylindrical coordinate system is shown in Figure A.1.

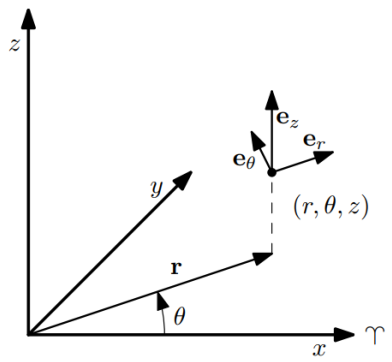


Figure A.1: Cylindrical coordinate system (2D depiction of a 3D space).

A.2. Hodographic Shaping

Shape-based methods are used in the preliminary mission design phase to explore large design spaces for low-thrust trajectories. In these methods, the low-thrust trajectory is assumed to be of a specific analytical shape, from which the required thrust to follow that shape is determined. Most of these methods shape the position of the spacecraft along its trajectory. Hodographic shaping, as created by Gondelach, takes a different approach by shaping in the velocity domain instead of in the position domain [14]. This is done by shaping the trajectory's velocity hodograph. When the hodographs of the departure and arrival trajectories are drawn, they can be connected by a line that represents the transfer trajectory. The velocity boundary conditions are met as this connection intercepts both the departure and arrival orbit in the hodograph. An example of this is given in Figure A.2.

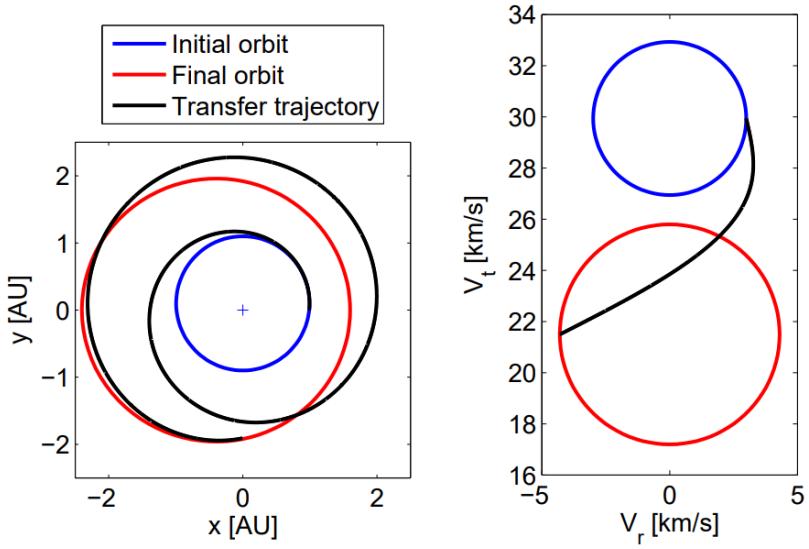


Figure A.2: Transfer between two orbits in the position domain (left) and velocity domain (right) [14]

This shape method uses the 3-dimensional, cylindrical coordinate system, meaning the shape must satisfy the equations of motion shown in the following equations.

$$\ddot{r} - r\dot{\theta}^2 + \frac{\mu}{s^3} r = f_r \tag{A.4}$$

$$\ddot{\theta} + 2\dot{r}\dot{\theta} = f_\theta \tag{A.5}$$

$$\ddot{z} + \frac{\mu}{s^3} z = f_z \tag{A.6}$$

Where $\dot{\square}$ and $\ddot{\square}$ denote the first and second derivative with respect to time, respectively. μ is the gravitational parameter of the central body, which for interplanetary transfers is the Sun, $s = \sqrt{r^2 + z^2}$, and f is the thrust-acceleration vector which is defined as the thrust vector divided by the instantaneous mass of the spacecraft. There are two approaches for hodographic shaping: a time-driven one and a polar angle-driven one. In the time-driven approach, the radial, transverse, and axial velocities are shaped with respect to time using mathematical functions. The position and acceleration of the trajectory can be retrieved as a function of time through the integration and differentiation of these functions. The ΔV can be obtained by solving the equations of motion and integrating the thrust acceleration over time. In the polar angle-driven approach, the radial and axial velocity and the time evolution are shaped as a function of the polar angle. The ΔV can be determined in a similar way as in the time-driven method.

The velocity functions are determined by summing multiple functions. Simple 'base' functions are used to keep the approach analytically feasible. These base functions can include constant, polynomial, trigonometric, and exponential terms, as well as combinations of them. The velocity function $V(t)$ (or $V(\theta)$) is then constructed for each direction as in Equation A.7.

$$V(t) = \sum_{i=1}^{n} c_i v_i(t) \tag{A.7}$$

With $v_i(t)$ being the base functions and c_i their coefficients. n must be greater than or equal to 3, as the functions have three boundary conditions to fulfill. For each direction, the velocity function has an initial velocity (V_0), a final velocity (V_f), and a change in position ($P_f - P_0$) boundary condition. A second boundary condition on position is not required as the initial position can be obtained with the constant when integrating the velocity function. When n is larger than 3, free parameters are introduced which need to be selected. From there the fixed parameters can be calculated using the boundary conditions as in Equation A.8, which needs to be calculated for all three velocity components separately.

$$\begin{aligned} \begin{bmatrix} v_1(0) & v_2(0) & v_3(0) \\ v_1(t_f) & v_2(t_f) & v_3(t_f) \\ \tilde{v}_1(t_f) - \tilde{v}_1(0) & \tilde{v}_2(t_f) - \tilde{v}_2(0) & \tilde{v}_3(t_f) - \tilde{v}_3(0) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \\ = \begin{bmatrix} V_0 - \sum_{i=4}^{n} c_i v_i(0) \\ V_f - \sum_{i=4}^{n} c_i v_i(t_f) \\ P_f - P_0 - \sum_{i=4}^{n} c_i \left[\tilde{v}_i(t_f) - \tilde{v}_i(0)\right] \end{bmatrix} \end{aligned} \tag{A.8}$$

An optimization procedure must be applied to select the parameters leading to an optimal shape. The optimality of the shape is determined by the amount of ΔV. Gondelach proposed to use the Nelder-Mead method for the optimization, but it was found that the BOBYQA optimizer performs better for machine-learning incorporated hodographic shaping in this study. A benefit of adding free parameters is that it leads to more flexibility in the shape, which leads to more optimal solutions. On the other hand, the optimization process will take more computational effort, thereby counteracting the analytic efficiency of the shaping method. Gondelach finds that a trajectory with two free parameters per direction, so 6 in total, is generally near-optimal but takes about 2 seconds to generate. In contrast, a trajectory without free parameters is not as optimal but only takes in the order of milliseconds to generate [14].

This research uses the time-driven version of hodographic shaping and assesses 0-DoF, 6-DoF, and 9-DoF trajectories. The radial and tangential velocities $V_{(r/\theta)}$ are given by Equation A.9 and axial velocity V_z is given by Equation A.10.

$$V_{(r/\theta)} = c_1 + c_2 t + c_3 t^2 + c_4 t \sin(0.5t\pi) + c_5 t \cos(0.5t\pi) + c_6 e^t \sin(0.5t\pi) \tag{A.9}$$

$$\begin{aligned} V_z = c_1 \cos(2\pi t(N+0.5)) + c_2 t^3 \cos(2\pi t(N+0.5)) + c_3 t^3 \sin(2\pi t(N+0.5)) \\ + c_4 t^4 \cos(2\pi t(N+0.5)) + c_5 t^4 \sin(2\pi t(N+0.5)) + c_6 e^t \sin(0.5t\pi) \end{aligned} \tag{A.10}$$

Where N is the number of revolutions of the trajectory. In the trigonometric functions the time t is scaled by a factor 2π over the Time of Flight (ToF). c_4,c_5 and c_6 are zero for a 0-DoF trajectory, c_6 is zero for a 6-DoF trajectory. The first five base functions per direction were selected because they were recommended by Gondelach. The sixth base function was arbitrarily chosen. By hard forcing different combinations, Gondelach showed that most different transfers require a unique combination of functions to obtain their most optimal trajectory. As of right now, there is no method available for the selection of these functions. A proper analysis of the choice of shaping functions should be able to significantly improve the performance of hodographic shaping.

B

Optimization Methods

In this chapter, the three main optimization methods that were used in this research are discussed. These optimizers are crucial for different parts of this work, so a basic understanding of them is desirable. Firstly, the method this research tries to improve, hodographic shaping, uses the Nelder-Mead method. The model built in this research itself uses the BOBYQA algorithm. Furthermore, the model is applied in a genetic algorithm, so an understanding of this is also required.

B.1. Nelder-Mead

The Nelder-Mead method is a single-objective, derivative-free, unconstrained optimization method [15]. It tries to minimize an objective function with n input variables, thus an objective function in an n-dimensional design space. This is done by constructing a simplex, a geometric shape, with $n + 1$ points and calculating the objective function at each point. Then, the point with the highest objective value will be mirrored through the simplex or the simplex will be expanded to form a new simplex. This means that the simplex will gradually move towards the lowest point. When the simplex shape changes sufficiently little, the optimization method has converged.

Gondelach found in his research that the Nelder-Mead simplex method performed best for the optimization of free parameters of a hodographic shape [14]. It was selected because it turned out to be very robust and only needed a small number of function evaluations when compared to differential evolution. However, he did not try out any other optimizers, and it has been stated that the Nelder-Mead method is not very efficient and a better algorithm could probably be found for all applications [16]. Furthermore, a different cost function is used in the parameter optimization when an ML model is involved. For those reasons, it was decided to research if other optimizers would perform better in our approach.

B.2. BOBYQA

The optimization method that turned out to perform better than Nelder-Mead in this research is the BOBYQA (Bound Optimization BY Quadratic Approximation) optimizer [17]. This is also a single-objective, derivative-free, unconstrained optimization method. The optimizer starts off with an initial guess around which it constructs a trust region. By evaluating points in the trust region, the optimizer constructs a quadratic approximation of the objective function in this region. By comparing the values found by the quadratic approximation and the actual objective values, the algorithm can then update the trust region to make the model more accurate. After each iteration, the trust region and the quadratic approximation are updated until a stopping criterion is reached and the optimum has thus been found.

It is not entirely sure why BOBYQA performed much better than Nelder-Mead in the application of this research. Possible reasons could be that Nelder-Mead can converge slower in problems of higher dimensions because it uses points of a simplex instead of quadratic approximation [18] or that BOBYQA is generally considered to be an algorithm with good convergence properties [19], while Nelder-Mead is an older algorithm. What is sure, is that BOBYQA consistently converged to the same optimums as

Nelder-Mead within 4-5 times fewer function evaluations, making it the preferred optimizer for the free parameters in this research. In this research it is implemented using Pygmo [20] and NLopt [1].

B.3. Genetic Algorithms

Genetic algorithms (GAs) are a type of evolutionary algorithm. These algorithms are, as the name suggests, inspired by biological phenomena. This inspiration can come from many biological events, ranging from evolution and mutation to animal swarm and foraging behavior [21]. GAs use the concept of a changing population with individuals that are candidate solutions. The population changes in an evolutionary manner to find the optimal solution to a problem. GAs are capable of handling optimization problems in complex and rugged fitness landscapes and are therefore useful for optimization of low-thrust trajectories [11]. They have been used consistently and successfully in comparable recent research [7].

In a genetic algorithm, the population is randomly initialized. This population is a collection of candidate solutions, for each of which the fitness is calculated. The fitness is a function that determines the quality of the solution. For low-thrust trajectory optimization, the individual often is a trajectory, and its fitness is a useful quantity like ΔV, as in this research. The population of a GA will then gradually change to better solutions, as the individuals are subject to selection, crossovers, and mutation to generate the 'offspring', individuals that make up the next generation of the population [22].

Selection is the process of selecting which parent individuals will combine to make a new individual. This can be done randomly, but it can also be done by selecting the best (highest fitness) parents to steer the population more towards the best solution. Crossover is the process of two parents combining to make a new individual. A mutation changes some parts of individuals randomly, which is helpful for getting out of local optima. When elitism is applied, the best individuals of a current generation will also appear in the next generation to avoid the population becoming less fit [23]. The resulting offspring will make up the new population, for which the process will start again until a set number of generations is reached, or the best solution has sufficiently converged. This is shown as a flow diagram in Figure B.1.

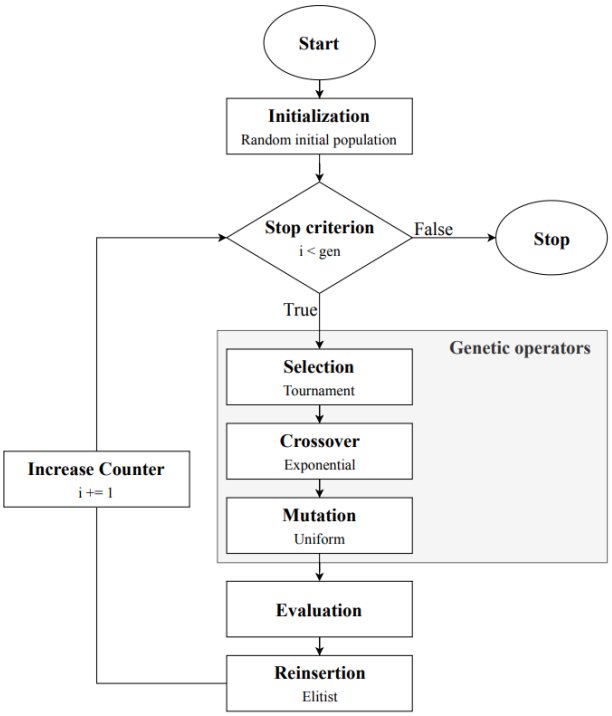


Figure B.1: Flow diagram of a generic genetic algorithm [24].

[1] https://nlopt.readthedocs.io/en/latest/NLopt_Algorithms/#bobyqa (retrieved in Oct. 2023)

C

Neural Networks

This research applies an artificial Neural Network (NN) to shape-based methods in low-thrust trajectory optimization. The NN is used in the optimization of the shape parameters by predicting the ΔV for a set of parameters. The optimization algorithm will then use this prediction to optimize these parameters to end up with the set of optimal parameters according to the NN. An NN is a type of supervised machine learning model. Supervised learning uses training data for which the output is known to train an algorithm to construct a function. When new data is presented, the algorithm can use this function to predict the new output. The output is discrete for classification problems and continuous for regression problems. This research deals with a continuous output and thus a regression problem. An artificial NN has proven to be a successful tool to deal with these problems[25].

As the name implies, artificial neural networks are inspired by biological neural networks. They have found many applications in fields such as image classification, speech recognition, and medical science, among others [10]. NNs generally consist of neurons ordered in an input layer, one or multiple hidden layers, and an output layer. A schematic representation of a feedforward neural network with three hidden layers is shown in Figure C.1.

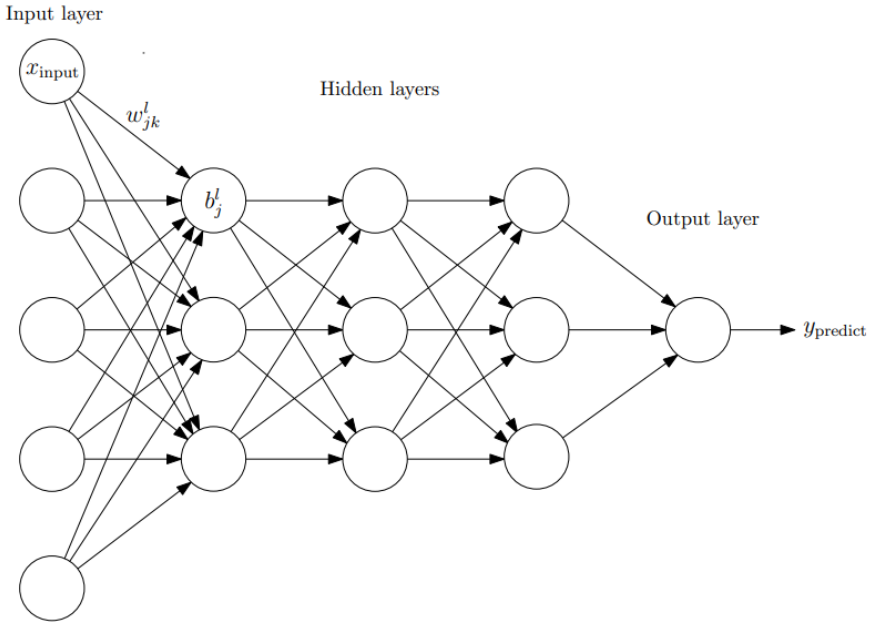


Figure C.1: Schematic representation of feedforward neural network with three hidden layers [24].

A single neuron j in layer l computes its activation a_j^l using Equation C.1 [26].

$$\boldsymbol{a}_j^l = \sigma(\sum_k \boldsymbol{w}_{jk}^l \boldsymbol{a}_k^{l-1} + \boldsymbol{b}_j^l) \tag{C.1}$$

Where k denotes the neurons of layer $l-1$, w_{jk} is the connecting weight between the two neurons j and k, b is the bias, and σ is an activation function. This activation is calculated for every neuron in every layer until the output layer is reached, which produces a singular value in the case of regression. For a single layer, the equation can be written in matrix form as in Equation C.2.

$$\boldsymbol{a}^l = \sigma(\boldsymbol{w}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l) \tag{C.2}$$

The size of the input and the output layer is determined by the problem at hand, with a regression problem generally having one output. The amount of neurons in the hidden layers is adjustable. An activation function is required to ensure that the activation is within a certain range desired for that specific network. Common activation functions are the sigmoid, which has an output between 0 and 1, the tanh, which has an output between -1 and 1, and the Rectified Linear Units (ReLU), which has an output between 0 and x. These functions are shown in the following equations.

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{C.3}$$

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{C.4}$$

$$\text{ReLU}(x) = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases} \tag{C.5}$$

Of these activation functions, the ReLU function has become the most popular in recent years because of its fast computation time and its ability to learn faster in networks that include many layers [10].

To obtain its weights and biases the neural network needs to be trained. This is done by using a training set for which the inputs and the outputs are known. The difference between the predicted outputs and the true outputs is calculated with a loss function. The derivative of this loss function with respect to each weight and bias can be traced back and calculated through all the different layers. This process is called backpropagation. The rate at which these weights and biases are changed depends on the selected learning rate. Furthermore, a validation set is used to prevent overfitting of the data.

The selected activation function and learning rate are two of the many hyperparameters in any neural network. Hyperparameters determine the way the NN is trained and its structure. Other examples of hyperparameters are the number of hidden layers and neurons per layer, the number of epochs in which the network is trained, and batch size, to name a few. The best option for these hyperparameters is problem- and network-dependent and thus should be carefully picked through fine-tuning the network. However, good initial guesses and suitable alternatives can often be found in literature. The properties and hyperparameters of the neural network used in this research are shown in Table C.1.

Property	**Setting**
Number of hidden layers	7
Neurons per layer	64
Activation function	ReLU
Optimizer	Adam
Batch size	200
Learning rate	0.001
Training/test data split	80/20
Weight initialization	Randomly between -1 and 1
Bias initialization	0
Loss function	MSE

Table C.1: Neural network settings used in this research.

D

Verification & Validation

To ensure that the thesis project is scientifically valuable, the models that it uses have to be verified and validated. The implementation of hodographic shaping, the linked trajectories that are optimized using the GA, and the behavior of the neural network are verified with respect to literature. The methods that are used to implement the dynamical model, the optimization methods, and the neural network are validated to guarantee they perform as intended.

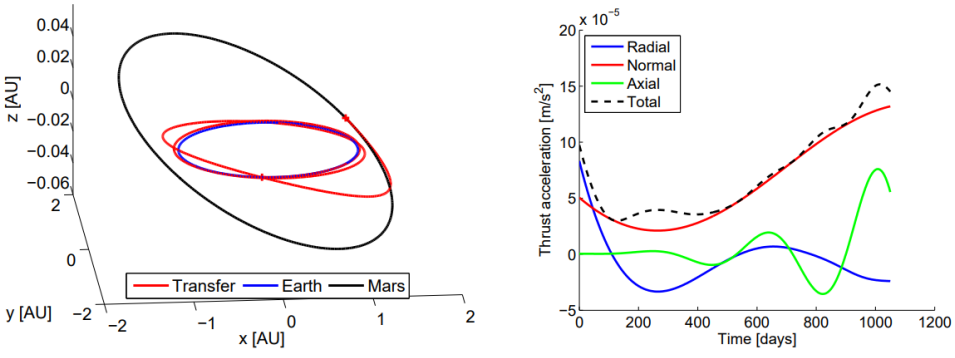
D.1. Verification

D.1.1. Hodographic Shaping Implementation

The hodographic shaping method is verified using the results Gondelach found in his paper for an Earth-Mars trajectory [14]. The trajectory and its thrust profile of the 0-DoF method found by Gondelach are shown in Figure D.1, and the reproduced results are shown in Figure D.2. Similarly, the same is done for the 6-DoF method in Figure D.3 and Figure D.4. The input and the results of both methods can be seen in Table D.1. The verification of the 6-DoF method also verifies the implementation of the optimization procedure for the free parameters.

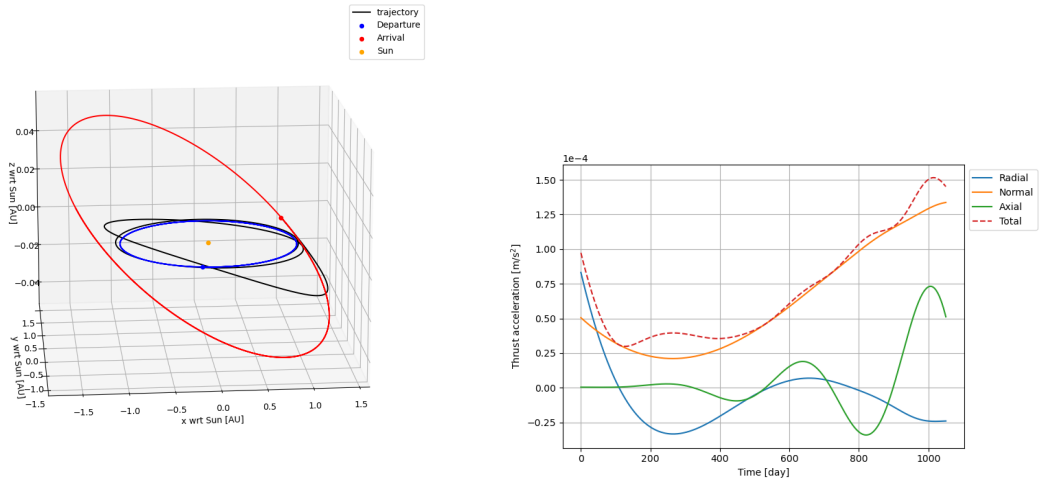
Method	Departure date [mjd2000]	ToF [days]	ΔV [km/s]	Max. Thrust [ms^{-2}]
0-DoF [14]	10025	1050	6.342	1.51×10^{-4}
0-DoF this work	10025	1050	6.341	1.52×10^{-4}
6-DoF [14]	9985	1100	5.771	1.50×10^{-4}
6-DoF this work	9985	1100	5.771	1.50×10^{-4}

Table D.1: Verification results for the implementation of the hodographic shaping method.



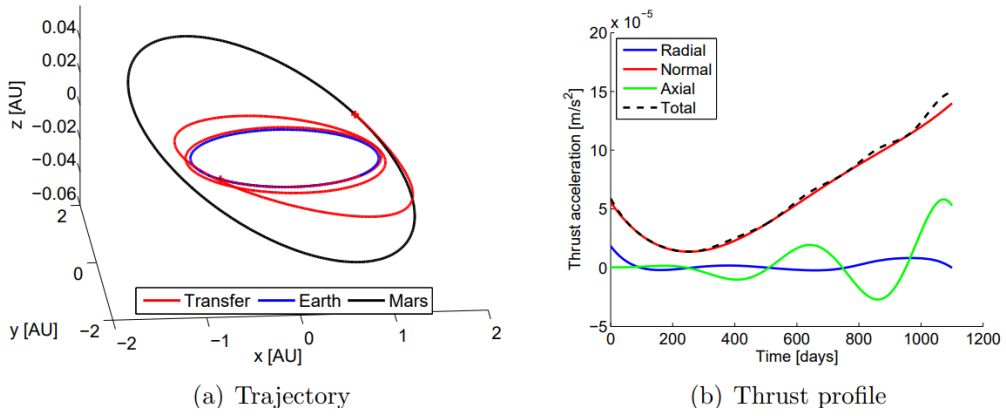
(a) Trajectory (b) Thrust profile

Figure D.1: Best found trajectory for 0-DoF shaping method by Gondelach [14].



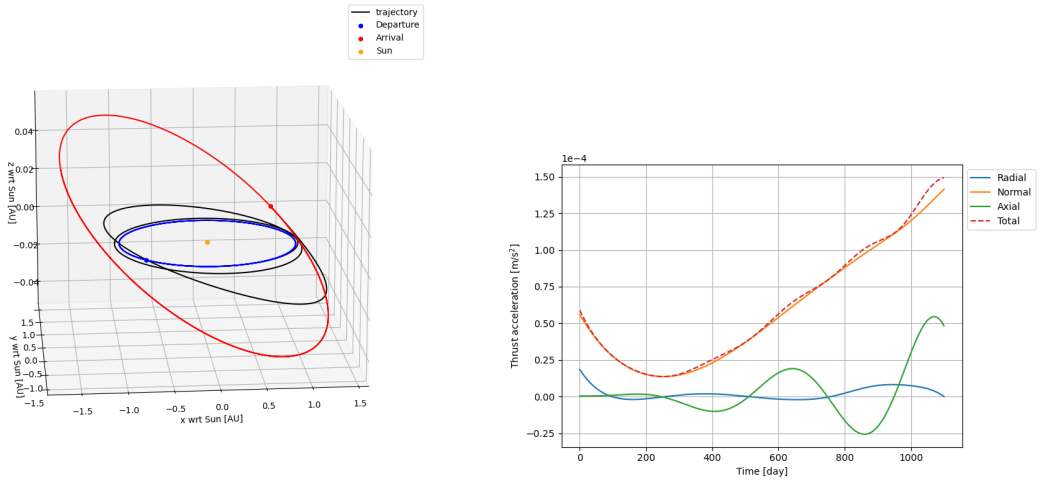
(a) Trajectory **(b)** Thrust profile

Figure D.2: Recreated 0-DoF trajectory for verification.



(a) Trajectory (b) Thrust profile

Figure D.3: Best found trajectory for 6-DoF shaping method by Gondelach [14].



(a) Trajectory **(b)** Thrust profile

Figure D.4: Recreated 6-DoF trajectory for verification.

D.1.2. Linked trajectories

The linked trajectories, that are optimized using the GA, are verified by recreating the best trajectories Stubbig found in his work [24]. He presented the inputs he used and the resulting ΔV and thrust profile. The same inputs are used here, after which the ΔV and thrust profile can be compared. The thrust profile for the DAWN mission trajectory he found and the one recreated in this work are shown in Figure D.5 and Figure D.6. The same is done for the GTOC 2 trajectory in Figure D.7 and Figure D.8. The ΔV's of all trajectories are given in Table D.2. It can be seen there are very slight differences in the results obtained by Stubbig and in this work. However, these differences are deemed sufficiently small that these results still give confidence that the linked trajectories implementation is solid enough for its application in this research.

Test Case	ΔV [km/s]
DAWN [24]	11.8
DAWN this work	11.9
GTOC 2 [24]	26.9
GTOC 2 this work	27.1

Table D.2: Verification of implementation of the linked trajectories.

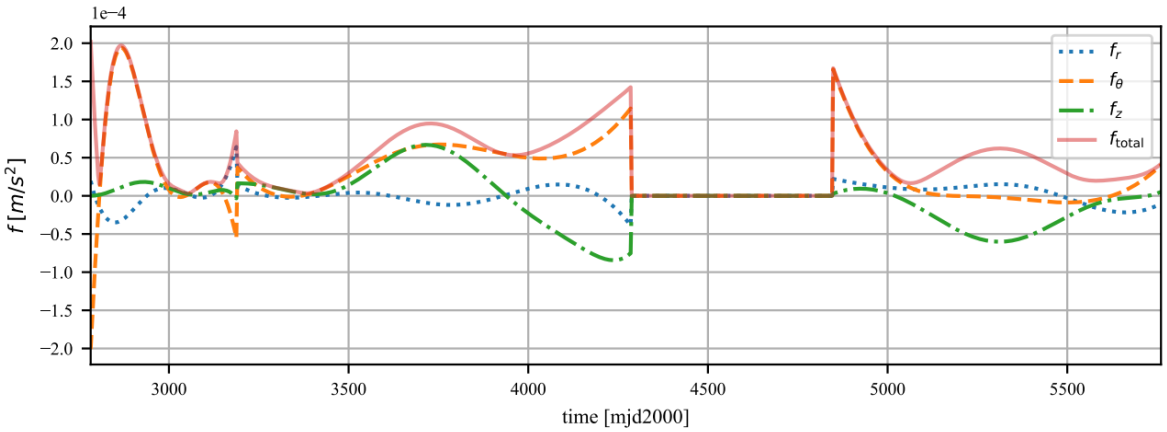


Figure D.5: Thrust profile of best DAWN trajectory found by Stubbig [24].

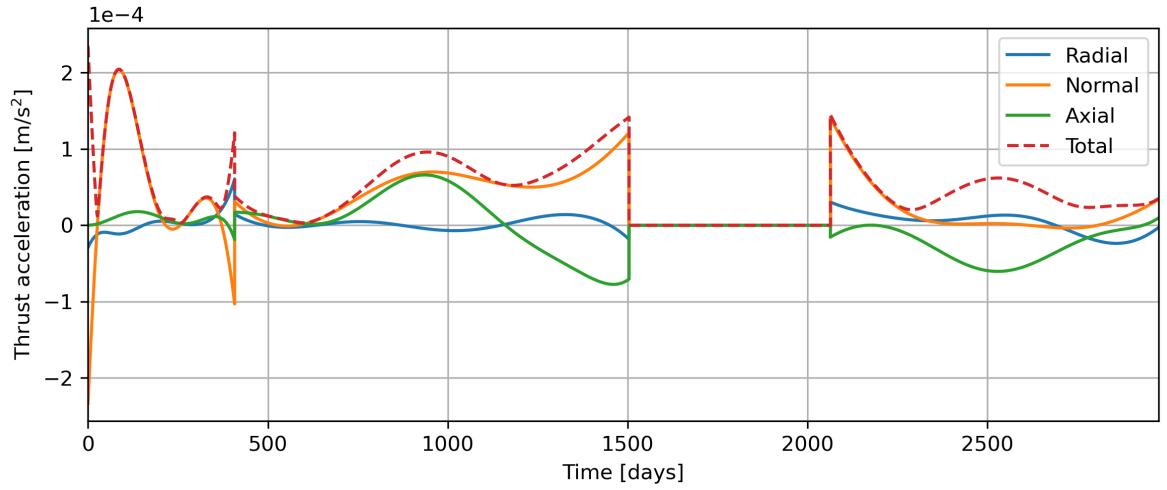


Figure D.6: Recreated thrust profile of best DAWN trajectory.

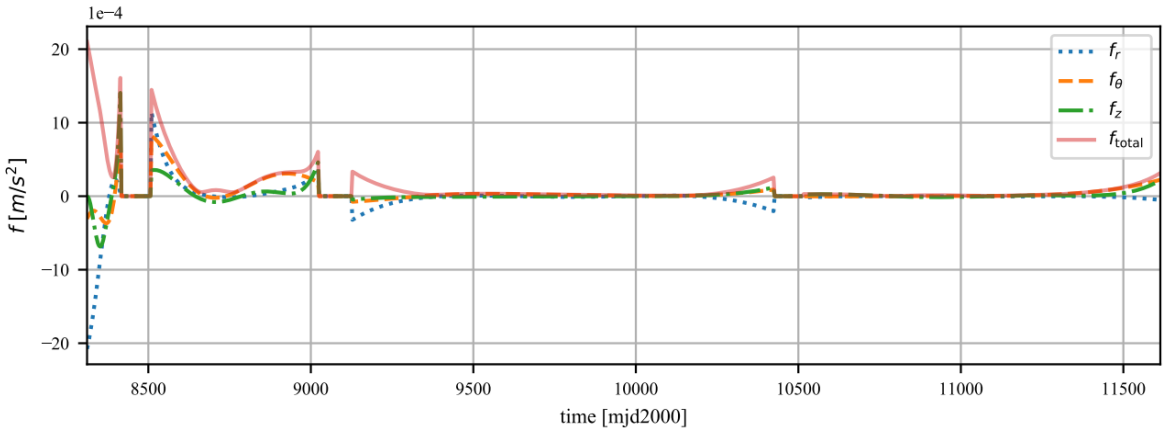


Figure D.7: Thrust profile of best GTOC 2 trajectory found by Stubbig [24].

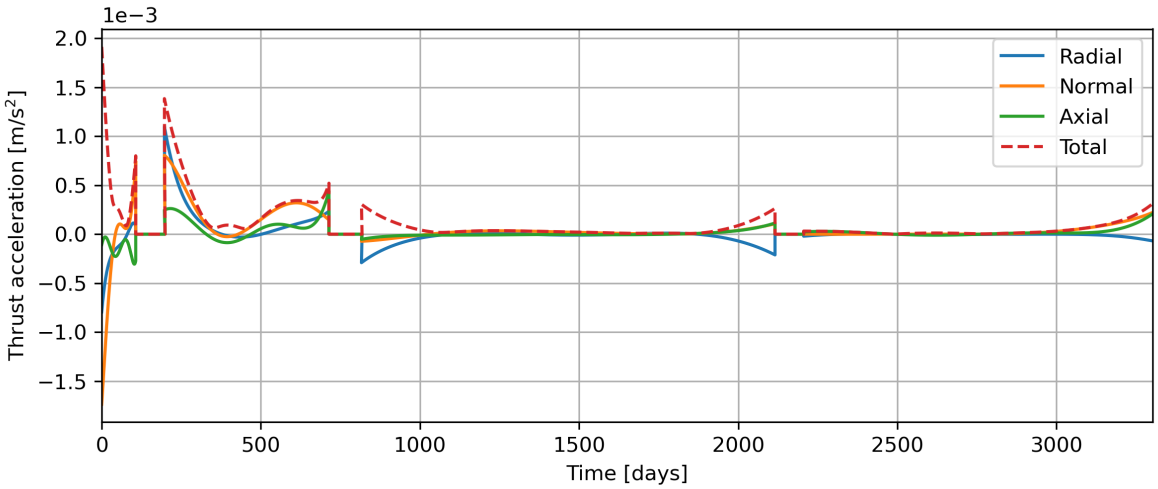


Figure D.8: Recreated thrust profile of best GTOC 2 trajectory.

D.1.3. Neural Network

The neural network is verified by observing if the model's behavior changes as predicted when certain parameters are changed. This is done by taking statements regarding well-known NN behavior from literature and comparing them with our own findings. These statements involve the network size, the batch size, and the learning rate [27]. Firstly, we know that increasing the number of layers of an NN generally increases its capacity to learn complex relationships and thus its accuracy but also increases its training time. In Figure D.9 we can confirm that this trend is clearly present in the data regarding accuracy. For training times this trend is not as obvious but it can still be seen that larger networks generally take longer to train than smaller ones. Secondly, we know that increasing the number of neurons per layer also generally increases the NN's capacity to learn complex relationships and its accuracy but also increases its training time. In Figure D.9 this trend is clear for both the MSE and training times.

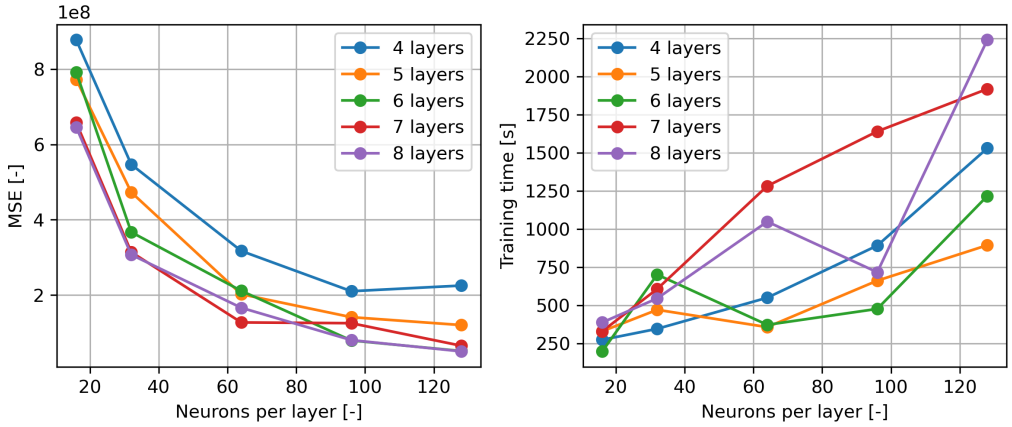


Figure D.9: The MSE and training time for different NN architectures for the general model 3.

Thirdly, we know that increasing the batch size for the training data typically decreases the NN training time but also decreases its accuracy. In Figure D.10, this trend is clear for both the MSE and training times. Finally, we know that increasing the learning rate allows for faster convergence and thus decreased training times. Typically it will, however, also overshoot the optimum and thus decrease the network accuracy. In Figure D.10, both trends also occur in our neural network, very clearly for training times and a bit more general for the MSE.

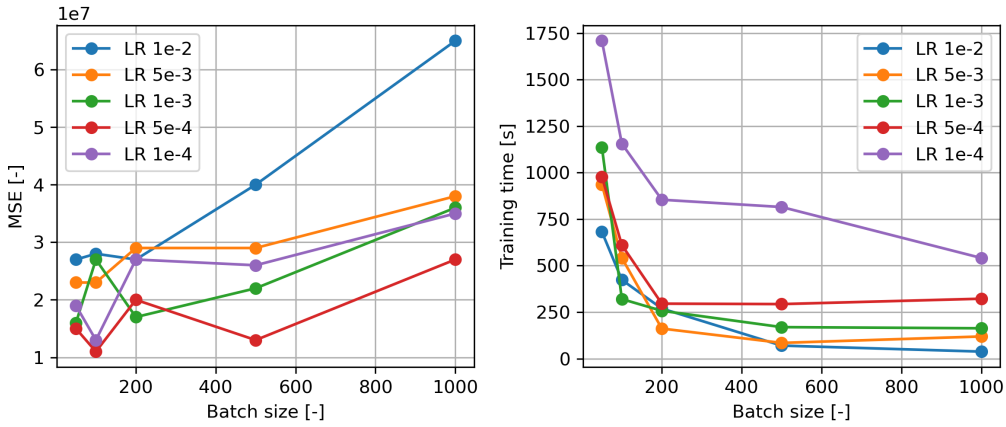


Figure D.10: The MSE and training time for different batch sizes and learning rates for the general model 3.

D.2. Validation

D.2.1. Dynamical Model

For preliminary low-thrust mission design, a simplified astrodynamical model using only a central attractor and a thrust force is often used [7]. Shape-based methods are generally accepted as a usable tool for preliminary mission design [5] and hodographic shaping is a shape-based method that is often considered [28][29]. The astrodynamical model is implemented with the Python interface of the TU Delft Astrodynamics Toolbox (Tudat) [30]. Tudat provides tools for astrodynamical calculations and numerical simulation. This toolkit has been validated to give useful scientific output and is used in numerous publications, for instance, recently by Hu et al. [31]. For these reasons, the dynamical model is deemed to give useful physical and scientific results.

D.2.2. Optimization Methods

The optimization methods are implemented using PyGMO. PyGMO is the Python adaptation of PaGMO (Parallel Global Multiobjective Optimization), a library designed by the European Space Agency (ESA) for optimization[20]. It is a flexible library able to solve single-objective and multi-objective optimization problems and is used in this research for free parameter optimization and for the construction of a genetic algorithm. It is often used in research for optimization problems in various disciplines [32][33] and therefore is validated for this research.

D.2.3. Neural Network

The neural network is implemented with Scikit-learn (Sklearn) [34]. Sklearn is an open-source library for artificial intelligence and machine learning that can be used in Python. It offers a wide variety of machine-learning models and allows for modifying and optimizing these models. This library is also widely used across various disciplines such as speech recognition [35] and protein structure prediction [36] but also in the optimization of low-thrust trajectories [37]. For these reasons, the neural network is deemed to be able to give scientifically valuable results.

E

Additional Results

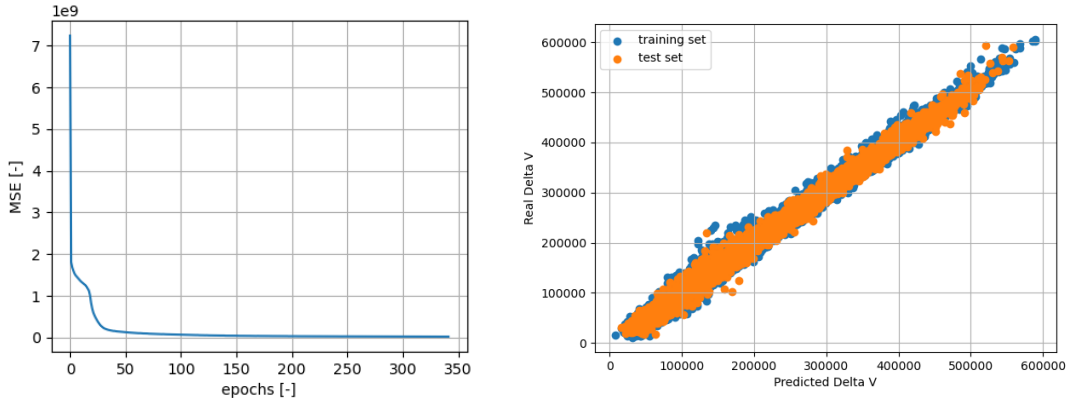
This appendix contains additional results which support the conclusions of the research but for which there was no place in the paper. This appendix aims to provide further insight into the way that the core findings of our paper were obtained. These results contain more information on the tuning of the neural network, a more extensive look at the grid search results, and the trajectories that were found by the genetic algorithm using the ML model.

E.1. Neural Network

Three aspects of the tuning of the neural network are further explored: the effect that generalization has on the network, the determination of the amount of training data, and the determination of network size.

E.1.1. Effect of Generalization

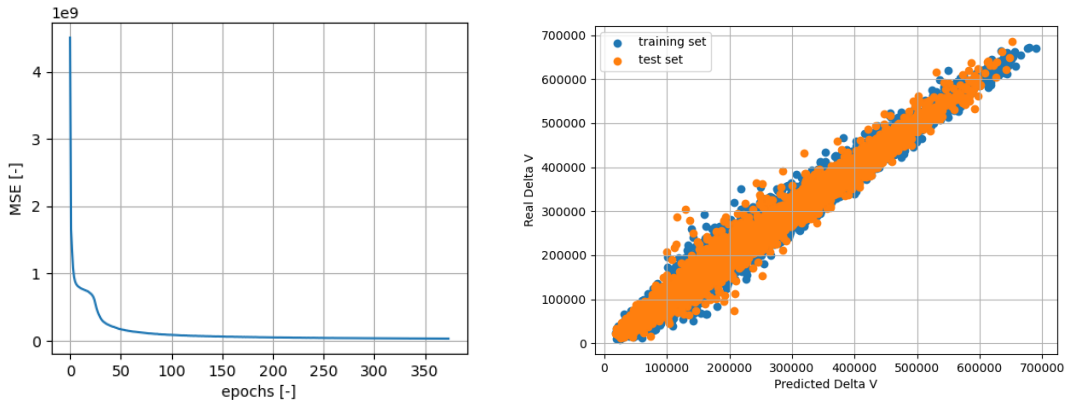
The effects of generalization are not only seen in the ability of the ML model to converge to optimal results but they can also be seen when looking at the training performance of the NN. The loss curve and scatter plot of actual versus predicted values for all three models are given below. The results of model 1, which is only trained on Earth-Mars N = 2 transfers, are given in Figure E.1. The results of model 2, which is only trained on Earth-Mars N = [0, 1, 2, 3] transfers, are given in Figure E.2. The results of model 3, the fully generalized model, are given in Figure E.3.



(a) Loss curve

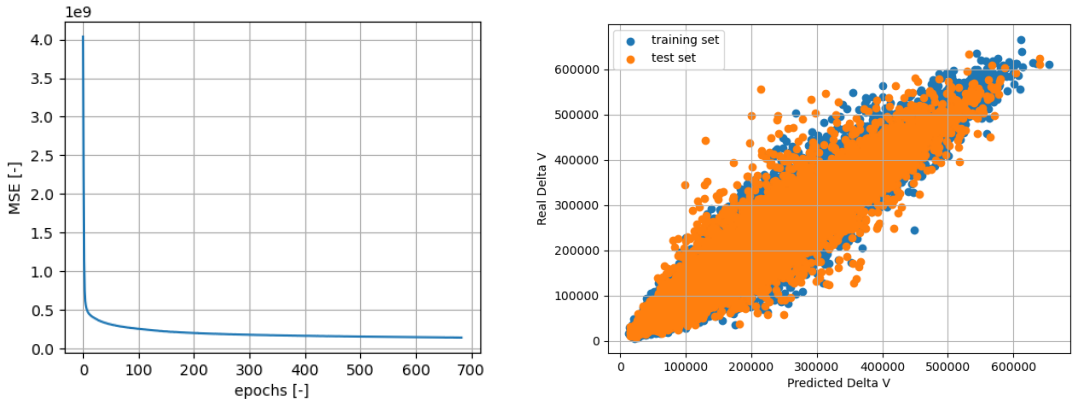
(b) Scatter plot

Figure E.1: Loss curve and scatter plot of actual versus predicted values of model 1 neural network training.



(a) Loss curve **(b)** Scatter plot

Figure E.2: Loss curve and scatter plot of actual versus predicted values of model 2 neural network training.



(a) Loss curve **(b)** Scatter plot

Figure E.3: Loss curve and scatter plot of actual versus predicted values of model 3 neural network training.

Three things stand out when comparing the training results of the three models. From the loss curves, it is clear that the NN takes more epochs to converge, and the NN converges to a higher final mean squared error (MSE) upon further generalization. Furthermore, the scatter plot results become more spread out. These results confirm expectations that the neural network performs worse and its regression capabilities decrease when the input data gets more generalized. Despite the decrease in performance, we have seen that the results of the general neural network remain acceptable for most test cases.

E.1.2. Network Architecture

In the paper, the neural network architecture was determined by analyzing the effect of the variation of the number of layers and the number of neurons per layer on a set of performance metrics. Only the results of this analysis of model 1 were shown. Here, the results of the same analysis on models 2 and 3 are shown in Figure E.5 and Figure E.6, respectively. This is done in order to confirm it was a correct assumption to take the same architecture for all models. For the sake of completeness, the results of model 1 are again shown in Figure E.4.

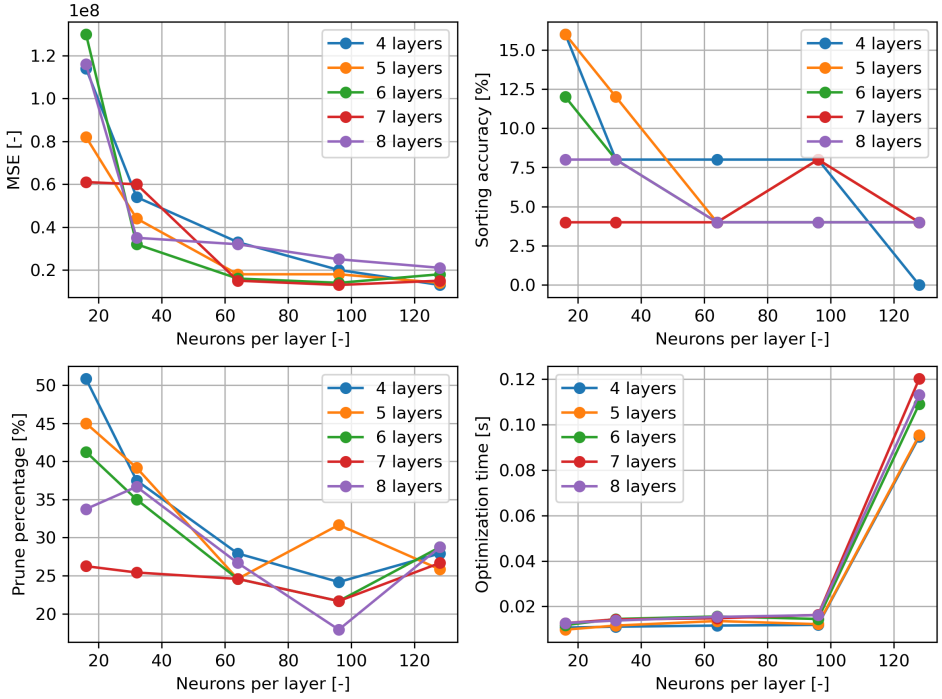


Figure E.4: Performance metrics behavior for model 1 in NN architecture grid search for 125,000 training samples.

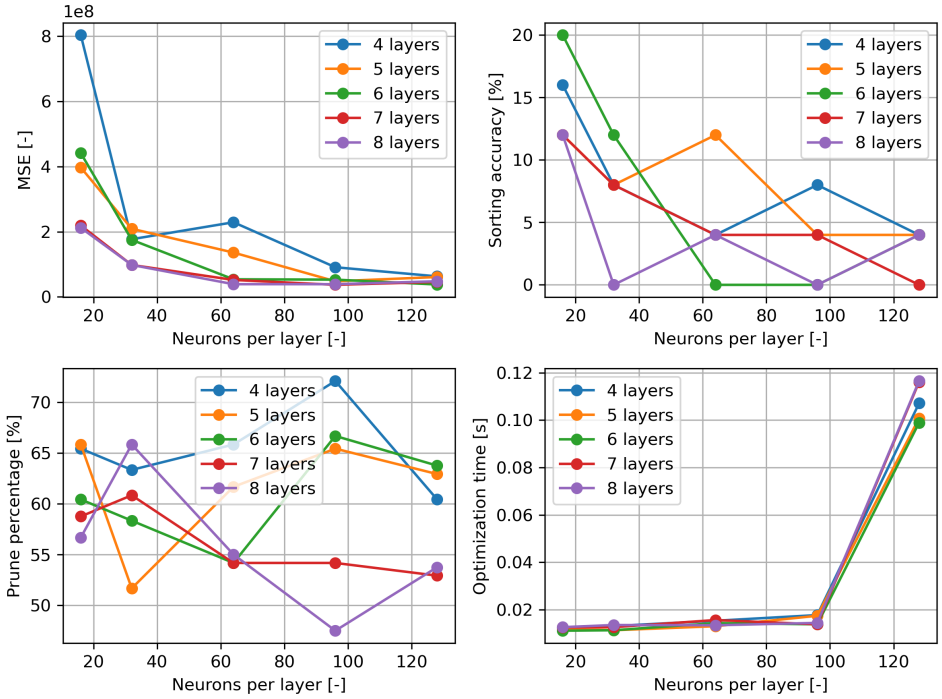


Figure E.5: Performance metrics behavior for model 2 in NN architecture grid search for 125,000 training samples.

If we compare all figures, there are four architectures that perform the best in almost every metric: networks with 7 or 8 hidden layers, with 64 or 96 neurons per layer. Networks with 128 neurons per layer are disregarded, as they increase the optimization time drastically. In order to keep the approach consistent, the least complex architecture of these four was used for all models. This means that all models used a neural network with a 7x64 architecture.

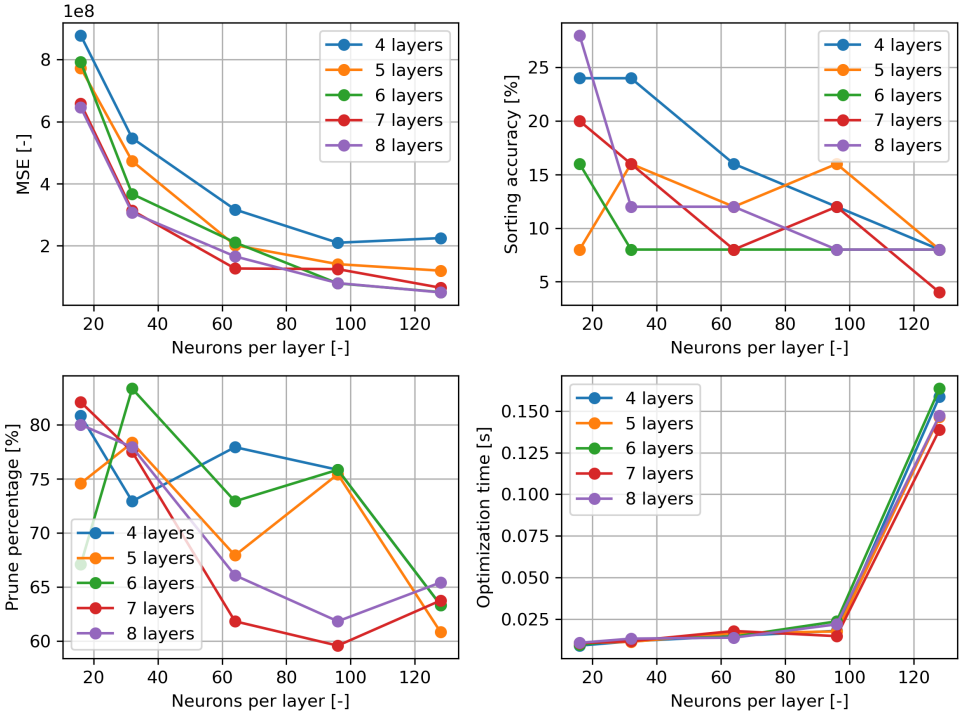


Figure E.6: Performance metrics behavior for model 3 in NN architecture grid search for 125,000 training samples.

E.1.3. Amount of Training Data

The ideal amount of training data per model was determined using the same performance metrics that were used to determine network architecture. The results are shown in Figure E.7

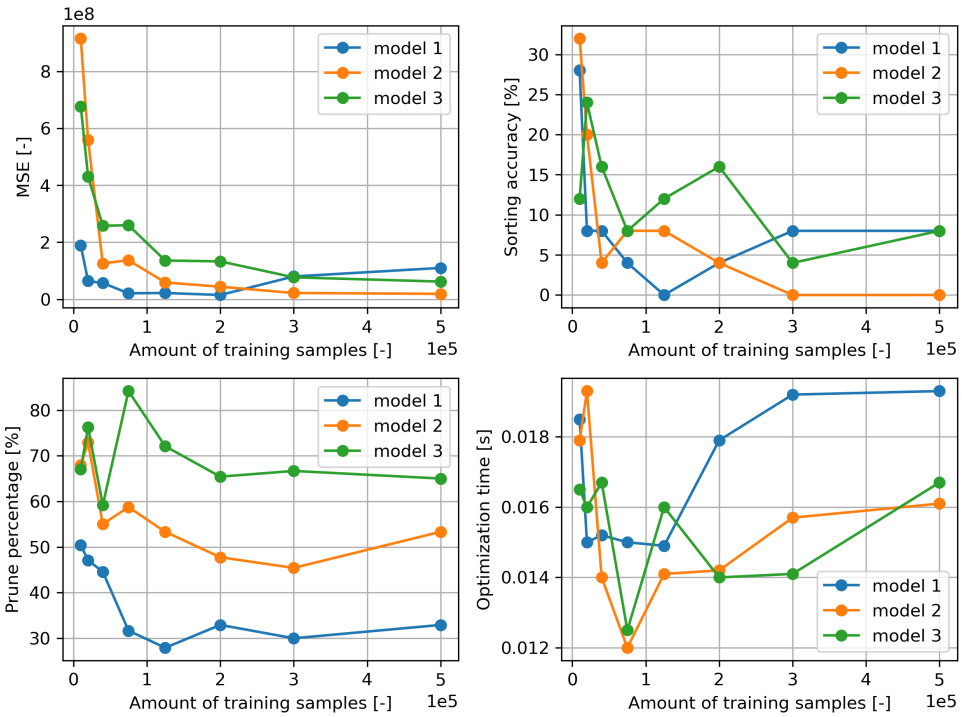


Figure E.7: Model performance when varying the amount of training data, using NN with a 7x64 architecture.

The trend present in the majority of the performance metrics is that performance improves when the amount of training data is increased. However, the model performs worse once this amount surpasses a certain number for some metrics. This is most prevalent in the optimization time. It seems that model performance plateaus, but its complexity still increases which has an effect on the time it takes for the model to work. This effect is undesirable, and for that reason, it was decided that the ideal amount of training samples is: 125,000 for model 1, 200,000 for model 2, and 300,000 for model 3.

E.2. Grid Search Results

The grid searches for all individual transfers that were assessed are shown in this section. In the paper, only the single most optimal trajectory per grid search was presented in order to make a comparison, since that was the way that we defined the performance of a shaping method. It is, however, also interesting to have a look at the grid searches themselves, as there it can be seen how our ML models compare to the traditional methods in places of lesser optimality. The 10 most optimal trajectories found by that method are indicated in red on every grid to show what every method indicates as an optimal region. Finally, the minimum ΔV found and the time it took to complete the grid search by every method are tabulated for all transfers.

Earth - Mars

The results of performing the grid search with model 1 on an Earth-Mars N = 2 trajectory are shown in Figure E.8, where they are compared with the traditional approach. Additional information on the results is tabulated in Table E.1.

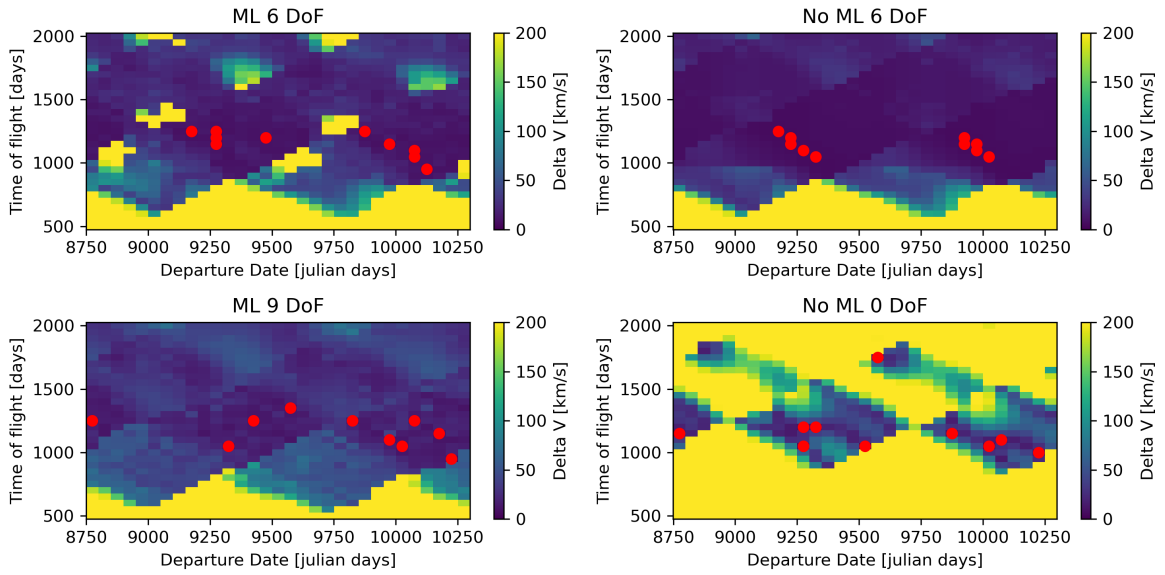


Figure E.8: Results of grid search of model 1 for Earth-Mars, N = 2 trajectories. The 10 best solutions for every grid search are indicated with a red dot.

Method	Minimum ΔV [km/s]	CPU Time [s]
No ML 6-DoF	5.78	2267
No ML 0-DoF	7.55	19.13
ML 6-DoF	6.91	18.61
ML 9-DoF	9.11	31.63

Table E.1: Information on the grid search results of model 1 for Earth-Mars, N =2 trajectories.

The ML models are generally able to identify the patterns that occur in such a grid search. In the 6-DoF model, there are some artifacts remaining, but for 9-DoF the graph smoothens out, and a result is achieved that is visually very similar to the traditional 6-DoF method. On the other hand, we see that the visual similarity does not yield more optimal results. The locations of the best results are more similar to the traditional method for the 6-DoF model than for the 9-DoF model. From the tabulated results, it can be concluded again how much faster the ML model is in the optimization of free parameters compared to the no-ML model. The grid search results of models 2 and 3 on an Earth-Mars transfer with N = [0, 1, 2, 3] are shown in Figure E.9 and Figure E.10 respectively. Additional information is given in Table E.2.

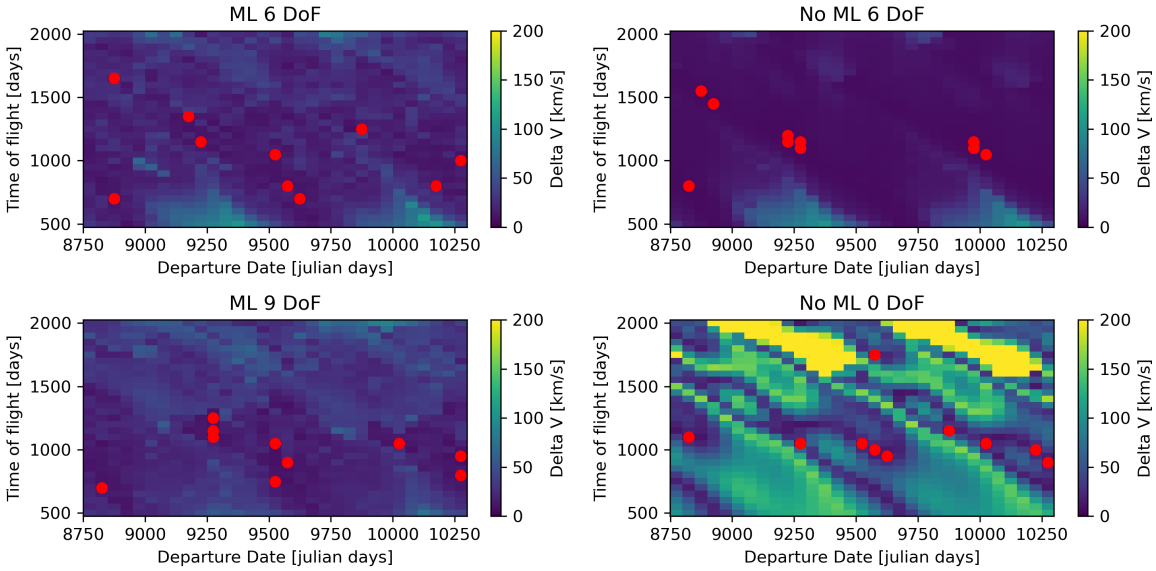


Figure E.9: Results of grid search of model 2 for Earth-Mars, N = [0, 1, 2, 3]. The 10 best solutions for every grid search are indicated with a red dot.

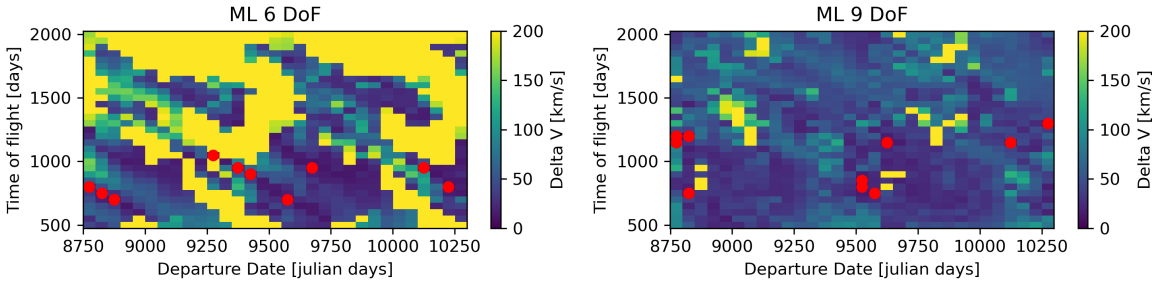


Figure E.10: Results of grid search of model 3 for Earth-Mars, N = [0, 1, 2, 3]. The 10 best solutions for every grid search are indicated with a red dot.

Method	Minimum ΔV [km/s]	CPU Time [s]
No ML 6-DoF	5.55	8612
No ML 0-DoF	7.55	73.82
Model 2 6-DoF	7.52	117.1
Model 2 9-DoF	6.67	136.0
Model 3 6-DoF	8.78	142.4
Model 3 9-DoF	8.72	160.4

Table E.2: Information on the grid search results of models 2 and 3 for Earth-Mars, N = [0, 1, 2, 3] trajectories.

With regards to model 2, we have a similar result as for model 1, where the ML models and no-ML seem to get similar results across the grid search. Model 3 performs significantly worse, however, and bears no resemblance. The model 3 Earth-Mars transfer was the test case in which our model performed worst in terms of optimality. It is clear that this poor performance did not only concern the optimum but was prevalent across the entire grid search.

Earth - Ceres

The results of performing the grid search with model 1 on an Earth-Ceres N = 2 trajectory are shown in Figure E.11. Additional information on the results is tabulated in Table E.3.

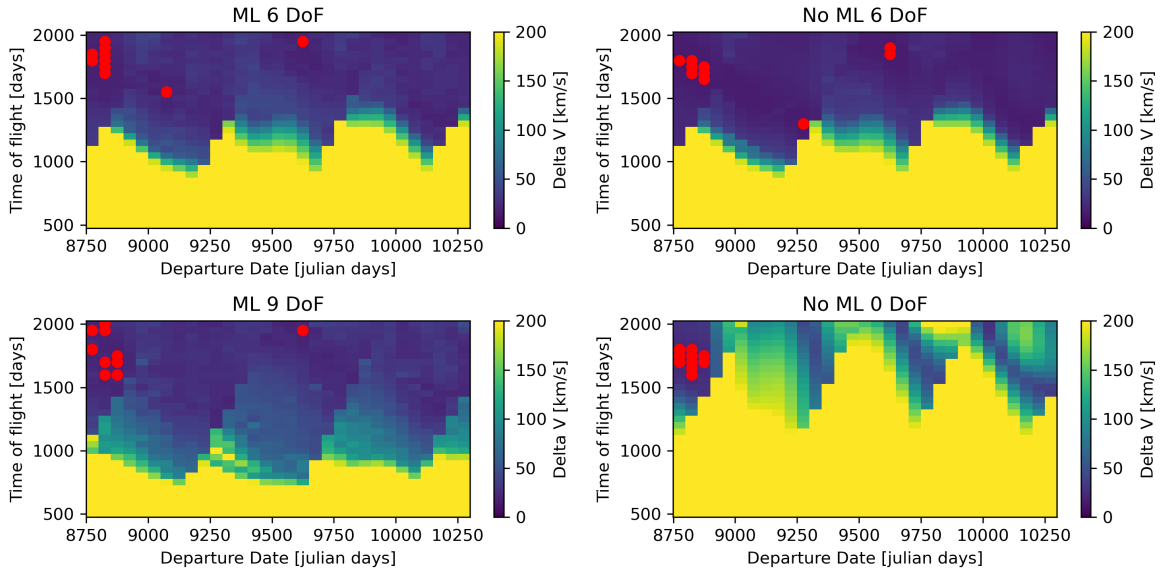


Figure E.11: Results of grid search of model 1 for Earth-Mars, N = 2 trajectories. The 10 best solutions for every grid search are indicated with a red dot.

Method	Minimum $\triangle$V [km/s]	CPU Time [s]
No ML 6-DoF	14.30	1965
No ML 0-DoF	20.88	22.09
ML 6-DoF	15.57	21.67
ML 9-DoF	17.28	35.34

Table E.3: Information on the grid search results of model 1 for Earth-Ceres, N =2 trajectories.

All methods find the most optimal trajectories in the same region for this transfer. The ability of the 9-DoF ML model to improve upon some of the less optimal trajectories found by the 6-DoF no-ML model is well visible in these plots. Upon inspection, it turns out that the 9-DoF ML model can find trajectories more optimal by more than 200 km/s in some regions. This shows promise that if the 9-DoF model can be improved, sufficient gains in optimality could also be made in more optimal regions. The grid search results of models 2 and 3 on an Earth-Ceres transfer with N = [0, 1, 2, 3] are shown in Figure E.12 and Figure E.13 respectively. Additional information is given in Table E.4.

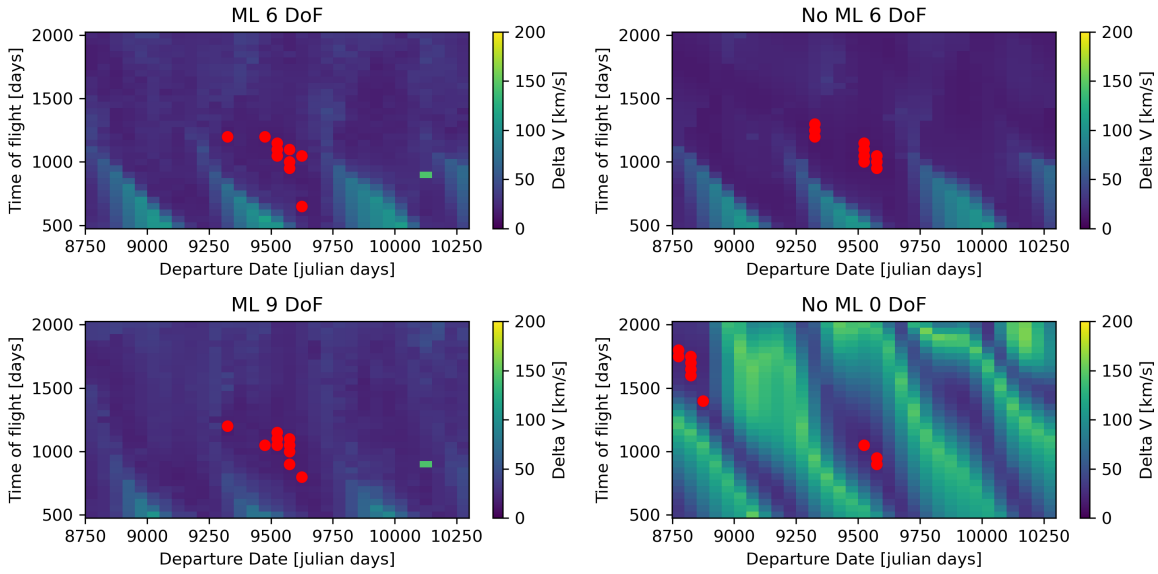


Figure E.12: Results of grid search of model 2 for Earth-Ceres, N = [0, 1, 2, 3]. The 10 best solutions for every grid search are indicated with a red dot.

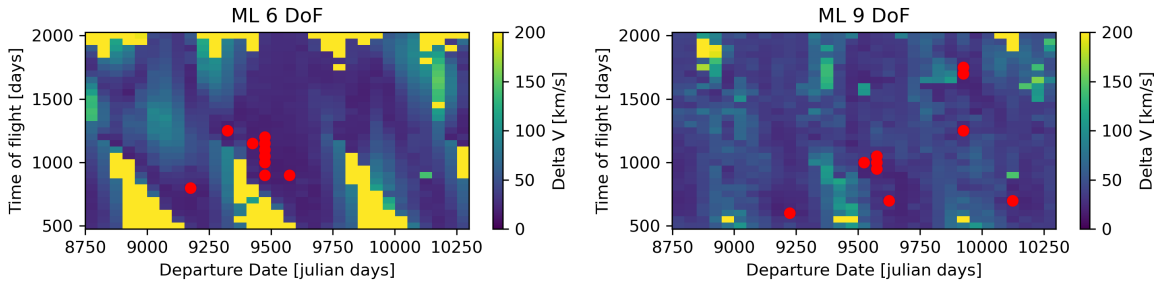


Figure E.13: Results of grid search of model 3 for Earth-Ceres, N = [0, 1, 2, 3]. The 10 best solutions for every grid search are indicated with a red dot.

Method	Minimum ΔV [km/s]	CPU Time [s]
No ML 6-DoF	13.28	9313
No ML 0-DoF	20.88	79.63
Model 2 6-DoF	14.69	118.7
Model 2 9-DoF	15.48	170.5
Model 3 6-DoF	14.77	145.0
Model 3 9-DoF	17.57	168.0

Table E.4: Information on the grid search results of models 2 and 3 for Earth-Ceres, N = [0, 1, 2, 3] trajectories.

We see here that the model retains the ability to recognize the patterns that occur in the grid search when generalizing, unlike it did for the Earth-Mars transfer. This leads to a good overall performance by the ML models for an Earth-Ceres transfer.

Mars-Earth

The grid search results of model 3 on a Mars-Earth transfer with N = [0, 1, 2, 3] are shown in Figure E.14. Additional information is given in Table E.5.

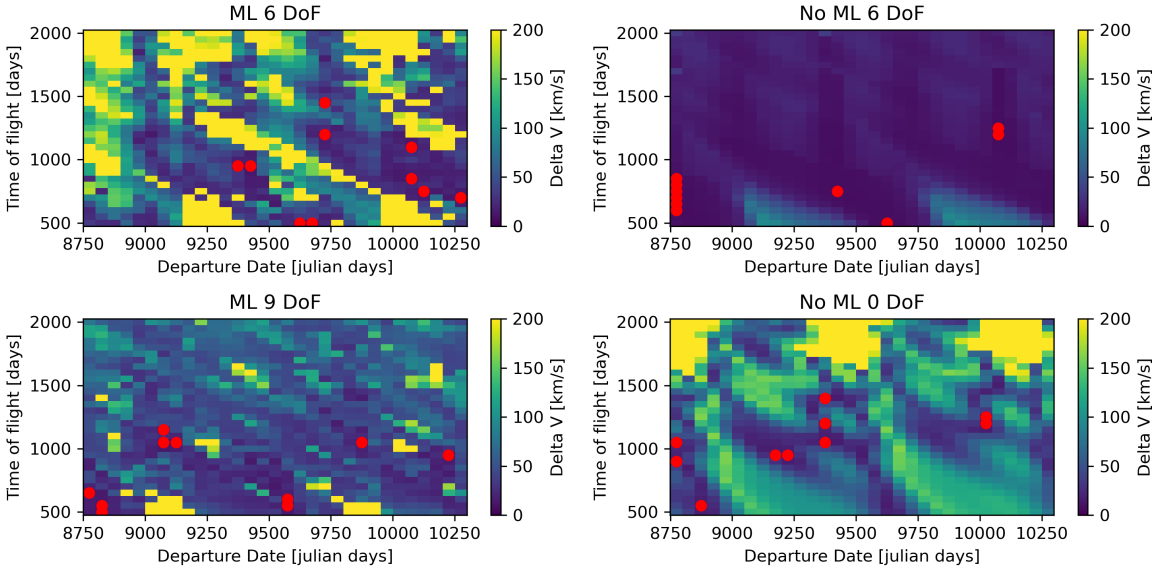


Figure E.14: Results of grid search of model 3 for Mars-Earth, N = [0, 1, 2, 3]. The 10 best solutions for every grid search are indicated with a red dot.

Method	Minimum ΔV [km/s]	CPU Time [s]
No ML 6-DoF	5.69	10300
No ML 0-DoF	12.85	94.66
ML 6-DoF	10.60	140.4
ML 9-DoF	9.11	161.4

Table E.5: Information on the grid search results of model 3 for Mars-Earth, N = [0, 1, 2, 3] trajectories

When looking at the numbers, we see that an inward transfer does not affect the performance of the ML model compared to the outward transfer, but visually, the grid search by the ML seems to perform poorly. What is also interesting is that the 9-DoF performance is improved. For this transfer, it shows its ability to find the most optimal regions and a more optimal overall trajectory.

Vesta-Jupiter

The grid search results of model 3 on a Vesta-Jupiter transfer with N = [0, 1, 2, 3] are shown in Figure E.15. Additional information is given in Table E.6. This particular transfer was not included in the training data of the NN for this analysis.

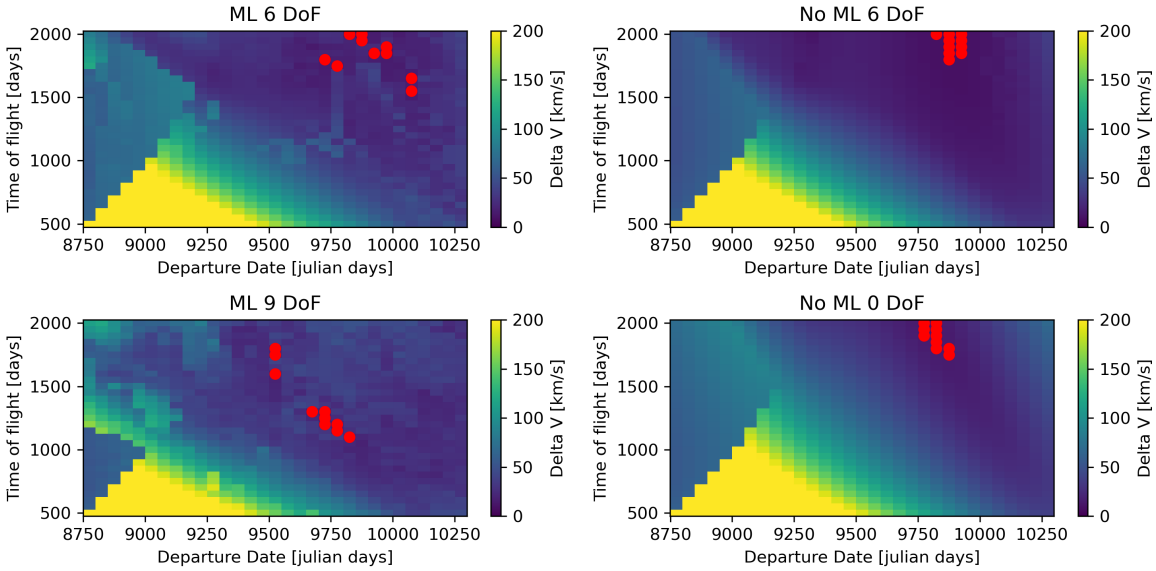


Figure E.15: Results of grid search of model 3 for Vesta-Jupiter, N = [0, 1, 2, 3]. The 10 best solutions for every grid search are indicated with a red dot.

Method	Minimum ΔV [km/s]	CPU Time [s]
No ML 6-DoF	8.20	6557
No ML 0-DoF	13.84	94.13
ML 6-DoF	10.36	137.1
ML 9-DoF	16.65	181.1

Table E.6: Information on the grid search results of model 3 for Vesta-Jupiter, N = [0, 1, 2, 3] trajectories

Despite not having this transfer in the training data, the ML models perform surprisingly well on it. Aside from some artifacts, the models distinguish the same patterns as the no-ML models. The 6-DoF ML model even has all its optimal trajectories in the same regions as the no-ML models. These results support our findings on the capabilities, limitations, and robustness of the ML model.

E.3. DAWN Trajectory

This section shows the best trajectory found by the ML model and the GA for the DAWN test case. This is done to give the GA results more physical meaning and show that the model is capable of generating realistic preliminary trajectories for more complex missions. The trajectory shown is the best trajectory found after 30 generations by the implementation of the specific ML model, which was only trained and used on the final Vesta-Ceres transfer. The total ΔV for this transfer is 15.26 km/s. A 3-D view, a 2-D view, and the thrust profile of this trajectory are shown in Figure E.16, Figure E.17, and Figure E.18 respectively.

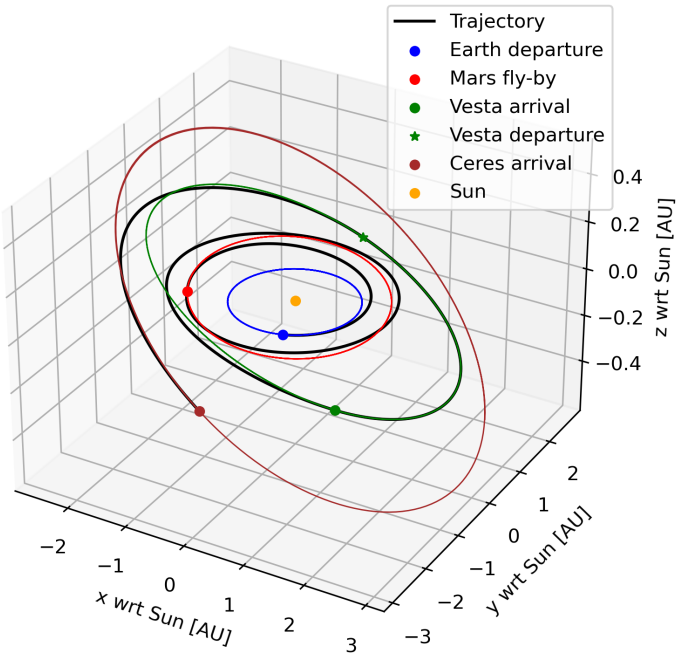


Figure E.16: 3-D view of the optimal DAWN trajectory found after 30 generations by the specific model and the GA.

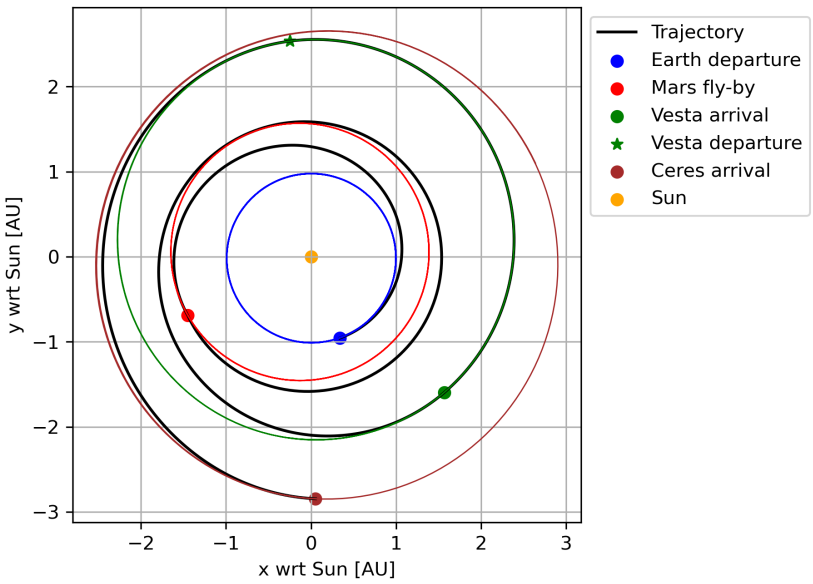


Figure E.17: 2-D view of the optimal DAWN trajectory found after 30 generations by the specific model and the GA.

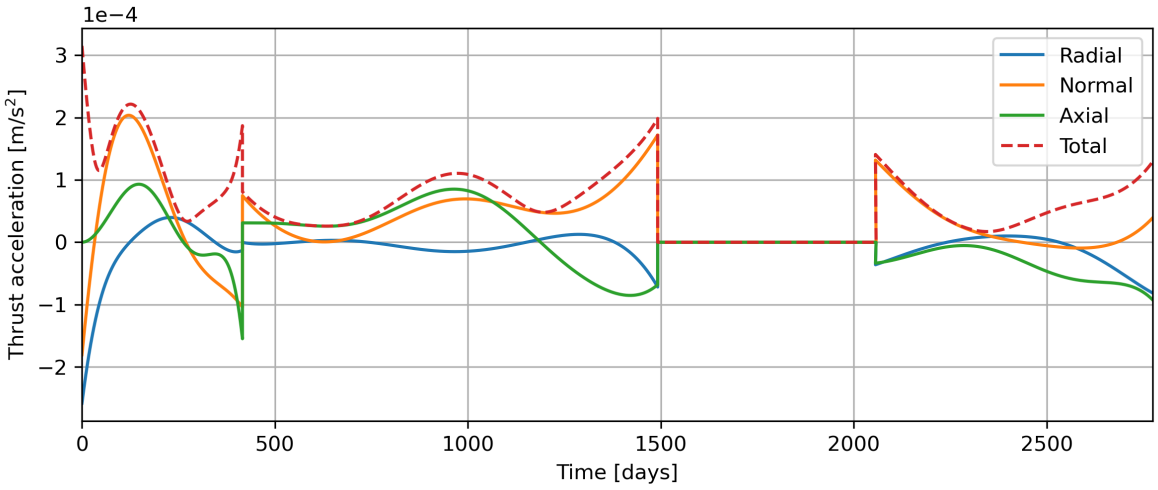


Figure E.18: Thrust profile of the optimal DAWN trajectory found after 30 generations by the specific model and the GA.

E.4. GTOC 2 Trajectory

The best trajectory found by the ML model and the GA for the GTOC 2 test case is shown here. The trajectory shown is the best trajectory found after 50 generations by the implementation of the specific ML model. This model was trained on all combinations of transfers between the asteroids the spacecraft had to visit. The total ΔV for this transfer is 55.36 km/s. A 3-D view, a 2-D view, and the thrust profile of this trajectory are shown in Figure E.19, Figure E.20, and Figure E.21.

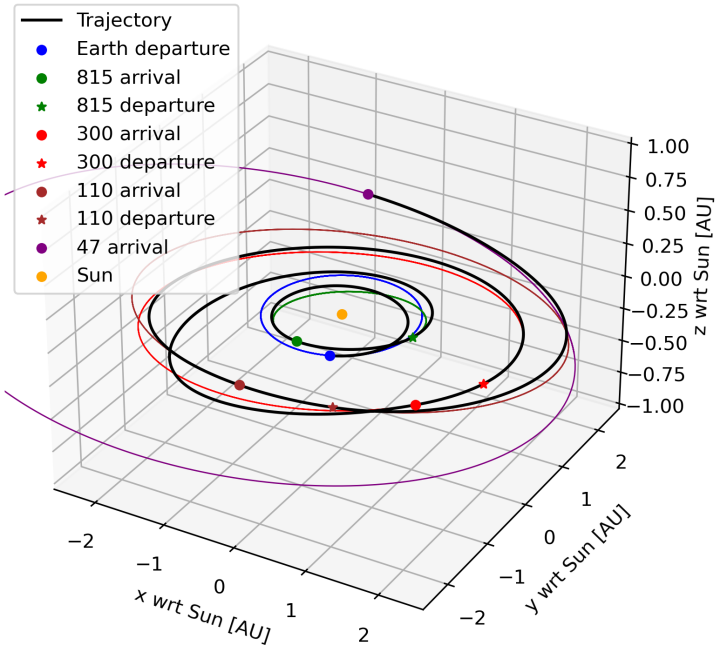


Figure E.19: 3-D view of the optimal GTOC trajectory found after 50 generations by the specific model and the GA.

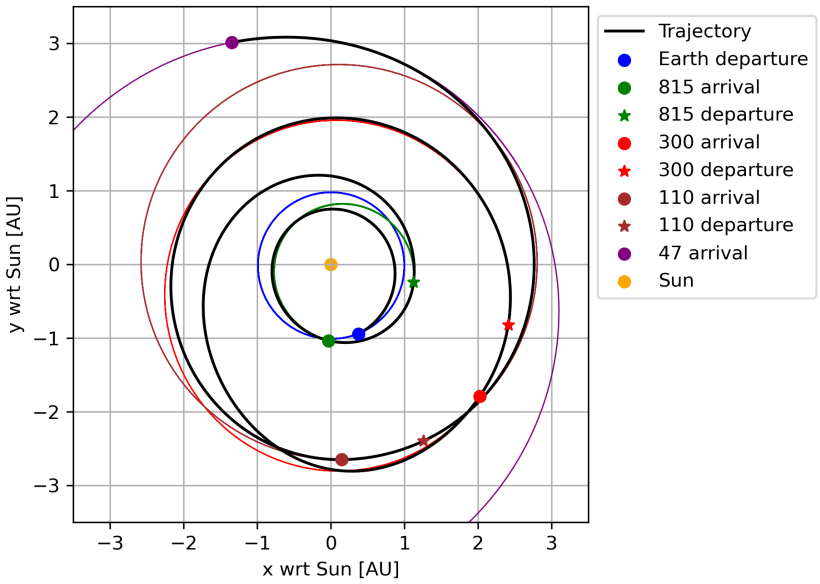


Figure E.20: 2-D view of the optimal GTOC trajectory found after 50 generations by the specific model and the GA.

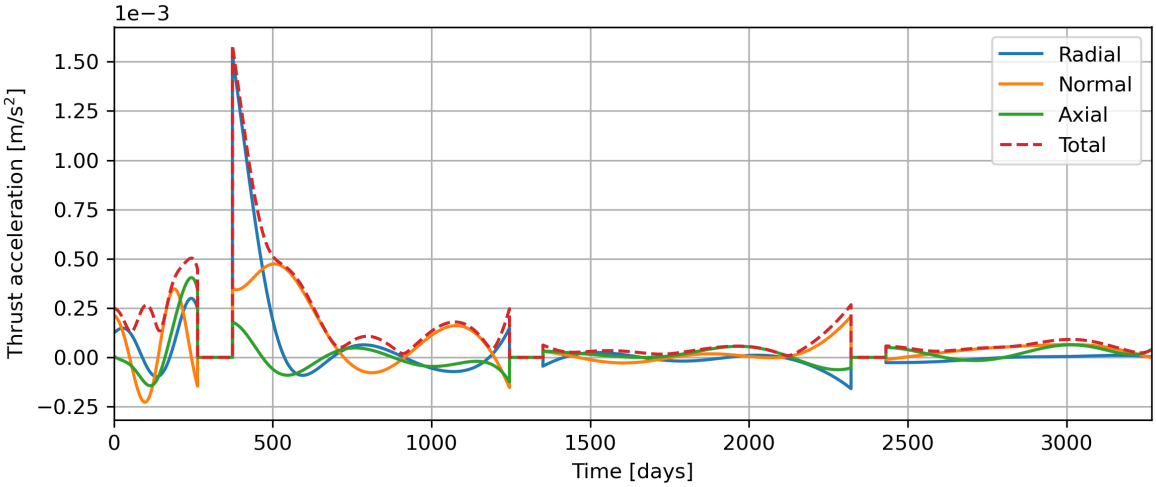


Figure E.21: Thrust profile of the optimal GTOC trajectory found after 50 generations by the specific model and the GA

References

[1] John W Dankanich. “Low-thrust propulsion technologies, mission design, and application”. In: *Aerospace technologies advancements* January (2010), pp. 219–240.

[2] Marc D. Rayman et al. “Results from the Deep Space 1 technology validation mission”. In: *Acta Astronautica* 47.2-9 (2000), pp. 475–487.

[3] CT Russell and CA Raymond. “The dawn mission to Vesta and Ceres”. In: *The dawn mission to minor planets 4 vesta and 1 ceres* (2012), pp. 3–23.

[4] Johannes Benkhoff et al. “BepiColombo—Comprehensive exploration of Mercury: Mission overview and science goals”. In: *Planetary and Space Science* 58.1-2 (2010), pp. 2–20.

[5] David Morante, Manuel Sanjurjo Rivo, and Manuel Soler. “A survey on low-thrust trajectory optimization approaches”. In: *Aerospace* 8.3 (2021), p. 88.

[6] Francesco Topputo and Chen Zhang. “Survey of direct transcription for low-thrust space trajectory optimization with applications”. In: *Abstract and Applied Analysis*. Vol. 2014. Hindawi. 2014.

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

[8] Madhusudan Vijayakumar and Ossama Abdelkhalik. “Shape-Based Approach for Low-Thrust Earth–Moon Trajectories Initial Design”. In: *Journal of Guidance, Control, and Dynamics* 45.1 (2022), pp. 103–120.

[9] Bernd Dachwald. “Optimization of very-low-thrust trajectories using evolutionary neurocontrol”. In: *Acta Astronautica* 57.2-8 (2005), pp. 175–185.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[11] Dario Izzo, Marcus Märtens, and Binfeng Pan. “A survey on artificial intelligence trends in spacecraft guidance dynamics and control”. In: *Astrodynamics* 3.4 (2019), pp. 287–299.

[12] Yaochu Jin. “Surrogate-assisted evolutionary computation: Recent advances and future challenges”. In: *Swarm and Evolutionary Computation* 1.2 (2011), pp. 61–70.

[13] Charles H Acton Jr. “Ancillary data services of NASA’s navigation and ancillary information facility”. In: *Planetary and Space Science* 44.1 (1996), pp. 65–70.

[14] David J Gondelach and Ron Noomen. “Hodographic-shaping method for low-thrust interplanetary trajectory design”. In: *Journal of Spacecraft and Rockets* 52.3 (2015), pp. 728–738.

[15] Jeffrey C Lagarias et al. “Convergence properties of the Nelder–Mead simplex method in low dimensions”. In: *SIAM Journal on optimization* 9.1 (1998), pp. 112–147.

[16] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[17] Michael JD Powell et al. “The BOBYQA algorithm for bound constrained optimization without derivatives”. In: *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge* 26 (2009).

[18] Mona Fuhrländer and Sebastian Schöps. “Hermite least squares optimization: a modification of BOBYQA for optimization with limited derivative information”. In: *Optimization and Engineering* (2023), pp. 1–27.

[19] Giuseppe Ughi, Vinayak Abrol, and Jared Tanner. “A model-based derivative-free approach to black-box adversarial examples: Bobyqa”. In: *arXiv preprint arXiv:2002.10349* (2020).

[20] Francesco Biscani, Dario Izzo, and Chit Hong Yam. “A global optimisation toolbox for massively parallel engineering optimisation”. In: *arXiv preprint arXiv:1004.3824* (2010).

[21] Alexandros Tzanetos, Iztok Fister Jr, and Georgios Dounias. “A comprehensive database of Nature-Inspired Algorithms”. In: *Data in brief* 31 (2020), p. 105792.

[22] Seyedali Mirjalili. “Genetic algorithm”. In: *Evolutionary algorithms and neural networks*. Springer, 2019, pp. 43–55.

[23] Darrell Whitley. “A genetic algorithm tutorial”. In: *Statistics and computing* 4.2 (1994), pp. 65–85.

[24] Leon Stubbig and Kevin Cowan. “Improving the Evolutionary Optimization of Interplanetary Low-Thrust Trajectories Using a Neural Network Surrogate Model”. In: *rendezvous* 2 (2021), t2.

[25] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[26] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, USA, 2015.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[28] Gao Tang, Fanghua Jiang, and Junfeng Li. “Fuel-optimal low-thrust trajectory optimization using indirect method and successive convex programming”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.4 (2018), pp. 2053–2066.

[29] Ehsan Taheri and Ossama Abdelkhalik. “Initial three-dimensional low-thrust trajectory design”. In: *Advances in Space Research* 57.3 (2016), pp. 889–903.

[30] Dirkx Dominic et al. “The open-source astrodynamics Tudatpy software. Overview for planetary mission design and science analysis”. In: *Europlanet Science Congress 2022*. 2022.

[31] Xuanyu Hu et al. “Sensitivity analysis of polar orbiter motion to lunar viscoelastic tidal deformation”. In: *Celestial Mechanics and Dynamical Astronomy* 135.2 (2023), p. 16.

[32] Mark Van der Merwe et al. “Learning continuous 3d reconstructions for geometrically aware grasping”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 11516–11522.

[33] Balakumar Sundaralingam et al. “Robust learning of tactile force estimation through robot interaction”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9035–9042.

[34] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[35] Alec Radford et al. “Robust speech recognition via large-scale weak supervision”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 28492–28518.

[36] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (2023), pp. 1123–1130.

[37] Haiyang Li et al. “Deep networks as approximators of optimal low-thrust and multi-impulse cost in multitarget missions”. In: *Acta Astronautica* 166 (2020), pp. 469–481.