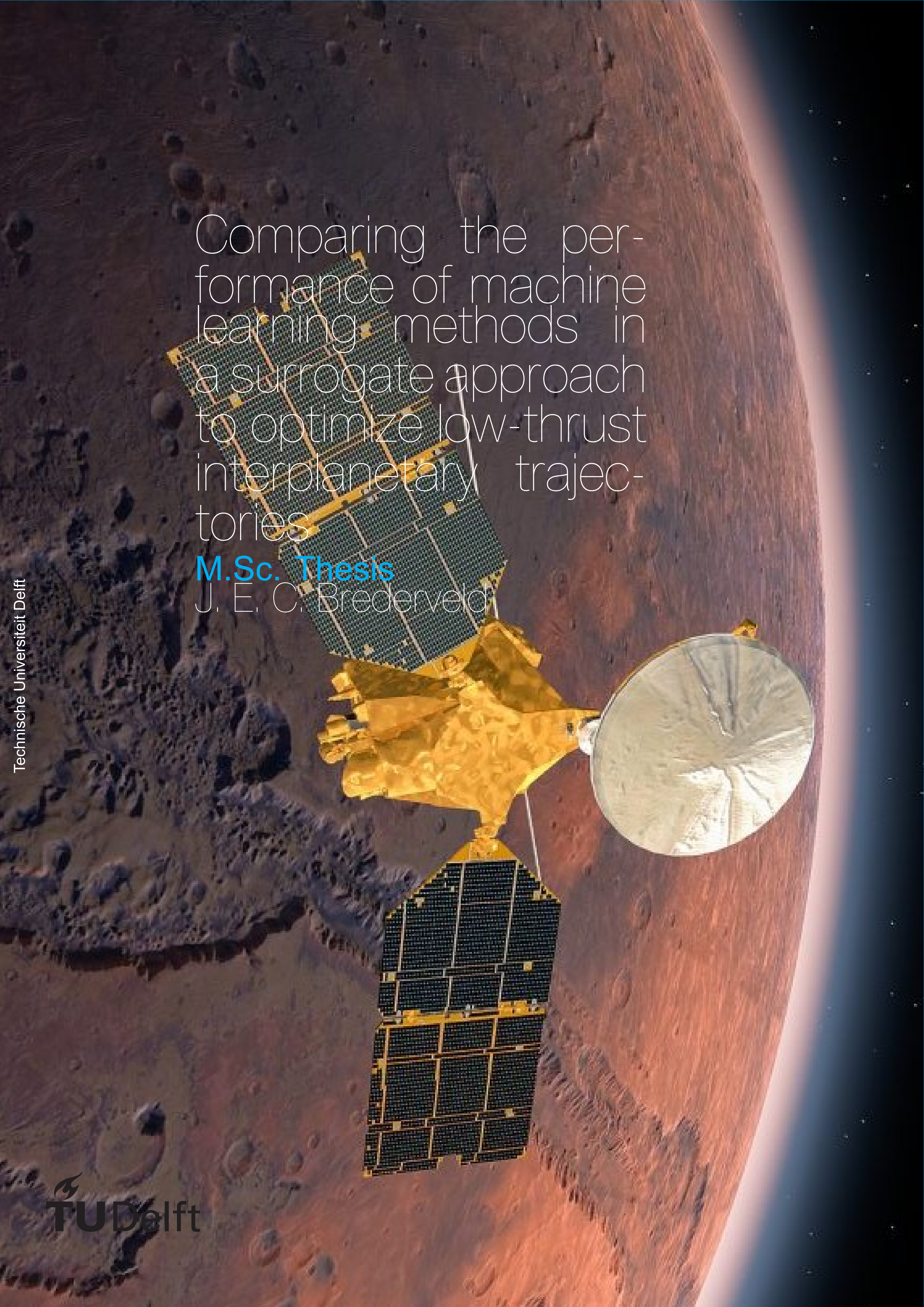


Comparing the performance of machine learning methods in a surrogate approach to optimize low-thrust interplanetary trajectories

M.Sc. Thesis

J. E. C. Brederveld

Technische Universiteit Delft

Comparing the performance of machine learning methods in a surrogate approach to optimize low-thrust interplanetary trajectories

M.Sc. Thesis

by

J. E. C. Brederveld

to obtain the degree of Master of Science
at space department of the faculty Aerospace Engineering,
Delft University of Technology,
to be defended publicly on Monday 25 January 2021 at 13:00.

Student number:	4140346	
Project duration:	September 1, 2020 – January 25, 2021	
Thesis committee:	Prof. Dr. Ir. P. N. A. M. Doe,	Committee chair
	Ir. K. J. Cowan MBA,	Supervisor
	Dr. J. Guo,	External Examinator

Preface

Writing a thesis is challenging in multiple aspects. Not only is the subject tough, staying focus throughout the period is a challenge as well, from the literature study to performing the simulations to writing the paper. The Coronavirus made it even more challenging: study places like the 9th floor master room were closed, no social contact was allowed for comfort, and I have not seen my supervisor in person the last 9 months. However, this also urged me to be creative. To simulate working together, daily hangout sessions were organized with Jelle and Marie, for which I want to thank them so much! If Marie catches up on Dutch easily, it is because of the daily puzzle Jelle and I did during our breaks. Furthermore, I would like to thank everyone who spent hours proofreading my paper. I know most of you were not familiar with the subject yet, but nevertheless, the feedback was really valuable and it made me achieve writing a paper that makes me proud. Furthermore, I would like to thank my supervisor, Kevin Cowan. From the moment I attended your lectures, I knew I wanted you to be my supervisor. Your never-ending enthousiasm and creativity helped me to define this thesis subject myself. Finally, I would like to thank my parents. They have supported me through my whole study period, however long it took.

J. E. C. Brederveld
Delft, November 2020

Abstract

Various machine learning algorithms have been applied to find optimal low-thrust satellite trajectories, however, no fair comparison of their accuracy has been made yet. In this paper, two common and promising supervised machine learning algorithms are compared for their regression capacities: the artificial neural network and the Gaussian process. A grid search is performed to evaluate the performance of the algorithms on different datasets, varying the input features for training, and the hyperparameters of the algorithms. The best performing Gaussian process and artificial neural network are applied as surrogate in a model that optimizes a trajectory from Earth to Mars using a differential evolution optimization strategy. The performance of the models is evaluated on both the Euclidean distance between the inputs corresponding to the predicted minima of the surrogate method and the nearest local minimum obtained with the shaping method, and the accuracy of the predicted ΔV budget required. For both quantities, the Gaussian process outperforms the artificial neural network. The minimum required ΔV budget to reach Mars was predicted more accurately, and the duration of the trajectory and the best moment to launch were found more often and accurately as well.

Contents

Nomenclature

Acronyms and Abbreviations

ANN	Artificial Neural Network
CPU	Central Processing Unit
DE	Differential Evolution
ESA	European Space Agency
Exposin	Exponential Sinusoid
GA	Genetic Algorithm
GP	Gaussian Process
GPR	Gaussian Process Regression
GPU	Graphics Processing Unit
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MJD	Modified Julian Date
ML	Machine Learning
MSE	Mean Squared Error
PP	Piecewise Polynomial
RBF	Radial Basis Function
ReLu	Rectified Linear Unit
St. d.	Standard Deviation
SVM	Support Vector Machine
SVR	Support Vector Regression
TOF	Time of Flight
XGB	Extreme Gradient Boosting

List of Symbols

$\mathbf{a}$	Activated ANN layer output	-
a	Normalized thrust acceleration	-
$\mathbf{b}$	Vector containing biases of the ANN	-
c_i	Coefficient of the velocity function of the hodographic shaping method	-
C_r	Crossover factor for the DE	-
f	Thrust acceleration	m/s^2
F	Mutation factor for the DE	-
k	Kernel function of the GP	-
k_0	Scale parameter for the exposin	m
k_1	Dynamic range parameter for the exposin	-
k_2	Winding parameter for the exposin	-
K	Covariance Matrix	-
L	Loss	-
ℓ	Length scale of the GP	-
N	Number of revolutions around the central body	-
N	Number of hyperparameters of an ANN	-
N_t	Number of training samples	-
N_v	Number of velocity functions	-
NP	Population size of the DE	-
r	Radial distance from the central body	m
S	Class of exposins	-
t_0	Departure date	days in MJD2000
v	Momentum term of the ANN	-
v_i	Velocity base function of the hodographic shaping method	m/s
V	Velocity	m/s
W	Matrix of weights of the ANN	-
$\mathbf{x}'$	Other input in the GP dataset	-
$\mathbf{x}$	Input vector of training/test/validation sample(s)	-
y	Output sample	-
$\mathbf{y}$	Output vector	-
$\mathbf{z}$	ANN layer output	-
r, θ, z	Cylindrical coordinates	[m, rad, m]
x, y, z	Cartesian coordinates	[m, m, m]
α	Thrust angle	rad
γ	Flight path angle	rad
γ_1	Initial flight path angle	rad
δ	Partial derivative of the loss function	-
ΔV	Velocity change	m/s
$\boldsymbol{\theta}$	Set of hyperparameters	-
μ	Mean value	-
$\boldsymbol{\mu}$	Mean function	-
σ	Activation function	-
σ_f	Signal variance	-
σ_n	Variance of the noise	-
φ	Phase angle of the exposin	rad
ψ	Transfer angle	rad

Subscripts

0	Initial value
f	Final value
i	Index
max	Maximum value
min	Minimum value
sc	Scaled
$*$	Previously unseen sample by the ML method

Superscripts

$\square^T$	Transpose of a matrix or vector
$\dot{\square}$	Derivative with respect to time
$\ddot{\square}$	Second derivative with respect to time
$\square'$	Next epoch or generation (for ANN or DE)

1 Introduction

In numerous fields of science, of which the amount is rapidly increasing, machine learning (ML) is being used in research. For the optimization of interplanetary satellite trajectories, multiple methods utilizing ML are devised already [1, 9, 11, 17, 22, 26]. Especially artificial neural networks (ANN) were investigated on multiple occasions, either to find the minimum ΔV budget [1, 22], the optimal final mass of the satellite [11], the minimum time of flight [10] or the optimal control sequence[16, 19]. The two main approaches for ANNs are in a direct approach, to find the control sequence that results in an optimal trajectory, or as surrogate for the fitness function in an optimization algorithm, of which the evolutionary neurocontroller is the first example[3]. Gaussian process regression (GPR) is mainly used for the latter as in [17] for example. Surrogate modelling is most useful for design of low-thrust interplanetary trajectories, since continuous thrust profiles are much harder to optimize than impulsive shots. It is currently unknown what the best ML-algorithm is to find optimal low-thrust trajectories. A broad comparison between multiple ML-algorithms showed that gradient boosting outperformed many other algorithms [11], but the hyperparameters of the ML-algorithms were not optimized since this is a time-consuming process. Performing the comparison to literature is very difficult since not only the type of mission influences the accuracy of the algorithms. The goal for this research is to assess the effectiveness of the ANN and GP in a surrogate model to optimize low-thrust interplanetary satellite trajectories. This is done by training both ML algorithms on various datasets, optimizing the hyperparameters, and performing the optimization approach using the surrogate model. The accuracy of the output is determined by comparing the outcome of the surrogate model to the optimal trajectory in the evaluated range. The Euclidean difference between the estimated input parameters of the ML-algorithm and the input of the optimal trajectory is the metric that determines the performance of both algorithms.

1.1. Research questions

With the objective known, research questions are formulated to reach this objective. The main research question answered in this thesis is:

How does the choice for a machine learning technique affect the prediction of an optimal low-thrust interplanetary satellite trajectory?

Because there are multiple factors influencing the answer of the main research question, these factors are addressed in the following sub-questions:

1. How can machine learning algorithms be used for optimization of low-thrust trajectories?
2. Why is a surrogate method a viable approach?
3. How does the training data influence the accuracy of the output?
4. How can the optimal hyperparameters for the ML-algorithm be determined?
5. How accurate do the ML-algorithms predict the outcome of the test dataset?
6. Why is there a difference between the performance of the algorithms?

7. How do the ML-algorithms compare with each other in terms of computational speed and accuracy?
8. How do the different ML-algorithms perform outside of the training range?

1.2. Thesis structure

The thesis consists of a standalone paper written in the AAS conference format, and additional background on the subjects described in the paper. In Chapter 2, the paper with the main results is presented. In Chapter 3, some additional results to the paper are presented in order to answer the research questions. Chapter 4, a conclusion is drawn by answering the research questions and recommendations for further research are given. Appendix A describes the shaping methods used in the paper. Appendix B describes the different machine learning algorithms used in the paper in more detail.

2 Paper

(Preprint) AAS XX-XXX

COMPARING THE PERFORMANCE OF MACHINE LEARNING METHODS IN A SURROGATE APPROACH TO OPTIMIZE LOW-THRUST INTERPLANETARY TRAJECTORIES

Jan Brederveld,[*] Kevin Cowan[†]

Various machine learning algorithms have been applied to find optimal low-thrust satellite trajectories, however, no fair comparison of their accuracy has been made yet. In this paper, two common and promising supervised machine learning algorithms are compared for their regression capacities: the artificial neural network and the Gaussian process. A grid search is performed to evaluate the performance of the algorithms on different datasets, varying the input features for training, and the hyperparameters of the algorithms. The best performing Gaussian process and artificial neural network are applied as surrogate in a model that optimizes a trajectory from Earth to Mars using a differential evolution optimization strategy. The performance of the models is evaluated on both the Euclidean distance between the inputs corresponding to the predicted minima of the surrogate method and the nearest local minimum obtained with the shaping method, and the accuracy of the predicted ΔV budget required. For both quantities, the Gaussian process outperforms the artificial neural network. The minimum required ΔV budget to reach Mars was predicted more accurately, and the duration of the trajectory and the best moment to launch were found more often and accurately as well.

INTRODUCTION

The design of low-thrust interplanetary satellite orbits is a challenging and time-consuming task. Expert knowledge is required to obtain results close to the global optimum, since the convergence for traditional optimizers is strongly dependent on the initial guess.[1] While traditional trajectories, modelled using Hohmann transfers, are initialized with an instantaneous rocket burst, low-thrust trajectories are designed for engines delivering continuous low thrust over (part of) the trajectory. One of the advantages of low-thrust trajectories is that they allow for a larger ΔV budget than trajectories with instantaneous burns, because the propellant systems are usually more efficient.[2] Missions like BepiColombo and DAWN, being the first mission to Mercury and the first mission orbiting two different solar circling target bodies (Ceres and Vesta) respectively, make use of the extended mission capacities of low-thrust trajectories. Designing these trajectories is a lot harder than for traditional trajectories, since the change in velocity can no longer be modelled in a discrete manner. There is a relation between the initial state of the satellite and the optimal trajectory to reach the destination, but identifying this relation is usually very difficult. With the advanced regression capabilities of various machine learning (ML) techniques, this relation can be made explicit and the optimization process can be accelerated.[3]

[*]MSc. Student, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, janbrederveld@gmail.com

[†]Education Fellow + Lecturer, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, k.j.cowan@tudelft.nl

Among different ML-algorithms, artificial neural networks (ANN)[3–5] and Gaussian processes (GP)[6–8] were shown to be very promising for inference. Both ANNs and GPs can replace the time-consuming propagation step in common optimization algorithms by being trained to approximate the fitness function. In this configuration, the ML-algorithm is a surrogate for the fitness function in the optimization process.[9] There are two important advantages of this approach: evaluating the surrogate is less time-consuming than evaluating the numerical fitness of the candidate solution, and the roughness of the fitness function is smoothed out, which makes it easier to find (locally) optimal solutions.[3] This approach comes, however, at the cost of the accuracy of the fitness function evaluation.

A fair comparison between the different ML-algorithms has not been made for low-thrust trajectory design yet; Most comparisons of ML-algorithms are performed without optimizing the hyperparameters, the parameters that define the dimensions and properties of the ML-algorithm, since this is a time-consuming task,[10] or do not take GPs into consideration.[11] The goal of this paper is to evaluate the effectiveness of both the ANN and the GP in the optimization process and find the set of hyperparameters that provides the best results. In order to make this comparison, multiple datasets of trajectories to Mars are generated. Both the ML-algorithms are trained on this data with varying input parameters and hyperparameters. Based on the comparison made in this paper, the best performing ML-algorithm is determined, together with the optimal hyperparameters and the influencing factors on the accuracy of the ML-techniques.

Firstly, the development of the datasets of trajectories for the use case of an Earth-Mars transfer is described and the two different ML-techniques are introduced. Secondly, the approach of training the ML-algorithms is described, and a large grid search is performed over the different datasets, hyperparameters of the ML-algorithms, training feature sets, and scaling methods. Finally, the best performing ANN and GP are tested in a surrogate approach utilizing a differential evolution algorithm for optimization, the outcome of which determines the most accurate ML-algorithm for this problem.

TRAJECTORY GENERATION

Two different shaping methods are used to generate the trajectory datasets, the exposin shaping method and the hodographic shaping method. Shaping methods assume the trajectory to have a pre-selected shape, and fit the parameters of the shaping method such that the trajectory satisfies the mission with a feasible thrust profile.[12] These datasets are used to investigate the performance of different features as inputs for the machine learning technique.

The first shaping method used in this paper is the exposin shaping method. This method was developed by Petropoulos and Longuski[12] and extended by Izzo with the application of Lambert targeting.[13] The exponential sinusoid used for shaping the trajectory, assuming tangential thrust, is defined by the following expression:

$$r = k_0 \exp(k_1 \sin(k_2\theta + \varphi)), \tag{1}$$

where θ is the polar angle and r the distance to the central body. k_1 is the dynamic range parameter which defines the ratio between apoapsis and periapsis distance. k_2 is the winding parameter, which defines how many rotations about the barycenter are finished before the periapsis is reached. φ defines the orientation of the exponential sinusoid and k_0 is the scaling factor.[12] Petropoulos and Longuski used the exposin shape and the assumption of tangential thrust to define the flight path angle, time of flight (TOF), and accelerations over the trajectory. Izzo showed that when θ is set as $\bar{\theta} = \psi + 2N\pi$:

$$\forall k_2 \; \exists \; S_{k_2} \left[r_1, r_2, \psi, N \right], \tag{2}$$

which is a class of exposins passing through both r_1 and r_2, the distances of the two bodies with respect to the central body, with a phase difference of $\psi + 2N\pi$. N is the number of rotations around the center body. This means that low-thrust trajectories designed using the exposin shaping method can either be defined using the full shape information, [r_0, $\bar{\theta}$, k_0, k_1, k_2, φ], or alternatively using the departure date, t_0, TOF, k_2, and N.[13] In order to generate a large number of trajectories from Earth to Mars, Bouwman implemented a 3D approach of the exposin shaping method. With this dataset, Bouwman showed both sets of parameters can be used for training a GP. The most accurate results were obtained using the second set of input parameters.[8] The same sets of parameters, or features in the field of ML, are used in this paper to perform the comparison between ANNs and GPs.

The other shaping method used is the hodographic shaping method, developed by Gondelach and Noomen.[14] The idea behind this method is that a trajectory can be modelled as the connection between the hodographs of the departure body and the arrival body respectively. By applying the right boundary conditions to the problem, the trajectory is suitable for the mission. The trajectory can be described by the velocity functions. These functions describe the velocity, v, over the trajectory in the normal (θ), radial (r), and axial (z) direction as a function of time, and are multiplied by a coefficient, c. Together they give the total velocity, V:

$$V(t) = \sum_{i=1}^{N_v} c_i v_i(t), \tag{3}$$

The required number of velocity functions equals the number of boundary conditions. For this problem there are three boundary conditions per direction: the initial velocity, the final velocity and the relative displacement. From the velocity, the thrust profile can be determined, which gives the ΔV budget required to arrive at the target body.

MACHINE LEARNING

Examples of ML-algorithms that were used before to perform regression analysis are ANNs, GPs, support vector machines, decision trees, and gradient boosting.[15] Among these, ANNs and GPs have shown to be the most promising. Both Gaussian process regression (GPR) and regression using ANNs are examples of supervised learning methods. The algorithms are trained on a dataset of inputs, $\mathbf{x}$, for which the outputs, $\mathbf{y}$, are known. In order to perform machine learning, the dataset is usually split up in two or three datasets, a training dataset, a test dataset, and for ANNs, a validation dataset as well. Training involves designing the weights and biases of the matrices in the algorithms such that the correct output is given for the input of the training dataset. To test the accuracy of the algorithms, the test dataset of samples that were not used for training is evaluated. The accuracy of the predictions based on the input of the test dataset in comparison to the output of the test dataset defines the accuracy of the ML-algorithm. The validation dataset prevents overtraining for ANNs, which is described in further detail later in this section.

Gaussian Processes

Even though both ML-algorithms are suitable for regression analysis, ANNs and GPs are very different. The GP is a non-parametric model which always perfectly matches the training dataset.

The behavior in between points can be modelled as the correlation between points using a multivariate Gaussian distribution. This behavior is mainly determined by the kernel or prior used for the GP, and the hyperparameters, $\boldsymbol{\theta}$: the length scale, ℓ, signal variance, σ_f, and the variance of the noise, σ_n. Since the trajectories are generated using a deterministic model, there is no noisy data so $\sigma_n = 0$. The GP is defined as follows:[16]

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{4}$$

where μ is the mean function, k is the kernel function, and $\mathbf{x}$ and $\mathbf{x}'$ are two different training samples. The kernel is a function of the training data and the hyperparameters. A common kernel is the squared exponential or radial basis function (RBF) kernel (Eq. (5)):

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{(\mathbf{x}-\mathbf{x}')^2}{2\ell^2}\right). \tag{5}$$

The GP is trained to estimate the mean and the standard deviation of the output of the test samples, $\mathbf{y}_*$:

$$\mathcal{GP}(\mathbf{x}_*) = p(y_*|X, \mathbf{y}, \mathbf{x}_*), \tag{6}$$

where X is the matrix containing all training input.

The optimal hyperparameters have been selected when the best a posteriori estimate occurs, i.e. at the maximum of $p(\boldsymbol{\theta}|X, \mathbf{y})$. Using Bayes' theorem and the marginal likelihood, it can be shown that this is equal to maximizing $\log p(\mathbf{y}|X, \boldsymbol{\theta})$, which is

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi, \tag{7}$$

in which K is the covariance matrix, given by

$$K = \begin{bmatrix} k(\mathbf{x_1}, \mathbf{x_1}) & k(\mathbf{x_1}, \mathbf{x_2}) & \cdots & k(\mathbf{x_1}, \mathbf{x}') \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}', \mathbf{x_1}) & k(\mathbf{x}', \mathbf{x_2}) & \cdots & k(\mathbf{x}', \mathbf{x}') \end{bmatrix}. \tag{8}$$

The maximum value for Eq.(7) can be determined using an optimization algorithm.

Artificial Neural Networks

The ANN tends to find a best-fitting approximation of the data by performing multiple matrix multiplications in the different layers. A layer consists of a matrix of weights, W, and a vector of biases, $\boldsymbol{b}$. The input vector is multiplied with the weights, the biases are added, and non-linearity is introduced using an activation function such as the rectified linear unit (ReLu) function, Eq. (10). The activated outcome, $\boldsymbol{a}$, is the input for the subsequent layer. The following steps are taken in the feedforward stage:

$$\boldsymbol{z} = W\boldsymbol{x} + \boldsymbol{b} \tag{9}$$

and

$$\boldsymbol{a} = ReLu(\boldsymbol{z}) = max(\boldsymbol{0}, \boldsymbol{z}). \tag{10}$$

The outcome of the final layer is compared to the training output and evaluated using a loss function. The loss functions used in this research are the mean absolute error (MAE) and mean absolute percentage error (MAPE), since they were used in comparable research.[8,17] The MAE and MAPE are given by

$$MAE = \frac{1}{N_t}\sum_{i=1}^{N_t}\left|y_{test_i} - y_{pred_i}\right| \quad , \quad MAPE = \frac{100\%}{N_t}\sum_{i=1}^{N_t}\left|\frac{y_{test_i} - y_{pred_i}}{y_{test_i}}\right| . \tag{11}$$

These loss functions are applicable to GPs as well. The objective, to fit to the training data as good as possible, is achieved by minimizing the loss function for the training dataset. The weights and biases of the different layers are updated based on their derivative with respect to the loss function.[18] After all training samples, N_t, are used, the validation dataset is evaluated, and the next epoch of training starts.

An issue with ANNs is overtraining. This implies that if the ANN is trained on the training dataset for too long, it could pick up on patterns that are only present in the training dataset. When these are not present in the data the training samples are taken from, the accuracy of the algorithm in predicting the test data might decrease with an increased number of training epochs. In order to detect overtraining a third dataset is generated, which is not used for training, but is evaluated after every epoch. This is the validation dataset. When overtraining occurs, the prediction accuracy of the training dataset improves over epochs, but the improvement stops for the validation dataset, and the performance might even decline. There are measures to prevent overtraining, such as early stopping, regularization or dropout, but these are not applied in this research since overtraining is taking part slightly for some networks, but is not forming an issue in this specific research.

METHOD FOR IMPLEMENTATION AND ANALYSIS

Machine learning consists of two important aspects, the ML-algorithm and the data used for training. Since they are independent from another, almost the same pipeline can be used for different machine learning algorithms. The approach followed in this research is shown in Figure 1.

At first, a dataset of low thrust trajectories is generated. From this dataset, a subset is generated by selecting the features the ML-algorithm will be trained on. This dataset is split into two or three parts, depending on the ML-algorithm. The data are scaled and the ML-algorithm is prepared for training. The hyperparameters for the algorithm or the kernels are chosen, the ML-algorithm is fitted on the training data, and if necessary evaluated using the validation dataset. The accuracy of the ML-algorithm is determined by comparing the outcome of the ML-algorithm with the output of the test dataset. The most accurate ML-algorithm is used in a surrogate model,[9] where the ML-algorithm is evaluated instead of the objective function. In order to find the best-performing dataset, feature set, and set of hyperparameters, a grid search is performed over the inputs depicted by the six blue diamonds in Figure 1. The steps performed to go from the inputs to a ML-algorithm that is suitable for use in a surrogate approach is described in the two sections below.

Generating the data

The input parameters selected in the first step of Figure 1 define the low-thrust trajectories the ML-algorithms are trained on. These are the type of mission, the trajectory design method, and the ranges of departure dates and travel times. Furthermore, the number of revolutions around the Sun, N, before reaching the target and the shape parameters were varied. Three different trajectory datasets are generated, of which the selected input parameters can be found in Table 1. A rendezvous mission to Mars is chosen to make verification of the minimum of the dataset possible.[8,14]

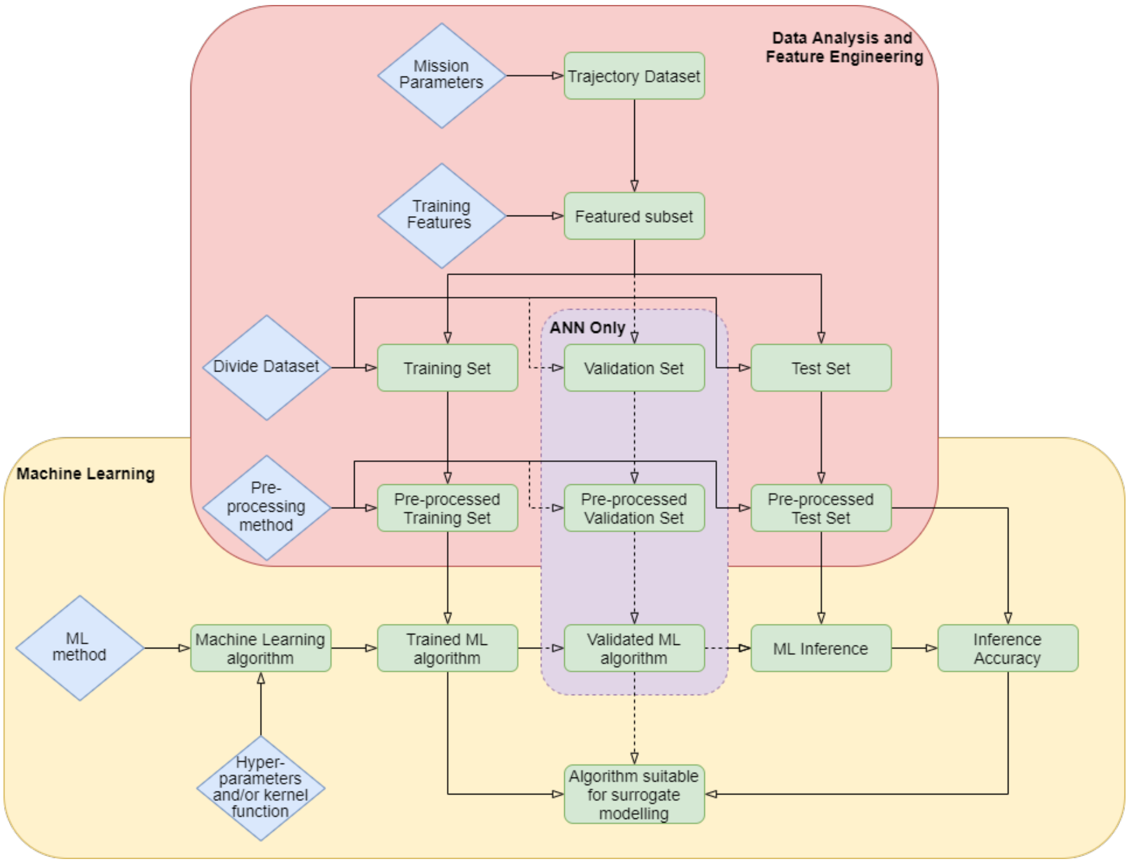


Figure 1. The pipeline of the algorithm used for the grid search. The red area contains the data analysis and feature engineering, the yellow area contains the machine learning. The purple area and dashed arrows are specific for ANNs.

Table 1. The low-thrust trajectory datasets used for training the different ML-algorithms.

Mission parameters	Dataset 1	Dataset 2	Dataset 3
Mission	Mars Rendezvouz		
Departure dates [MJD2000]	7671-10227		
Departure dates	01-01-2021 to 31-12-2028		
Mission Duration [days]	500-2000		
Shaping Method	Exposin	Hodographic	Hodographic
N	1	1	0-5
Timestep [days]	1	10	40
Number of trajectories	233814	38656	2432

Both shaping methods are similar in that the time required to generate a trajectory is on the order of seconds.[8,17] The exact duration depends on the computational power used to generate the trajectories, but in general the hodographic shaping method takes a bit longer to generate a trajectory, because of the free parameters that need to be optimized.

In the second step, the features selected to train on are chosen. The exposin and hodographic shaping methods have very different features to describe the trajectory. Bouwman showed a GP can successfully be trained on two different sets of features of the exposin shaping method,[8] so the same two feature sets are chosen for this research. The first set consists of three features: t_0, TOF, and k_2. The second set consists of the full shape information of the exposin shaping method. These features are r_0, ψ k_0, k_1, k_2, and φ as defined in Eq. (1). Both these sets of features were mapped to the required ΔV to travel the specified trajectory.

The feature sets selected for the hodographic shaping method consist of the features given in Table 2. Stubbig showed that ANNs can successfully be trained on the first set,[17] albeit with other input parameters for the dataset of trajectories. The second and third feature set are selected to show the differences between training on cylindrical and Cartesian coordinates. The last set is very simple and practical for an optimization approach since it is continuous.

Table 2. The feature sets utilized for training the ML-algorithm on the hodographic datasets (Dataset 2 and 3).

Feature set	Features	Number of features
1	Full cylindrical state information at departure and arrival, departure date and TOF	14
2	Position difference between Earth and Mars in cylindrical coordinates and TOF	4
3	Position difference between Earth and Mars in Cartesian coordinates and TOF	4
4	The departure date and the TOF	2

The third step of preparing the data before training the algorithm is to divide the data over a training set, a test set, and in case an ANN is to be trained, a validation set. The sizes of these sets depend on the dataset, as well as on the ML-algorithm. Training the GP involves inverting a square matrix with the size of the number of training samples, n. Therefore increasing the number of training samples increases the training time with $\mathcal{O}(n^3)$.[19] The shape of the ANN is not dependent on the amount of training data; increasing the amount of training data leads to a linear increase in the required training time. To generate the training set, random samples are chosen from the full set of trajectories. The validation dataset is picked randomly from the data left, and the test dataset is picked last.

The last step of preparing the data before training the ML-algorithm is scaling the data. It is known that scaling the data does improve the performance of non-decision based ML-algorithms like the ANN and GP.[20,21] It prevents the input variable with the largest range to be dominant in determining the trend.[22] Moreover, the optimization algorithm within the ANN converges faster when the input variables are scaled equally.[23] Nawi et al. showed that pre-processing can improve the performance of ANNs on classification problems[*] by up to 95%.[23] For scaling the data, three approaches were selected. The first approach is to not scale the data, in order to create a benchmark. The second approach is standardization or Z-score normalization (Eq. (12)). The last approach is to normalize the data (Eq. (13)). Standardizing the data gives the data a mean of 0, and a standard deviation of 1 by subtracting the mean and dividing by the standard deviation. Normalization scales the data between -1 and 1. The two equations for the scaling methods are:

$$x_{sc} = \frac{x - \bar{x}}{\sigma(x)} \tag{12}$$

and

$$x_{sc} = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1. \tag{13}$$

[*]Classification is performed with a discrete implementation of ANNs, regression with the continuous implementation.

The benefit of normalization is that the domain for the optimization approach is known. The downside however, is that this scaling method is very sensitive to outliers in the data. For standardization, it is the other way around. The scaling parameters are defined using the training dataset, because only the training samples are considered known for the ML-algorithm.

Hyperparameters for grid search

Both the GP and the ANN have a set of hyperparameters that require optimization. The hyperparameters of the GP are optimized by maximizing the log-likelihood of the hyperparameters for the training samples (Eq. (7)), so the factors that influence the accuracy most are the kernel and the number of training samples. In this paper, 17 different kernels are tested. The kernels used are (every different combination of) the RBF kernel, the twice differentiable Matérn kernel, and the piecewise polynomial (PP) kernel of order two. These kernels are selected because they are common for GPR (RBF and Matérn kernels), or were shown to provide good results on trajectory optimization problems (PP kernel[8]). The number of training samples varied per dataset. For dataset 1 and 2, the number of training samples varied between 1000, 2000, or 3000, and for dataset 3, the GP was either trained on 800, 1200, or 1632 training samples. The activation function is set to be the ReLu function, and the optimizer of the ANN is set to be Adam with Nesterov moment, because they have shown to be the most effective.[24]

For the ANN, there are many more hyperparameters which can be varied. These are determined using a grid search over the hyperparameters. Among methods for discovering the optimal hyperparameters,[25] grid search, random search and a tree-structured Parzen estimators approach have been used for similar problems.[11,26] The benefit of a grid search is that not only a set of good hyperparameters is found, the effect of changing them can be investigated as well. The selected ranges for the grid search are given in Table 3.

Table 3. Selected ranges for the hyperparameters of the ANNs

Hyperparameter	Dataset 1	Dataset 2	Dataset 3
Number of hidden layers	0-3	0-3	0-3
Nodes per layer	16, 32, 64, 128	16, 32, 64, 128	16, 32, 64, 128
Activation function	ReLu	ReLu	ReLu
Loss function	MAPE	MAPE	MAE
Optimizer	Nadam	Nadam	Nadam
Batch size	32, 128, 512, 2048	32, 128, 512, 2048	32, 64, 128
Number of epochs	100, 300, 500	100, 300, 500	100, 300, 500
Training set size	2000 or 20000	20000	800, 1200, 1632
Validation set size	1000	1000	200
Test set size	1000	1000	600

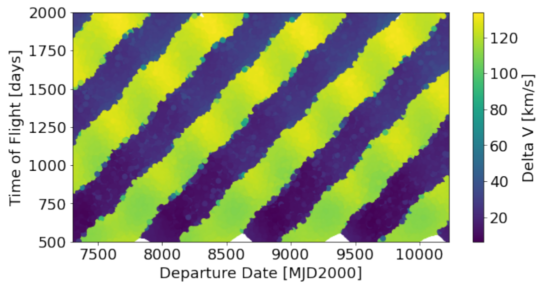
An ANN without hidden layers consists of two layers, an input and an output layer. Networks with three hidden layers do have five layers in total. In order to decrease the time required to perform the grid search, the runs in which is iterated over 300 and 500 epochs are a continuation of the runs over 100 or 300 epochs respectively. This approach decreased the time required to perform the grid search by about 45%.

RESULTS

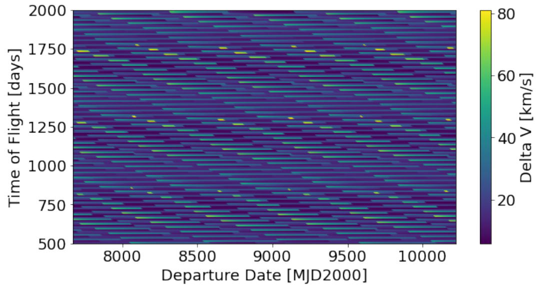
In this section, a selection of the results of the grid search is presented. Some of the results of the grid search are very well known or have been described in other research.[27] When these results are in line with previous research, they are not shown again.

Input datasets

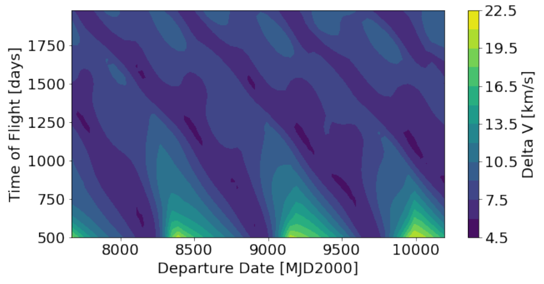
To find the source of the different outcomes for the ML-algorithms, it is important to first analyze the data the algorithms were trained on. To show the difference between the datasets, the porkchop plots of the trajectories are given in Figure 2.



(a) Porkchop plot of dataset 1, generated with the Exposin shaping method.



(b) Porkchop plot of dataset 2, generated with the hodographic shaping method.



(c) Porkchop plot of dataset 3, generated with the hodographic shaping method.

Figure 2. Porkchop plots of the different datasets in this paper.

As can be seen in Figure 2, the different datasets all display different characteristics, depending on the techniques used and the input parameters given in Table 1. These characteristics largely influence the performance of the ML-algorithms trained on these datasets, as will become clear in this section. Generating trajectories using the exposin shaping method (Dataset 1, Figure 2(a)) results in a dataset that has alternating regions of high and low ΔV values, and only a few intermediate values. The first hodographic shaped trajectory dataset (Dataset 2, Figure 2(b)) consists of trajectories with ΔV values divided more equally over the range, however, there are no large clusters of trajectories with similar ΔV values. The last dataset (Dataset 3, Figure 2(c)) shows the ΔV values ranging rather equally over the whole range, and clear regions of higher and lower ΔV values. Even though the ranges and distributions of the output of the different methods vary a lot, the ΔV budgets required to perform the optimal trajectory are within 0.5 km/s from each other, with 6.26 km/s for the first dataset, 5.83 km/s for the second and 5.78 km/s for the last. The standard deviations of the datasets are 49.9, 12.16 and 2.17 km/s. The standard deviations of the datasets are an important indication of what accuracy can be obtained when performing regression analysis.

As can be seen by comparing the plots in Figure 2, the method of generating the trajectories is not the most important property that defines what the plots look like. The smoothness is mainly determined by the number of revolutions around the Sun that are allowed, and the effect of this property is evaluated in this paper. The purpose of using different trajectory design methods is mainly to compare the performance of different feature sets that can be used for training the ML-algorithms, but also to be able to make a comparison with other literature.

Results of the grid search

Subsequently, the grid search is used to find the hyperparameters of the ANN that yield the best performance, and the kernels that provide the best performing GPs. The metric of the performance of the algorithms is the loss function (Eq.(11)). The goal of the ML algorithm, to train on the training data, is achieved at the minimum of the loss function. Another metric used in this paper is the difference between the test output, $\mathbf{y}$, and the output of the algorithm, called prediction error in the rest of this paper. This metric is important because it has a meaningful value (ΔV) which makes comparing different algorithms and different datasets possible.

The influence of some of the hyperparameters on the loss function is known very well, so the effect of changing them is not shown in this paper as long as the effect is supported by the data. A summary of the effects is given in Table 4. The main trade-off for these hyperparameters is accuracy versus time. Balancing these hyperparameters is key, since there is a limit in the improvement of accuracy that can be achieved by tuning them. Increasing the number of layers, or number of nodes per layer improves the accuracy of smaller models, but it introduces too many free parameters at a certain point. This increases the risk of overtraining, and increases the computational effort as well.[4]

Table 4. Effect of increasing different hyperparameters for the ANN.[17]

Trend for hyperparameter	Accuracy	Computational effort
Increase number of layers	Increases	Increases
Increase number of nodes per layer	Increases	Increases
Increase batch size	Decreases	Decreases
Increase number of epochs	Increases	Increases

The inputs and hyperparameters of the ML-approach that are investigated further in this paper are the trajectory datasets, the scaling methods, the feature sets, the kernels used for the GP and the number of training samples.

Differences between datasets Supported by the data in Table 5, one of the first observations is that the average prediction error for the different trajectory datasets is very much dependent on the input dataset itself.

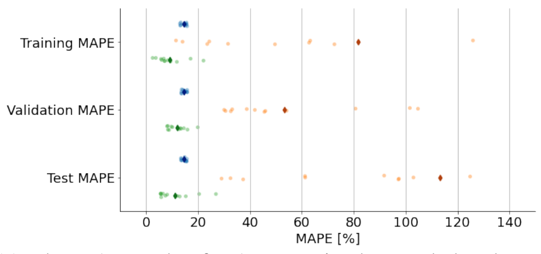
For the first and second dataset, it was required to filter results obtained by the ANN's before comparing. Due to the initialization of the weights in the different layers, the initial value of the MAPE loss function is about 100. Since the objective of the ML-algorithm is to decrease the training loss, runs where the training loss exceeded 200 were deemed incorrect, and therefore disregarded. There is another class of results that is erroneous, but not filtered. When the standard deviation of the prediction error is larger than the standard deviation of the dataset itself, the accuracy of the

Table 5. Standard deviation of the prediction error of the ANNs and GPs for the trajectory datasets.

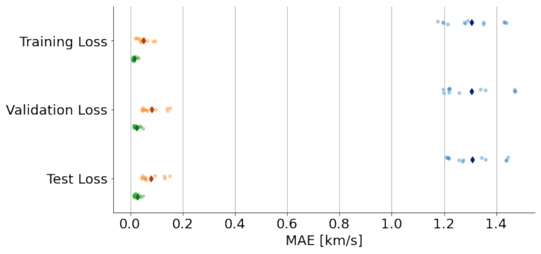
	ANN			GP		
	Dataset 1	**Dataset 2**	**Dataset 3**	**Dataset 1**	**Dataset 2**	**Dataset 3**
Mean [km/s]	41.40	12.86	0.83	43.82	12.13	0.69
St. d. [km/s]	35.08	9.55	0.43	18.01	0.93	0.68
Min [km/s]	12.09	11.47	0.16	12.34	10.60	0.07
Max [km/s]	550.6	227.2	1.81	255.1	35.80	2.25

outcome of the trained ML-algorithm is less accurate than using the mean of the dataset as prediction for every sample. These data are not filtered since they are considered an unsatisfying result, but not an error in the algorithm. The reason the algorithm does not converge to the minimum loss function is most likely related to the peakiness of the dataset, which increases the chances of the optimizer of the algorithms to be stuck in a local minimum. The results in Table 5 show that it is easier to find a configuration for which the ANN is capable of accurately predicting the required ΔV budget for dataset 3, the least peaky dataset, since the grid search resulted in smaller standard deviations of the prediction error, with about a factor $10 - 100$ for both the mean and minimum values.

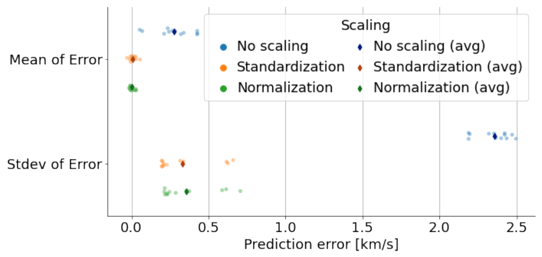
Scaling The effect of scaling is tested on dataset 3. To show the effects of scaling, the ML-algorithms were trained on non-scaled data, the normalized data and the standardized data. In Figure 3, the results from training the ANNs are shown using circles, with the average depicted by a diamond in the same color. Both the MAPE as the MAE are considered in Figure 3, since an important conclusion can be drawn from comparing both loss functions.



(a) The MAPE value for ANNs trained on scaled and non-scaled data of dataset 3.



(b) The MAE value for ANNs trained on scaled and non-scaled data of dataset 3.



(c) The standard deviation of the prediction for ANNs trained on scaled and non-scaled data of dataset 3.

Figure 3. Comparison of the performance of ANNs trained on dataset 3 for different scaling techniques. The same legend applies to all plots.

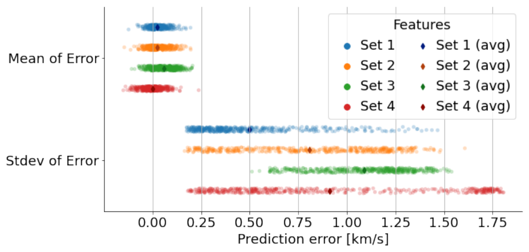
Based on the MAPE in Figure 3(a), it might look like using normalized data improves the accuracy of the ANN slightly over non-scaled data, and standardization decreases the accuracy significant. However, this does not follow from the mean and the standard deviation of the prediction error in Figure 3(c). In order to find out whether scaling improves the performance or not, verification is performed using the MAE over the same training-, validation- and test-datasets. In Figure 3(b), it is shown that the lowest loss function is obtained after normalizing the data, but performing standardization also results in a lower loss than not scaling the data. The reason for the difference between the loss functions and the standard deviation of the error after scaling is that the loss function is a metric of the ML-algorithm, while the standard deviation of the error is a metric of the output data. When standardizing the data, the MAPE is expected to go up since the divisor in Eq. (11) is close to zero. For the MAE, the loss is expected to decrease the most after normalization, since the difference between the test data and the expected output scales with the data (Eq. (11)).

Concerning the performance of scaling, it can be concluded from Figure 3(c) that scaling the data does improve the performance of the ML-algorithms on the test dataset. For the non-scaled data, the standard deviation of the prediction error approximately equals the standard deviation of dataset 3. I.e. it is expected that the outcome of the ANN equals the mean of the training dataset for all test samples. This implies that, without scaling, no meaningful training took place on this dataset. An F-test showed that the decrease of the standard deviation of the error after scaling is statistically significant, but the difference between standardization and normalization is not. Singh[22] states that standardization has a slight edge over normalization, since the effectiveness is less dependent on the input data of the problem. For GPR, similar effects are observed, as can be seen in Table 6. The MAPE for the standardized data is orders of magnitude higher than for the non-scaled data and normalized data, but the standard deviation of the prediction error is slightly lower after scaling the data. The most important difference is that for the non-scaled data, none of the ANNs were trained successfully, while the standard deviation of the prediction error for some of the GPs decreased to below 0.3km/s.

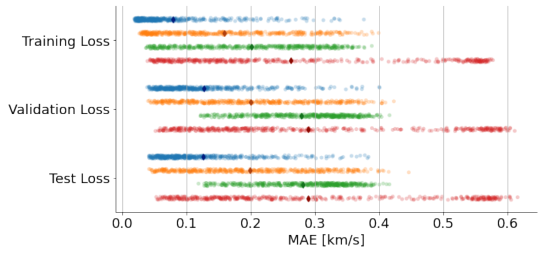
Table 6. Summary of the results of training the ANNs, shown in Figure 3, and results of training GPs on the same dataset.

	ANN		**GPR**	
No scaling	**Minimum**	**Mean**	**Minimum**	**Mean**
MAPE [%]	13.43	14.65	1.780	45.30
MAE [km/s]	1.21	1.31	0.159	3.845
St. d. of the prediction error [km/s]	2.19	2.36	0.289	1.811
Standardization				
MAPE [%]	29.07	113.1	17.27	161.5
MAE [km/s]	0.044	0.079	0.037	0.382
St. d. of the prediction error [km/s]	0.196	0.333	0.073	0.638
Normalization				
MAPE [%]	5.593	11.04	7.176	181.6
MAE [km/s]	0.014	0.025	0.033	0.617
St. d. of the prediction error [km/s]	0.213	0.357	0.057	0.680

Features The features selected for training the ML-algorithm are essential for the performance. This is shown best by comparing the performance of the different ML-algorithms on dataset 1 and 3. The feature sets for dataset 1 are the full exposin shape information and the reduced set developed by Izzo. The features trained on for dataset 3 are given in Table 2. For this analysis, only the standardized data are evaluated in order to make the datasets as comparable as possible. Figure 4 and Table 7 show the results of training the ANNs and GPs on different features within dataset 3.



(a) The mean and the standard deviation of the prediction error for the ANNs trained on the different feature sets.



(b) The training loss, validation loss and test loss for the ANNs trained on the different feature sets. The selected loss function is the MAE.

Figure 4. Comparison of the performance of ANNs on different feature sets of dataset 3. The feature sets in these plots correspond to the feature sets in Table 2.

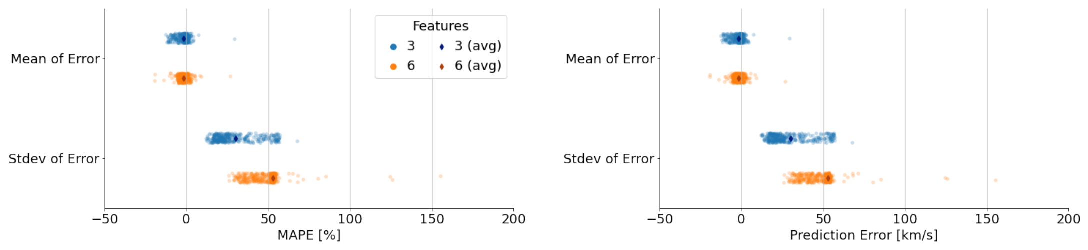
In Figure 4, it can be seen that the best performance is obtained using feature set 1, with 14 input parameters (2 times 6 state variables, departure date and TOF). The difference between the feature sets is small, but statistically significant. Furthermore, feature set 3, containing the difference in state variables in Cartesian coordinates performs worse than the other three in terms of standard deviation of the error, validation loss and test loss. This means that the difference in Cartesian coordinates is not a straightforward method of describing a trajectory. The reference frame lacks patterns that are present in other coordinate frames (such as the difference in radius, which is almost constant for trajectories from Earth to Mars). These patterns are very important to define the trend. In the training loss, this difference is not visible, which implies overtraining took place. There are measures to prevent overtraining, however, since the other feature sets perform better, it is recommended to continue research on one of the other feature sets.

The same feature sets are preferred when training GPs, according to Table 7. Using the full state information as input provides the most accurate results, both in terms of the loss function and the standard deviation of the prediction error.

Table 7. Summary of the results of training the ANNs and GPs on different feature sets for dataset 3. The results for the ANNs are also shown in Figure 4.

	ANN		**GPR**	
Set 1, Full state information	**Minimum**	**Mean**	**Minimum**	**Mean**
MAE [km/s]	0.042	0.127	0.036	0.378
St. d. of the prediction error [km/s]	0.175	0.496	0.071	0.593
Set 2, Cylindrical coordinates				
MAE [km/s]	0.042	0.199	0.122	0.290
St. d. of the prediction error [km/s]	0.165	0.807	0.238	0.525
Set 3, Cartesian coordinates				
MAE [km/s]	0.119	0.281	0.250	0.552
St. d. of the prediction error [km/s]	0.511	1.088	0.528	0.971
Set 4, TOF and Departure date				
MAE [km/s]	0.053	0.289	0.068	0.432
St. d. of the prediction error [km/s]	0.181	0.911	0.127	0.661

Bouwman showed that training on a larger number of input parameters does not always improve the accuracy of GPR.[8] It was shown that the GPs trained on three input parameters outperformed the GPs trained on the full shape information. Using the same dataset, generated using the exposin shaping method (Dataset 1), the same effect is shown for ANNs in Figure 5.



(a) The mean and the standard deviation of the prediction error of the ANNs trained on the different feature sets of dataset 1.

(b) The training loss, validation loss and test loss for the ANNs trained on the different feature sets of dataset 1. The selected loss function is the MAPE.

Figure 5. Comparison of the performance of ANNs trained on different feature sets of dataset 1, the Exposin shaped dataset. The set with 3 features consists of the t_0, TOF and k_2. The feature set with 6 input variables consists of the full shape information.

The difference between the two feature sets shows the importance of feature engineering. For both algorithms goes: more features is not necessarily better, and some features decrease the loss function for a ML strategy more than others. Adding more features might improve the performance of the ML-algorithm, but also increases the time required to obtain a trained algorithm since more parameters need to be tuned. The feature set can be optimized in terms of accuracy and computational effort by testing individual features using a leave-one-out strategy, and determining whether features are correlated.

Kernels The different kernels selected for the GPs are also trained most successfully on dataset 3. The kernels all produced comparable results, so the differences are a matter of detail. In Figure 6, it can be seen that all kernels used in this paper are more or less suitable for GPR, since the minimum standard deviation of the prediction error is below 0.5 km/s. The best average performance is obtained using kernels 5, 6, 7, 12, 13, 15, 16, or 17. These kernels all consist of a PP kernel, added to another kernel. The other kernels sometimes results in GPs that are not trained. This can be seen by the number of GPs that result in a standard deviation of the prediction error above 2 km/s. Furthermore, the best performing GP with kernels 3, 9, 10, 11 and 14 is not as accurate as the GPs with other kernels. These kernels all consist of the product of a PP kernel with another kernel. Indeed, these kernels do not perform well specifically on the feature set containing the full state information; the set that resulted in the most accurate GPR. This shows the complexity of selecting the right kernels for the problem. The same kernel can improve or deteriorate the performance of the GP, depending on whether it is added to or multiplied with other kernels.

There is no single kernel that performs best on all feature sets. The kernels that perform best on each feature set are given in Table 8.

Table 8. Best performing kernels per feature set.

Feature set	Best performing kernel	Kernel Index
1	RBF+Matérn+PP	7
2	RBF+Matérn	4
3	(RBF+PP)×Matérn	13
4	Matérn + PP	6

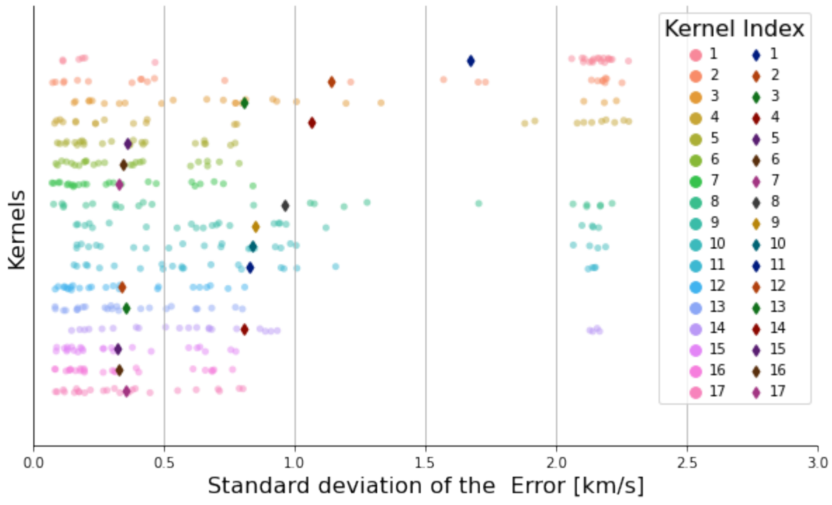
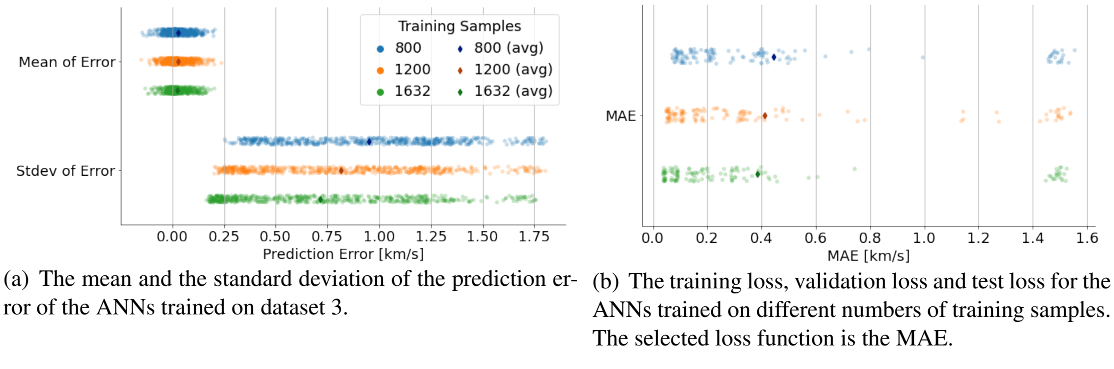


Figure 6. The standard deviation of the prediction error for GPs with different kernels. The dots depict the data, the diamonds give the mean value of the data.

Training Samples The number of training samples required to obtain a good result is the most important property that divides the two ML-algorithms evaluated in this paper. For very complex problems, ANNs can be trained on millions of training samples if necessary.[20] The practical limit for GPs is typically a couple thousands due to the computationally expensive matrix inversion. The most noticeable and well known trend for both algorithms is that more training data yields better results, because it gives the algorithms more information to generalize the underlying model. As mentioned before, ANNs are also less prone to overtraining when more distinct training samples are available.

In Figure 7, the mean and standard deviation of the prediction error, as well as the training-, validation-, and test loss of training different ANNs on the third dataset are shown for different training set sizes. Not only does a larger number of training samples contribute to a lower loss, the decrease in the test- and validation loss is larger than the training loss. The results in Table 9 show that the test loss for 800 training samples is about triple the training loss, and at 1632 samples, it is only double. Thus, the larger training dataset indeed reduces overtraining.

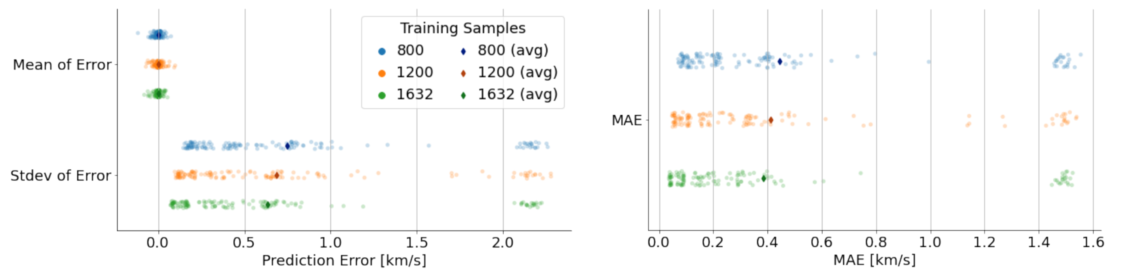


(a) The mean and the standard deviation of the prediction error of the ANNs trained on dataset 3.

(b) The training loss, validation loss and test loss for the ANNs trained on different numbers of training samples. The selected loss function is the MAE.

Figure 7. Comparison of the performance of ANNs trained on different numbers of training samples of dataset 3.

The same trend of the loss and standard deviation decreasing with an increasing number of training samples is present for GPR, as can be seen in Figure 8.



(a) The mean and the standard deviation of the prediction error of the GPs trained on different number of training samples from dataset 3.

(b) The training loss, validation loss and test loss for the GPs trained on different numbers of training samples from dataset 3. The selected loss function is the MAE.

Figure 8. Comparison of the performance of GPs trained on different numbers of training samples of dataset 3.

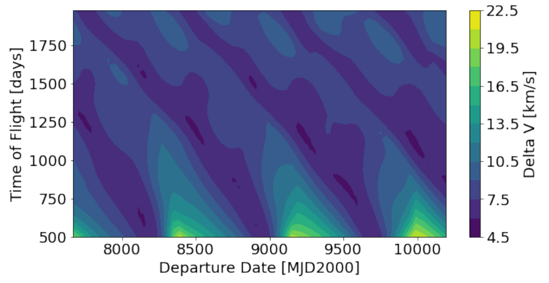
There is a cluster of data points with a standard deviation of the prediction error larger than 2 km/s, about the same order of magnitude as the standard deviation of the ΔV budget in dataset 3. These data points are related to the set of kernels that do not perform well on the data. The kernels suffering most from this issue can be found in Figure 6. A summary of the results shown in Figures 7 and 8 is given in Table 9.

Table 9. Summary of the results shown in Figures 7 and 8.

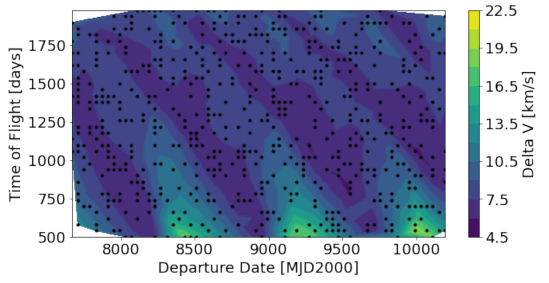
	ANN		GPR	
800 training samples	**Minimum**	**Mean**	**Minimum**	**Mean**
St. d. of the prediction error [km/s]	0.253	0.951	0.146	0.745
Training MAE [km/s]	0.022	0.193	-	-
Test MAE [km/s]	0.067	0.261	0.071	0.445
1200 training samples				
St. d. of the prediction error [km/s]	0.202	0.814	0.096	0.684
Training MAE [km/s]	0.020	0.174	-	-
Test MAE [km/s]	0.047	0.219	0.048	0.411
1632 training samples				
St. d. of the prediction error [km/s]	0.164	0.713	0.071	0.633
Training MAE [km/s]	0.019	0.160	-	-
Test MAE [km/s]	0.042	0.191	0.036	0.384

Algorithms selected for surrogate modelling The best performing ANN and GP are selected based on Tables 5, 6, 7, and 9. Both ML-algorithms perform best on dataset 3, using the full state information (feature set 1), standardized input data, and the maximum number of training samples. Concerning the hyperparameters, the ANN consists of 5 layers, 128 nodes per layer, a batch size of 32 samples and the MAE loss function. For the GPR, the selected kernel is the sum of the RBF, Matérn and PP kernel as given in Table 8. To compare the accuracy of both algorithms, an example of the prediction error is visualized in Figure 9. This is done by creating contour plots of the difference between the outcome of the ML-algorithm and the output of the test dataset. The input values of the test data are depicted using black dots.

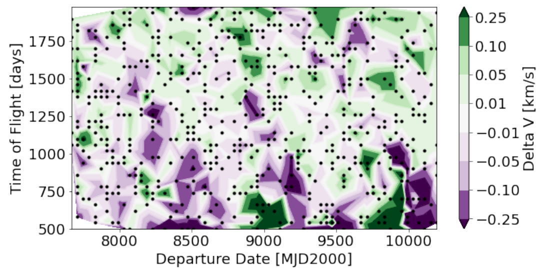
Figures 9(c) and 9(d) show the prediction error of the ANN and the GP respectively. For the



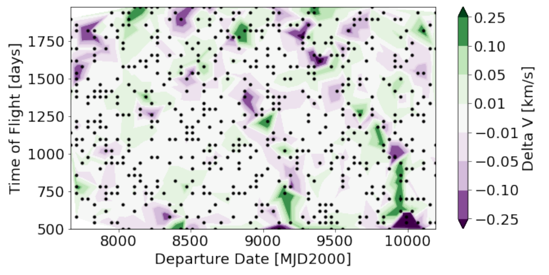
(a) Porkchop plot of the full dataset.



(b) Porkchop plot of the test data.



(c) Difference between the test data output and the output (prediction error) for the ANN.



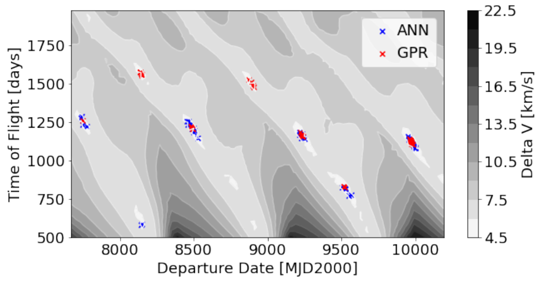
(d) Difference between the test data output and the output (prediction error) for the GP.

Figure 9. Estimates of the required ΔV budget for the different ML-algorithms.

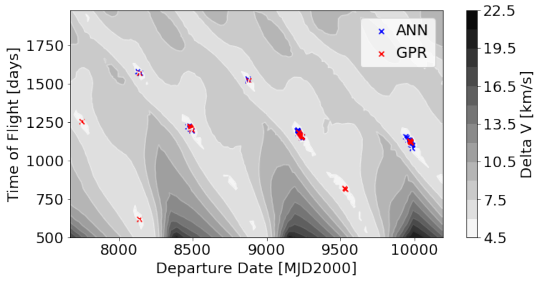
ANN, clearly more color is visible than for the GP, which shows the ANN predicted more often output with a deviation from the test output than the GP. This implies the GP predicts the required ΔV budget more accurately than the ANN. This conclusion is supported by the mean and standard deviation of the prediction error. The mean of the prediction error of the ANN is -0.012 km/s, and the standard deviation 0.18 km/s. For the GPR, the mean of the prediction error is -0.005 km/s, and the standard deviation 0.07 km/s. These numbers show that the GP outperforms the ANN on predicting the ΔV budget required to perform a low-thrust satellite trajectory to Mars after training on 1632 training samples. Of these two ML-algorithms, the GP took on average about three times longer to train on the dataset. There is no reason to assume that ANNs that require the same amount of training time would obtain better results. Increasing the width and depth of the ANNs does only improve the accuracy up to a certain level, which hints to a limit beyond which overtraining takes place. Increasing the number of epochs over which the ANN is trained only causes more overtraining.

Differential evolution

The next step in finding the optimal trajectory is to use an optimization algorithm to find the function input that yields the minimum ΔV budget. The differential evolution (DE) algorithm (a common population based algorithm used for finding optimal trajectories[2]) is selected to test the performance of the trained ML-algorithms in the surrogate configuration. In this set of experiments, a DE/best/1/bin with F = 0.5, Cr = 0.7 and NP = 15 is used. When no constraints are applied, the DE algorithm only gives feasible solutions on continuous input spaces.[28] Of the four feature sets given in Table 2, only set 4 is continuous. However, using the ephemeris data of Earth and Mars, the other feature sets can be restricted to the same two-dimensional continuous space as well. Since the pattern in Figure 9(a) is repetitive, multiple local minima can be found. These minima can have similar ΔV values, which makes it difficult for the DE-algorithm to find the global minimum, i.e. the set of input parameters that yield the optimal trajectory. Another difficulty that arises when having multiple local minima with similar ΔV values is that the location of the minimum determined by the DE-algorithm depends on the training data used to train the machine learning algorithms. To show the most likely inputs and output for the optimal trajectory, the experiment of training the ML-algorithms and finding the optimum using the DE-strategy was repeated 50 times. The results are shown in Figure 10.



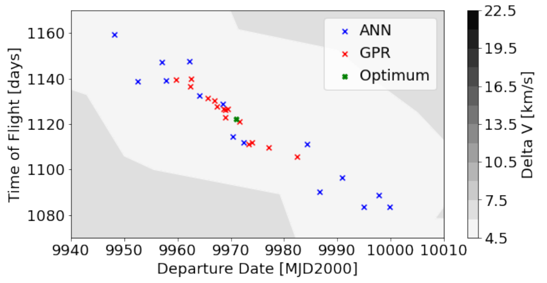
(a) Predicted input variables (Departure date and TOF) for the optimal trajectory to Mars, obtained using the surrogate model trained on these input variables.



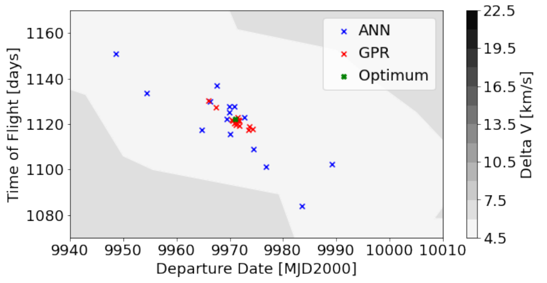
(b) Predicted input variables (Departure date and TOF) for the optimal trajectory to Mars, obtained using the surrogate model trained on the state at departure and arrival.

Figure 10. Locations of the 50 determined local optima on the domain [7670, 10190] for the departure date and [500, 2000] for the TOF, obtained using a surrogate configuration DE-strategy with a trained ANN (blue) or GP (red).

A closer look at Figure 10 shows that predictions obtained using GPR are on average closer to the nearest local minimum, shown in Figure 11. The chosen domain contains the global minimum within the training dataset, located at (9985, 1100).[14,27]



(a) Figure 10(a) on the domain [9940, 10010] for the departure date and [1070, 1170] for the TOF.



(b) Figure 10(b) on the domain [9940, 10010] for the departure date and [1070, 1170] for the TOF.

Figure 11. Distribution of the data points located in the vicinity of the local minimum in Figure 10. Input variables of the optimal trajectory are shown in green.

To determine the accuracy of both ML-algorithms, there are two important properties: the accuracy of the predicted input variables for the optimal trajectory, i.e. the TOF and departure date, and the accuracy of the predicted output, i.e. the ΔV budget.

To determine the accuracy for the input, the Euclidean distance between the estimate and the nearest local minimum is used since both the departure date as the TOF are measured in days. Table 10 provides the confirmation of what can be seen in Figure 11. The GP is more accurate and precise, both in predicting the input as the output of the optimal trajectory.

Table 10. Accuracy of the predictions of the input and the output for the different ML algorithms.

	Average distance to nearest local minimum [days]	Mean of the output [km/s]	Standard deviation of the output [km/s]
GPR, feature set 3	10.6	5.777	0.025
ANN, feature set 3	27.2	5.770	0.042
GPR, feature set 1	7.44	5.776	0.020
ANN, feature set 1	14.8	5.752	0.032

The best results are as expected obtained after training on feature set 1. In Table 11, the predicted inputs and output of the local minima in Figure 10(b) are given individually.

Table 11. Local minima found by the DE algorithm in the surrogate configuration (Departure date, time of flight, number of points closest to the minimum and the ΔV required in km/s to perform the trajectory) and the closest local minimum output of the hodographic shaping algorithm.

Hodographic			**ANN**				**GPR**			
Dep. date [MJD2000]	**TOF [days]**	**ΔV [km/s]**	**Dep. date [days]**	**TOF [days]**	**# -**	**ΔV [km/s]**	**Dep. date [days]**	**TOF [days]**	**# -**	**ΔV [km/s]**
8145	602	5.87			0		-9	+17	1	5.87
8139	1573	5.80	-4	+5	2	5.81	0	-1	1	5.80
8486	1213	5.79	-5	+1	9	5.69	-3	+4	10	5.77
8875	1527	5.80	-3	-3	2	5.75	+1	-3	1	5.79
9228	1167	5.77	-5	-5	21	5.71	+1	-2	20	5.76
9505	840	5.83			0		+21	-22	2	5.82
9971	1122	5.76	-3	-3	16	5.70	0	0	15	5.75

Most of the minima obtained by the DE-strategy have a departure date around 9228 or 9971 in the MJD2000 system,[29] and a TOF of 1167 or 1122 days. The surrogate method involving the GP also gives these inputs as the trajectories requiring the lowest ΔV budget. In order to assess the accuracy of these given local minima, the output of the surrogate method served as the general area for a grid search using the hodographic shaping method. The output of this search is also given in Table 11. Most of the local minima predicted by the surrogate method are reasonably close to the minima obtained with the hodographic shaping method, especially those obtained using GPR. The biggest difference between the ANN and the GPR in Table 11 is the accuracy of the predictions of the required ΔV value. The GP is more accurate, the average prediction of the ΔV value is 0.01 km/s off. This cannot be said for the departure date and the TOF. For most of the local minima, the prediction of the GP is closer to the local minimum. However, there are two outliers, where there is 1 prediction close to the local minimum, but off by up to 22 days for the TOF and the departure date. These might be an issue, so it is important to repeat the optimization process multiple times to make sure there is more valuable data than outliers.

CONCLUSION

Three main conclusions can be drawn. The foremost conclusion is that the GP provides more accurate results in a surrogate method than the ANN. Not only does the GP predict the inputs of the optimal trajectory more accurately than the ANN, but the ΔV required to perform the optimal trajectories is predicted more accurately by the GP as well. This conclusion is not surprising after showing that the GP's predicting capabilities outperform the ANN on all three datasets tested, and on the two best performing feature sets. It also required less training samples to obtain a similar or better result than the ANN. This came at the cost of the computational effort; Training the best performing GP consistently took about three times as long as training the best performing ANN.

The second conclusion is that the dataset defines whether it is possible to use a surrogate method. Both ML-algorithms performed best on the dataset with the lowest deviation in the data, although the smallest number of trajectories was used for training. The standard deviation of the error in the prediction of the ΔV budget for dataset 1 and 2 exceeded the minimum value for the ΔV budget, which makes the ML-algorithms unusable. The most notable difference between the three datasets is the number of jumps in adjacent ΔV values. These jumps are the largest in dataset 1, and the smallest in dataset 3. They might have caused the optimization algorithm for the loss function of the ML-algorithm to be stuck in a local minimum.

The final conclusion is that scaling the data greatly benefits the performance of both ML-algorithms. When no scaling is applied to dataset 3, the accuracy of the predictions of the ML-algorithms is not better than predicting the mean value of the dataset as ΔV budget. Both standardization and normalization decreased the standard deviation of the prediction error by about 90%. Between those methods, no significant difference was found.

REFERENCES

[1] B. Dachwald, "Optimization of very-low-thrust trajectories using evolutionary neurocontrol," *Acta Astronautica*, Vol. 57, No. 2-8, 2005, pp. 175–185, 10.1016/j.actaastro.2005.03.004.

[2] A. Shirazi, J. Ceberio, and J. A. Lozano, "Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions," *Progress in Aerospace Sciences*, 2018.

[3] C. Ampatzis and D. Izzo, "Machine learning techniques for approximation of objective functions in trajectory optimisation," *Proceedings of the ijcai-09 workshop on artificial intelligence in space*, 2009, pp. 1–6.

[4] D. Izzo, C. I. Sprague, and D. V. Tailor, *Machine learning and evolutionary techniques in interplanetary trajectory design*, pp. 191–210. Springer, 2018.

[5] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, Vol. 2, No. 5, 1989, pp. 359–366.

[6] H. Shang and Y. Liu, "Assessing accessibility of main-belt asteroids based on Gaussian process regression," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 5, 2017, pp. 1144–1154.

[7] H. Peng and X. Bai, "Gaussian Processes for improving orbit prediction accuracy," *Acta Astronautica*, Vol. 161, 2019, pp. 44–56.

[8] L. Bouwman, Y. Liu, and K. Cowan, "Gaussian process models for preliminary low-thrust trajectory optimization," *Proceedings of the AAS/AIAA Astrodynamics Conference Aug. 11-15, 2019, Portland* (E. Hussein, K. Horneman, C. Scott, and B. Hansen, eds.), Advances in the Astronautical Sciences Series, AAS, 2020. 2019 AAS/AIAA Astrodynamics Specialist Conference ; Conference date: 11-08-2019 Through 15-08-2019.

[9] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[10] A. Mereta, D. Izzo, and A. Wittig, "Machine learning of optimal low-thrust transfers between near-earth objects," *International Conference on Hybrid Artificial Intelligence Systems*, Springer, 2017, pp. 543–553.

[11] H. Li, S. Chen, D. Izzo, and H. Baoyin, "Deep networks as approximators of optimal low-thrust and multi-impulse cost in multitarget missions," *Acta Astronautica*, 2019.

[12] A. E. Petropoulos and J. M. Longuski, "Shape-based algorithm for automated design of low-thrust gravity-assist trajectories," *AAS/AIAA Astrodynamics Specialists Conference*, Vol. 41, 2004, p. 10.

[13] D. Izzo, "Lambert's problem for exponential sinusoids," *Journal of guidance, control, and dynamics*, Vol. 29, No. 5, 2006, pp. 1242–1245.

[14] 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.

[15] D. Izzo, M. Märtens, and B. Pan, "A Survey on Artificial Intelligence Trends in Spacecraft Guidance Dynamics and Control," *arXiv preprint arXiv:1812.02948*, 2018.

[16] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Cambridge: the MIT Press, 2006.

[17] L. Stubbig and K. Cowan, "Improving the evolutionary optimization of interplanetary low-thrust trajectories using an artificial neural network surrogate model," 2020. 2020 AAS/AIAA Astrodynamics Specialist Conference ; Conference date: 9-08-2020 Through 12-08-2020.

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

[19] G. H. Golub and C. F. Van Loan, "Matrix computations, 4th," *Johns Hopkins*, 2013.

[20] D. Izzo, E. Öztürk, and M. Märtens, "Interplanetary Transfers via Deep Representations of the Optimal Policy and/or of the Value Function," *arXiv preprint arXiv:1904.08809*, 2019.

[21] S. F. Crone, J. Guajardo, and R. Weber, "The impact of preprocessing on support vector regression and neural networks in time series prediction," *DMIN*, 2006, pp. 37–44.

[22] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, 2019, p. 105524.

[23] N. M. Nawi, W. H. Atomi, and M. Rehman, "The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks," *Procedia Technology*, Vol. 11, 2013, pp. 32–39.

[24] D. Tailor and D. Izzo, "Learning the optimal state-feedback via supervised imitation learning," *Astrodynamics*, Vol. 3, No. 4, 2019, pp. 361–374, 10.1007/s42064-019-0054-0.

[25] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[26] C. Sánchez-Sánchez and D. Izzo, "Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 5, 2018, pp. 1122–1135, 10.2514/1.G002357.

[27] L. Stubbig and K. Cowan, *Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories*. Master thesis, Delft University of Technology, 2019.

[28] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, Vol. 11, No. 4, 1997, pp. 341–359.

[29] D. D. McCarthy, "The julian and modified julian dates," *Journal for the History of Astronomy*, Vol. 29, No. 4, 1998, pp. 327–330.

3 Additional results

In addition to the results presented in Chapter 2, other results are obtained that support the paper and answer the research questions that were not answered in the paper. In this section, the verification of well known trends for hyperparameters of ANNs is presented, as well as a complete list of the kernels used for the GPs and a comparison of the computational times for GPs. Furthermore, it is verified that the class imbalance problem is not deteriorating the accuracy of the results, the performance of the both algorithms outside the training range is shown, and the optimal trajectory is shown.

3.1. Verification of results of the grid search for ANNs

In the paper, some trends for the hyperparameters of the ANN are determined based on literature. In order to verify that the grid search was performed correctly, these trends need to be present in the data as well. Since training the ANNs was performed most accurately for dataset 3 in the paper, generated using the hodographic shaping method, this dataset is used to show the trends. Similar results were obtained for the other datasets, but the data was very noisy and large clusters of similar inaccurate results were obtained since the ANNs did not always train successfully. Furthermore, the data was normalized in all of the experiments described in this section, because in the paper is shown that scaling is essential to obtain accurate results.

Not only are the MAE of the ANNs and statistics on the output determined in the grid search, the time required to train the ANNs was logged as well[1]. The computation time is not a very reliable measure, since it depends on the specifications of the CPU and GPU, but some general trends can be spotted in the data. Especially when comparing the computation time to the number of parameters that need to be optimized, which can be determined as follows:

$$N_{Par} = \left(N_{Nodes} \cdot N_{feat} + N_{Nodes}\right) + \left(N_{Layers} \cdot \left(N_{Nodes}^2 + N_{Nodes}\right)\right) + \left(N_{Nodes} \cdot N_{Outputs} + N_{Outputs}\right) \tag{3.1}$$

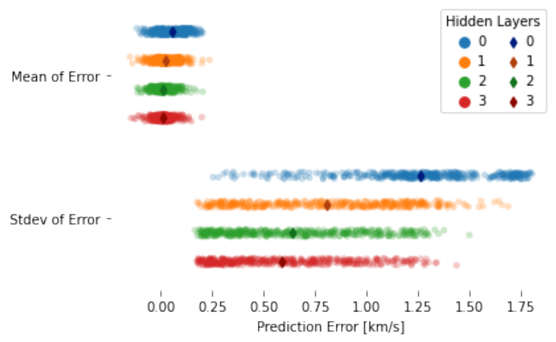
This means that the largest network, containing 3 hidden layers with 128 nodes per layer, trained on input with 14 features, contains 51585 parameters that are optimized on 1632 training samples in 51 batches of 32, repeated 500 times.

1. *Increasing the number of layers (depth) of an ANN increases the accuracy of the ANN, but also increases the time required to train the ANN.*

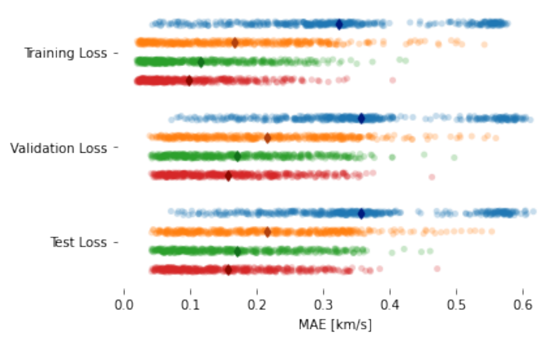
   In the grid search described in the paper, the number of hidden layers was varied between 0 and 3. An ANN without hidden layers consists of an input and output layer only, and an ANN with 3 hidden layers consists of 5 layers in total. The comparison between the ANNs is shown in Figure 3.1 and Table 3.1.

   In Figure 3.1 and Table 3.1 is shown that the first part of the statement can be verified. The standard deviation of the prediction error decreases and the training-, validation- and test losses do so as well. Since no measures are taken to prevent overtraining, the test and validation loss

[1] Intel (R) Core (TM) i7-6700HQ CPU 2.60GHz, 8 GB RAM, NVIDIA Quadro M1000M GPU.



(a) The training loss, validation loss and test loss of the ANNs containing different numbers of hidden layers.
The selected loss function is the MAE.



(b) The mean and the standard deviation of the prediction error of the ANNs different numbers of hidden layers.

Figure 3.1: Comparison of the performance of ANNs containing different numbers of hidden layers. The same legend applies to both figures.

have great similarity. The largest improvement is found between 0 and 1 hidden layers. What is most interesting is that the average value of the standard deviation of the prediction error and the different losses decreases as more layers are added. The minimum of the standard deviation and the losses, however, does not decrease when more than one hidden layer is introduced. This means that for small ANNs, it is beneficial to add more layers. For wide ANNs, containing many layers per node, no increase in performance is obtained when more hidden layers are added.

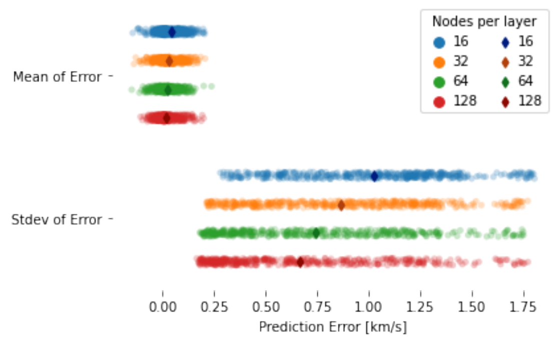
Hidden Layers	**0**		**1**		**2**		**3**	
	Min	Mean	Min	Mean	Min	Mean	Min	Mean
Mean of prediction Error [km/s]	-	0.058	-	0.023	-	0.012	-	0.012
Stdev of prediction Error [km/s]	0.253	1.263	0.175	0.807	0.165	0.641	0.175	0.591
Training Loss [km/s]	0.043	0.322	0.020	0.166	0.019	0.115	0.019	0.098
Validation Loss [km/s]	0.071	0.356	0.039	0.214	0.040	0.169	0.041	0.157
Test Loss [km/s]	0.071	0.356	0.042	0.214	0.042	0.169	0.042	0.157
Computation Time [mm:ss]	00:02	02:28	00:06	02:56	00:12	03:10	00:08	03:35

Table 3.1: Comparison of the performance of ANNs containing different numbers of hidden layers.

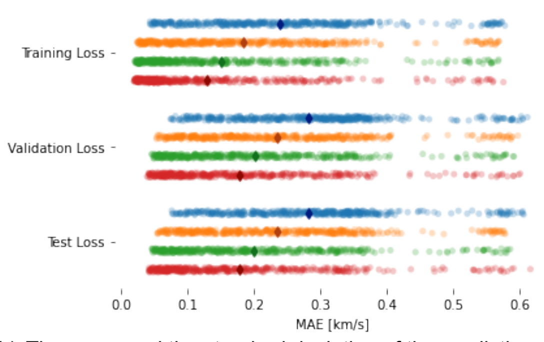
The general trend for the computational effort is that adding more layers increases the time required to train the ANN. Since the number of parameters that need to be optimized increases linear with the number of layers (see Eqn.(3.1)), it is expected that the computational time also increases linearly with an increasing number of layers. For the mean computation time, this is roughly true. Table 3.1 also shows why the computation time is not a reliable measure to evaluate though. The minimum computation time for an ANN with three hidden layers is lower than for two hidden layers. in this thesis, no research is performed to find out where the difference comes from, but it is expected that the GPU underclocks when overheating. Since these runs were not performed in one go, the temperature of the GPU might have varied over time. Even though the computation time lacks the desired accuracy, the general trend stated in the paper is confirmed.

2. *Increasing the number of nodes per layer (width) increases the accuracy of the ANN, but also increases the time required to train the ANN.*
   In the grid search, the number of nodes per layer was varied between 16, 32, 64, and 128. The comparison is shown in Figure 3.2 and Table 3.2.

   Similar to increasing the number of layers, increasing the number of nodes results in a lower standard deviation of the prediction error. The mean value of the standard deviation decreases from 1.025 for 16 nodes per layer, to 0.669 for 128 nodes. Unlike for the number of layers, increasing the number of nodes per layer does decrease the minimum value of the standard deviation of the prediction error as well. In Figure 3.2, both the standard deviation of the prediction



(a) The training loss, validation loss and test loss for the ANNs containing different numbers of nodes per layer. The selected loss function is the MAE.



(b) The mean and the standard deviation of the prediction error of the ANNs for different numbers of nodes per layer.

Figure 3.2: Comparison of the performance of ANNs containing different numbers of nodes per layer. The same legend applies to both figures.

error, as the losses consist of 2 clusters of points. The clusters with the highest standard deviation or loss value are mainly related to the inaccuracy of the ANNs without hidden layers.

Nodes per layer	**16**		**32**		**64**		**128**	
	Min	Mean	Min	Mean	Min	Mean	Min	Mean
Mean of prediction Error [km/s]	-	0.039	-	0.027	-	0.021	-	0.018
Stdev of prediction Error [km/s]	0.280	1.025	0.212	0.867	0.181	0.742	0.165	0.669
Training Loss [km/s]	0.043	0.237	0.024	0.184	0.020	0.150	0.019	0.129
Validation Loss [km/s]	0.074	0.282	0.053	0.234	0.046	0.200	0.039	0.178
Test Loss [km/s]	0.076	0.282	0.054	0.235	0.046	0.199	0.042	0.178
Computation Time [mm:ss]	00:02	02:37	00:03	02:42	00:04	02:50	00:05	04:00

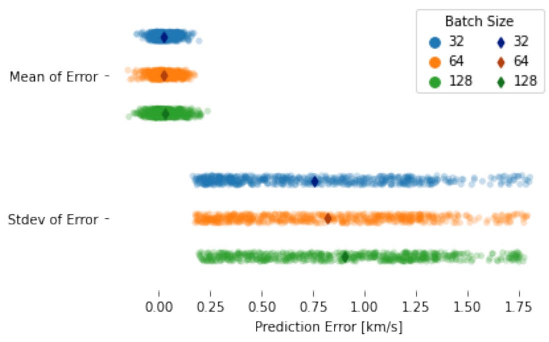
Table 3.2: Comparison of the performance of ANNs containing different numbers of nodes per layers.

In Table 3.2, it can be seen that the computation time increases as the number of nodes per layer increase. Eqn. (3.1) shows that the number of parameters that are tuned increase quadratically with the number of nodes per layer. The computation time can not be used to show this quadratic trend, but the relation stated in the paper is confirmed.

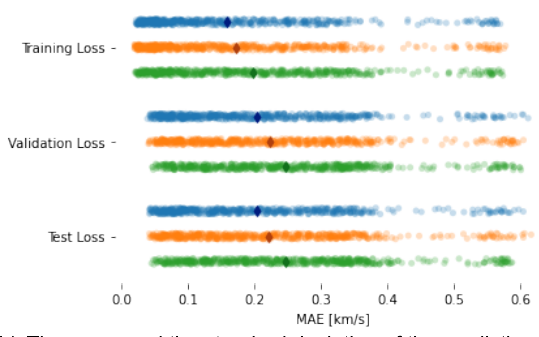
3. *Increasing the batch size of the training samples decreases the accuracy of the ANN, but also decreases the time required to train the ANN.*
   For the batch size, the trend is opposite to the trend for the other hyperparameters. This is because the batch size divides the dataset in a number of batches before the training of the ANN takes place. A batch size of 32 samples for 1632 training samples results in 51 updates of the ANN every epoch. In the grid search presented in the paper, the batch sizes were varied between 32, 64, and 128 training samples. The comparison is shown in Figure 3.3 and Table 3.3.

   In Figure 3.3, it can be seen that the standard deviation of the prediction error and the losses increase with an increasing batch size. The effect of changing the batch size is not as big as changing the other hyperparameters. Changing from 32 samples to 128 samples per batch increases the minimum standard deviation of the prediction error from 0.165 km/s to 0.195 km/s, whereas changing the number of nodes per layer from 32 to 128 decreases the minimum standard deviation from 0.212 km/s to 0.165 km/s. The reason the change for the batch size is not so large is because an increase in batch size does not decrease the number of training samples seen by the ANN. It only decreases the number of updates. Given enough epochs, the performance of ANNs with different batch sizes should be similar, the time required to get to this limit is different for every ANN.

   The relation between the computation time and the batch size is expected to be reciprocal. A batch size of 32 for 1632 training samples, results in 51 updates of the ANN every epoch. Batch



(a) The training loss, validation loss and test loss for the ANNs training on data in different batch sizes. The selected loss function is the MAE.



(b) The mean and the standard deviation of the prediction error of the ANNs training on data in different batch sizes.

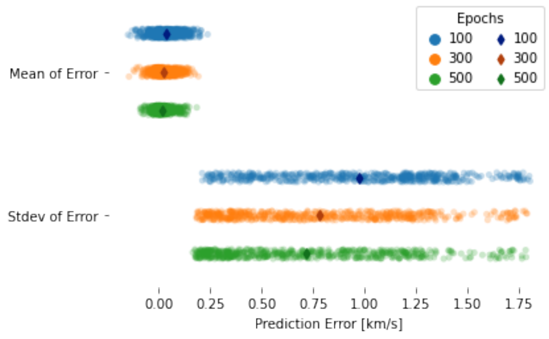
Figure 3.3: Comparison of the performance of ANNs training on data in different batch sizes. The same legend applies to both figures.

Batch Size	32		64		128	
	Min	Mean	Min	Mean	Min	Mean
Mean of prediction Error [km/s]	-	0.024	-	0.023	-	0.031
Stdev of prediction Error [km/s]	0.165	0.755	0.175	0.821	0.195	0.901
Training Loss [km/s]	0.021	0.157	0.019	0.172	0.020	0.197
Validation Loss [km/s]	0.039	0.202	0.040	0.222	0.046	0.247
Test Loss [km/s]	0.042	0.203	0.042	0.222	0.045	0.247
Computation time [mm:ss]	00:03	04:15	00:02	02:49	00:02	02:03

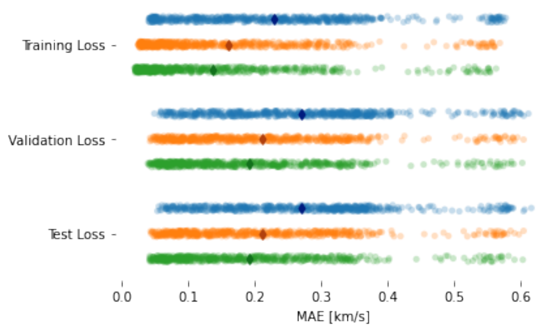
Table 3.3: Comparison of the performance of ANNs Training on data in different batch sizes.

sizes of 64 and 128 result in 26 and 13 updates per epoch. Even though the exact nature of the relation is unclear, the standard deviation of the prediction error decreases with an increase of batch size, and so does the computation time, so the relation stated in the paper is confirmed.

4. *Increasing the number of epochs the ANN is trained over increases the accuracy of the ANN, but also increases the time required to train the ANN.*
   In the grid search presented in the paper, the number of epochs is varied between 100, 300 and 500. The comparison between these numbers is shown in Figure 3.4 and Table 3.4.



(a) The training loss, validation loss and test loss for the ANNs training over different numbers of epochs.
The selected loss function is the MAE.



(b) The mean and the standard deviation of the prediction error of the ANNs training over different numbers of epochs.

Figure 3.4: Comparison of the performance of ANNs training over different numbers of epochs. The same legend applies to both figures.

The first part of the statement in the paper is conditionally true. From the data in Figure 3.4 and Table 3.4, it can be seen that the accuracy indeed increases over an increasing number of

number of epochs	100		300		500	
	Min	Mean	Min	Mean	Min	Mean
Mean of prediction Error [km/s]	-	0.037	-	0.023	-	0.019
Stdev of prediction Error [km/s]	0.210	0.975	0.183	0.783	0.165	0.719
Training Loss [km/s]	0.040	0.229	0.024	0.160	0.019	0.137
Validation Loss [km/s]	0.049	0.269	0.042	0.211	0.039	0.192
Test Loss [km/s]	0.053	0.269	0.043	0.211	0.042	0.192
Computation Time [mm:ss]	00:02	01:40	00:04	03:02	00:06	04:24

Table 3.4: Comparison of the performance of ANNs training over different numbers of epochs.

epochs, since the standard deviation of the error is decreasing, and so are the loss functions. However, the decrease is not linear, which raises the question whether is stops. The example of the training- and validation loss as function of the number of epochs in Figure 3.5 shows indeed that especially for the validation loss, the decrease stops after a number of epochs.

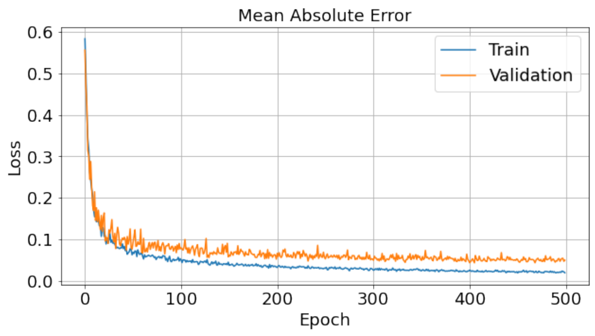


Figure 3.5: The training- and validation loss of an ANN plotted against the number of epochs the ANN is trained on.

As can be seen in Figure 3.5, the training loss decreased more than the validation loss. Whether and when this is considered overtraining is unclear. It is expected that the training loss is lower than the test loss, since the ANN obtained information on the training samples, but not on the test samples. When the validation loss starts to increase over epochs, it is certain that overtraining is taking place. However, one might argue that when the validation loss at 300 samples equals the loss at 500 samples, it is not necessary to train over 500 samples and early stopping is performed to prevent overtraining.
The fact that training over a larger number of epochs takes more time is evident. In order to train over 300 epochs, the ANN has to be trained over 100 epochs first. The relation between the amount of epochs and the computation time seems also the most clear of all. Since every new epoch is a repetition of the previous one, the relation between the computation time and the number of epochs is expected to be linear. Indeed, on average performing 200 more epochs after the first 100 takes 1 minute and 24 seconds. following this trend back to 0 epochs, the average time to create the structure of the ANN and fill it with pseudo-random numbers is 58 seconds. Indeed it was observed that larger ANNs require more time to start, but again, the computation time is not a very reliable measure so no accurate conclusions can be drawn from this data. The fourth statement in the paper is therefore considered practically true. Up to a certain level, the accuracy keeps increasing with an increasing number of epochs, and when this is no longer the case, measures to prevent overtraining are usually applied.

3.2. Comparison of the computation time for GPs

For all configurations of ML-algorithms evaluated in the grid search, the time required to perform the calculations is determined as well. This time is not a very accurate measure because is it dependent on many factors, of which some cannot directly be influenced. However, in the previous section is shown that there is a general trend present in the time required to perform the calculations on dataset 3 using ANNs. The trends for the GPs are shown in Tables 3.5, 3.6 and 3.8. To make the comparison easier, the MAE and the mean and standard deviation of the prediction error are repeated from the paper. The parameters that were varied in the grid search are: the feature sets, the number of training samples and the kernels.

Feature Sets	**Set 1**		**Set 2**		**Set 3**		**Set 4**	
	min	mean	min	mean	min	mean	min	mean
Mean of the prediction error [m/s]	-	2.494	-	-1.576	-	-2.273	-	0.291
Std of the prediction error [km/s]	0.071	0.593	0.238	0.525	0.528	0.971	0.127	0.661
Test loss [km/s]	0.036	0.378	0.122	0.290	0.250	0.552	0.068	0.432
Computational time [mm:ss]	00:01	06:57	00:03	02:00	00:04	01:52	00:01	00:23

Table 3.5: Comparison of different features sets used by the GPs, in terms of accuracy and computation time. Since the value of the mean of the prediction error is very small, it is given in m/s instead of km/s.

In Table 3.5, it is shown that the average computational time is dependent on the number of features. The average time to fit a GP for 2 features is about half a minute, for 4 features it is 2 minutes and for 14 features, it is 7 minutes. The trend is that the more features the training samples have, the longer the GP takes to train.

Number of training samples	**800**		**1200**		**1632**	
	min	mean	min	mean	min	mean
Mean of the prediction error [m/s]	-	-0.668	-	0.549	-	-0.679
Std of the prediction error [km/s]	0.146	0.745	0.096	0.684	0.071	0.633
Test loss [km/s]	0.071	0.445	0.048	0.411	0.036	0.384
Computational time [mm:ss]	00:01	01:08	00:01	02:43	00:02	04:33

Table 3.6: Trend of the number of training samples used to train the GPs, in terms of accuracy and computation time. Since the value of the mean of the prediction error is very small, it is given in m/s instead of km/s.

According to the Table 3.6, the training time does increase with an increasing number of training sample, as stated in literature. Whether this increase is cubic cannot be determined with only three different numbers of training samples.

To identify the best performing kernel, seventeen different kernels were investigated in the grid search. These kernels are every combination of the radial basis function (RBF) kernel, the Matérn kernel and the piecewise polynomial (PP) kernel of order 2. These kernels are named K1, K2 and K3 respectively in Table 3.7. The order in which the kernels are numbered in the paper is given in Table 3.7.

1	K1	**4**	K1+K2	**8**	K1×K2	**12**	K1×(K2+K3)	**15**	K1×K2+K3
2	K2	**5**	K1+K3	**9**	K1×K3	**13**	K2×(K1+K3)	**16**	K1×K3+K2
3	K3	**6**	K2+K3	**10**	K2×K3	**14**	K3×(K1+K2)	**17**	K2×K3+K1
		7	K1+K2+K3	**11**	K1×K2×K3				

Table 3.7: The different kernels used in the grid search. K1=RBF, K2=Matérn, K3=PP.

The performance and computational time required for all the kernels is shown in Table 3.8. In this table, it can be seen that the kernels do have the largest influence on the computational time. The most important observations concerning the accuracy and required computation time of the kernels are all related to Kernel 3, the PP kernel:

- Adding K2 and K3 is relatively time-consuming
- Adding K3 to any kernel gives accurate results
- Multiplying a kernel with K3 takes very little time (Kernel 14 takes on average even less time to train than kernel 4)

Rasmussen writes about the third conclusion[15] that the piecewise polynomial kernel provides compact support: When the distance between two points exceeds a certain threshold, the covariance becomes exactly zero. This results in sparsity of the covariance matrix, which decreases the required time to invert the matrix. The kernel used in the surrogate optimization algorithm is kernel 7, since it performed best on feature set 1. However, based on the observations above, kernel 5 and kernel 13 are also very promising, since they require less time to perform the computations than kernel 7, but the decrease in performance is not very large.

Kernel	Mean [m/s]	Std. [km/s]		Test loss [km/s]		Computational time [mm:ss]	
		Min	Mean	Min	Mean	Min	Mean
1	1.161	0.113	1.672	0.063	1.148	00:04	00:44
2	-0.876	0.074	1.137	0.039	0.749	00:04	00:55
3	-0.244	0.156	0.808	0.087	0.480	00:01	00:28
4	2.431	0.082	1.062	0.039	0.693	00:07	03:42
5	-3.471	0.092	0.358	0.046	0.182	00:01	01:37
6	1.510	0.086	0.345	0.039	0.171	00:13	02:31
7	-1.701	0.071	0.328	0.036	0.165	00:22	07:11
8	-7.148	0.078	0.962	0.040	0.610	00:07	03:20
9	-4.132	0.164	0.850	0.087	0.512	00:02	01:01
10	1.732	0.166	0.839	0.091	0.501	00:04	01:08
11	1.197	0.156	0.826	0.081	0.490	00:06	01:52
12	3.519	0.084	0.336	0.042	0.166	00:22	04:55
13	2.265	0.083	0.354	0.042	0.172	00:02	04:58
14	3.148	0.146	0.807	0.083	0.480	00:02	02:41
15	-2.424	0.087	0.322	0.043	0.163	00:23	04:33
16	1.053	0.082	0.329	0.040	0.165	00:26	03:18
17	-2.543	0.085	0.352	0.045	0.176	00:04	02:39

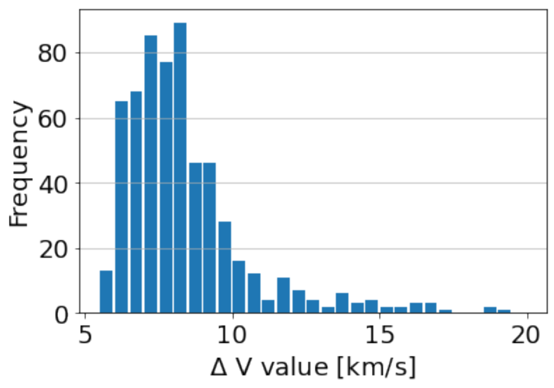
Table 3.8: Performance and computational time for each kernel. Since the value of the mean of the prediction error is very small, it is given in m/s instead of km/s.

Utilizing feature set 1, 1632 training samples and kernel 7, the average training time for the GP used in the surrogate model was 28 minutes and 19 seconds. This is about three times as long as training the ANN, which took on average 10 minutes and 47 seconds.

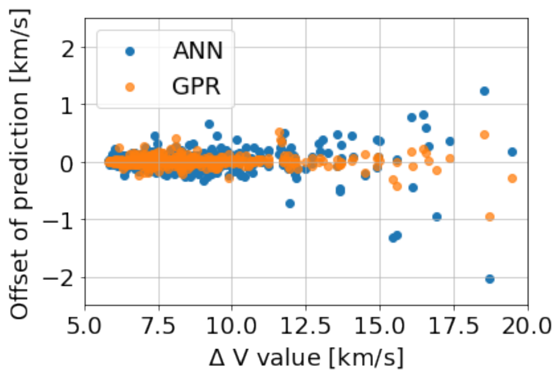
3.3. Class imbalance problem

The class imbalance problem is an issue mostly known for complicating classification problems [25], but it subtly influences the inference capabilities of ML-algorithms as well. For classification problems, when a vast majority of the samples belong to class A and very little to class B, the algorithm will predict all samples to be of class A in order to maximize the prediction accuracy. A similar issue occurs for regression problems. For data that has an output distributed normally over the output range, the largest offset between the test data and the predictions will be in the regions with the highest or lowest ΔV values because the lowest amount of data is available int these regions. To check whether this is an issue for this problem, the distribution of the samples is shown, and the offset of the prediction as function of the ΔV value.

Figure 3.6(a) shows an example of the distribution of a test dataset. The training dataset follows the same distribution, since they are drawn from the same dataset and the number of samples is big enough. In Figure 3.6(b) is shown how well the ANN and GP predict the ΔV budget of the test samples. Figure 3.6 shows that there is some imbalance present in the distribution of the test samples, but for this problem, it is not an issue. The ΔV value of the majority of the data is in between 5 and 10 km/s. In Figure 3.6(b), it can be seen that for this range, the data is the most accurate. When the density of the data is smaller (in between 12 and 20km/s), the offset of the prediction is larger. Since the optimal trajectory has the lowest ΔV value and plenty of data is available in this range, the prediction error is small and the imbalance is not an issue.



(a) Distribution of the ΔV value of a test dataset.



(b) Predictions of the ΔV value of the test samples.

Figure 3.6: Distributions of the ΔV values of a test dataset, and the offset of the predictions.

3.4. Performance of the algorithms outside the training range

Both ML-algorithms are trained on trajectories departing between 01-01-2021 (7671 in MJD2000) and 31-12-2027 (10227 in MJD2000). The pattern of the ΔV value out of this range is similar, as can be seen in Figure 3.7.

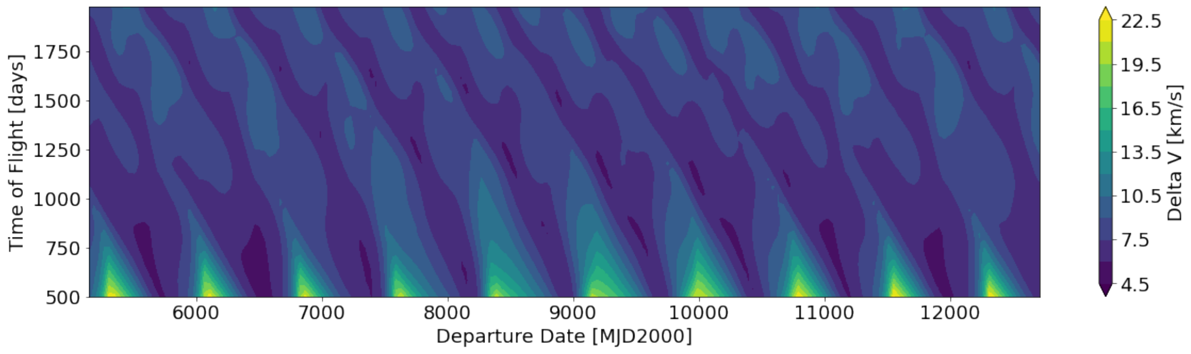


Figure 3.7: Porkchop plot of departures dates from 01-01-2014 to 31-12-2034.

It is interesting to see whether the ML-algorithms have picked up on this pattern, or only predict the data accurately within the training range. In general, ML-algorithms do not perform well outside the training range. The GP is modelled to have the mean value of the dataset, and the deviation is defined by the samples in the training dataset, and in between these samples by the kernel function. Therefore, outside the training range, the GP is expected to return the mean value of the training dataset as output. The ANN on the other hand consist of a combination of polynomials, of which the shape is defined by the amount of layers, the amount of nodes per layer and the activation functions. Therefore, it is expected that the behavior of the ANN is similar to that of polynomial regression. The reason to look into the performance is because in multiple papers, it is mentioned that the performance of ANNs is reasonable even outside the training range [10, 24]. The results of predictions for the required ΔV budget by ANNs and GPs trained on the departure date and the time of flight only (feature set 3 in the paper) is shown in Figures 3.8 and 3.9.

As can be seen, the behavior of the ANN and the GP is as expected. The required ΔV budget predicted by the ANN quickly rises for the earlier departure dates in the left side of the plot. On the right side, the same is happening, but a bit slower. The GP on the other hand predicts a value of around 8.45 km/s for most of the input parameters out of the training range. This is indeed about the same value as the mean of the dataset. When feature set 1 is used to train the ML-algorithms on, the behavior of the results out of the training range changes drastically. The predictions of the required ΔV budget for samples out of the training range for the ANN and the GP are shown in Figures 3.10 and 3.11.

In Figures 3.10 and 3.11 can be seen that the trend present within the range of the training data

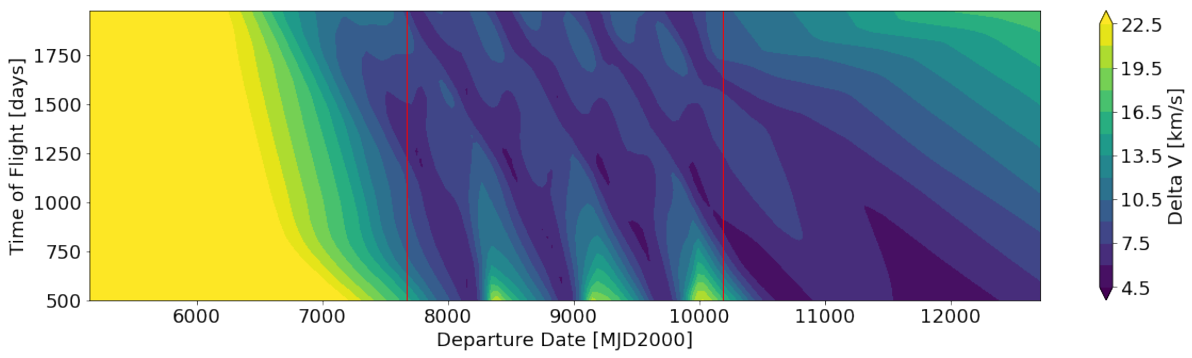


Figure 3.8: Prediction of the required ΔV budget by the ANN for departure dates outside the training range. The training range is located between the two red lines.

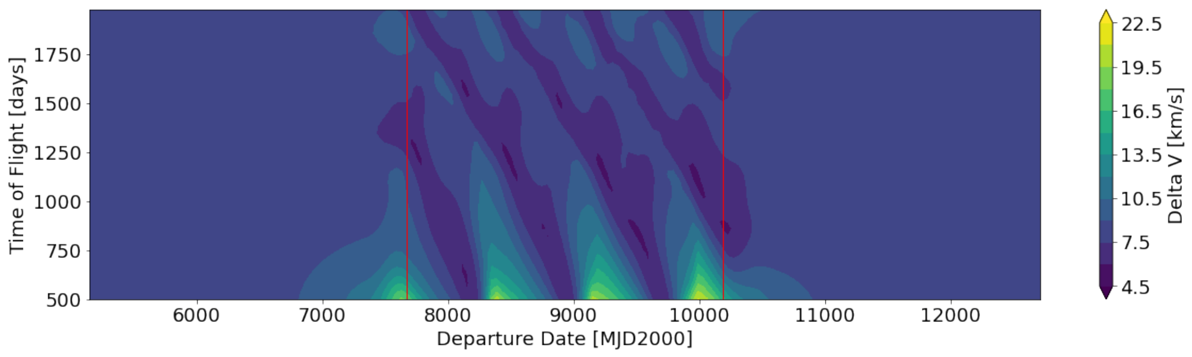


Figure 3.9: Prediction of the required ΔV budget by the GP for departure dates outside the training range. The training range is located between the two red lines.

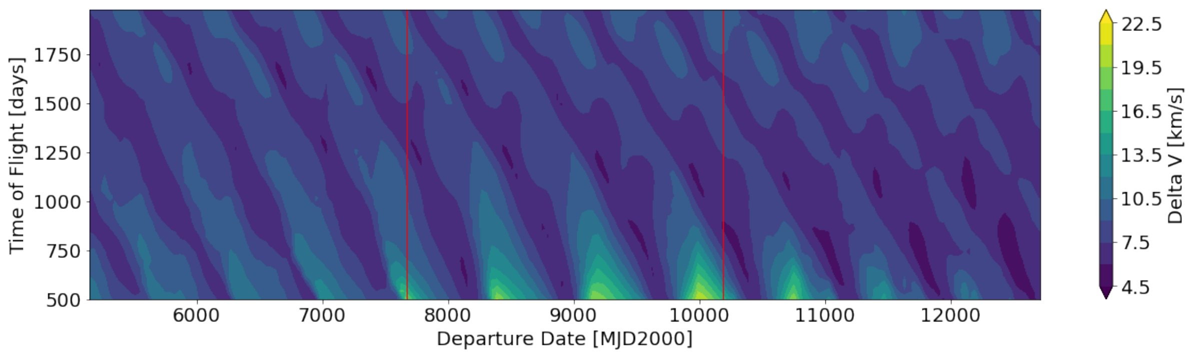


Figure 3.10: Prediction of the required ΔV budget by the ANN for departure dates outside the training range. The training range is located between the two red lines.

extended out of the training range. The trend fades out both for the ANN as the GP, but this gives some insight in what is happening. In order to have the ML-algorithm predict the ΔV budget for a trajectory to Mars with a certain departure date and duration, the state of the Earth at departure and the state of Mars at arrival is determined. These states however have a limited range, since Earth and Mars are both orbiting the Sun. Of the 14 states the ML-algorithm is trained on, 13 are mostly located within the training range. The only parameter that is truly out of the training range is the departure date. Unfortunately, this parameter is expected to be a very important feature for the prediction, since removing it decreases the accuracy of the ML-algorithms. Based on the statistics of the performance of the ML-algorithms, presented in Table 3.9, it can be concluded that the ANN performs best outside the training range.

Interestingly, the performance of the GP has decreased after adding more features. This is because statistically, it is more beneficial to predict the mean value for the whole dataset than to incorrectly

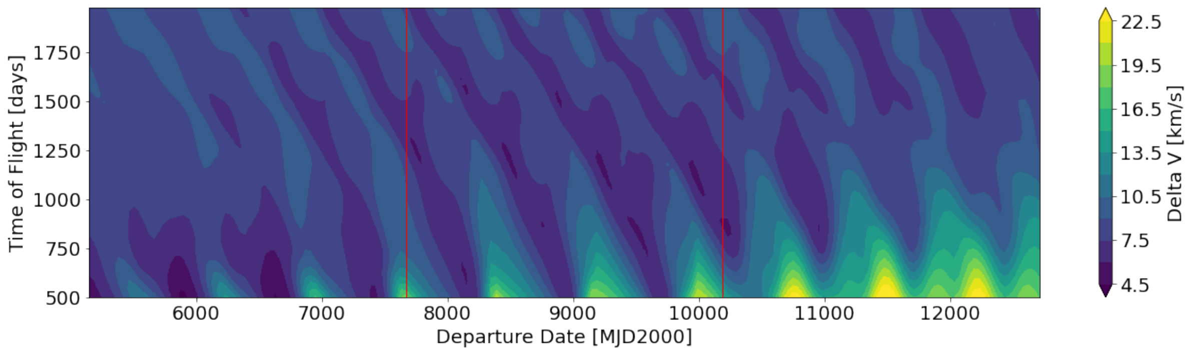


Figure 3.11: Prediction of the required ΔV budget by the GP for departure dates outside the training range. The training range is located between the two red lines.

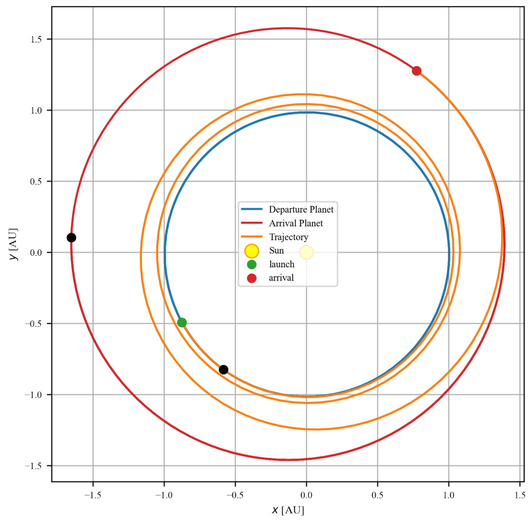
Feature Set	Set 1		Set 4	
	ANN	GP	ANN	GP
Mean of the prediction error [km/s]	5.837	0.241	-0.114	0.626
Std. of prediction error [km/s]	9.720	1.596	1.503	2.120

Table 3.9: Performance of the ML-algorithms on departure dates ranging from 01-01-2014 to 31-12-2034. Feature set 1 is the full state information, and feature set 4 consists only of the departure date and TOF.

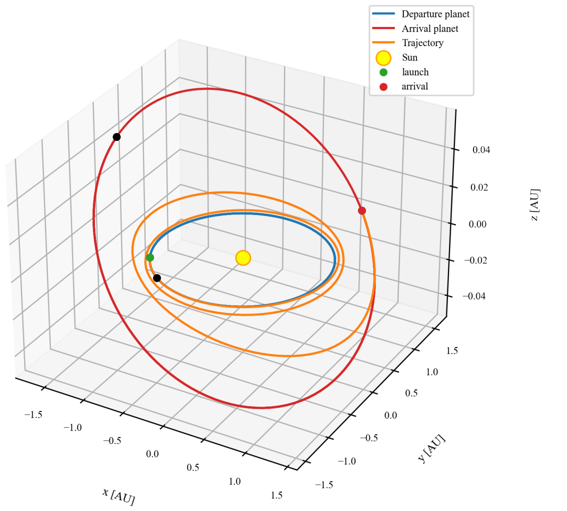
predict patterns. For finding the departure date and duration of the optimal trajectory outside the training range, this is not helpful. Although the ANN performed best outside the training range, it is still not accurate

3.5. The optimal low-thrust trajectory

The outcome of the surrogate model, consisting of the DE-algorithm evaluating a trained GP, is a trajectory with launch date 9971 MJD2000 (20-04-2027) and a duration of 1122 days, that requires a ΔV budget of 5.763 km/s. During this trajectory, 2 revolutions around the Sun are completed. The shape of the trajectory is shown in Figure 3.12.



(a) Two-dimensional view of the optimal Earth-Mars transfer.



(b) Three-dimensional view of the optimal Earth-Mars transfer. Not on scale.

Figure 3.12: Overview of the optimal Earth-Mars transfer. Images created using the tools developed by Stubbig [21].

The difference in inclination between the plane of the trajectory of the Earth and Mars is very small, so in Figure 3.12(b), the z-axis does not have the same scale as the x- and y-axis.

The acceleration profile corresponding to the trajectory is shown in Figure 3.13. The largest acceleration is along the tangential direction, which is not surprising since accelerating in this direction is the most efficient. Interestingly, the satellite does not start changing the inclination of the orbital plane immediately. Only after 200 days, effort is put in changing it. The maximum value of the total acceleration is 1.344×10^{-4} m/s^2

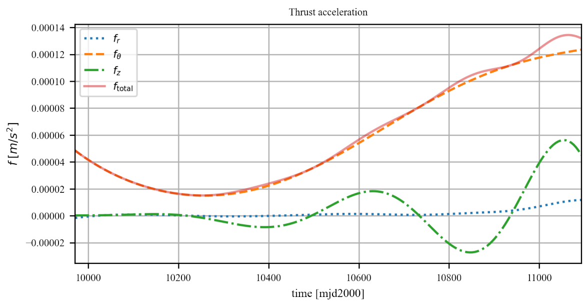


Figure 3.13: Acceleration of the satellite in the radial, tangential and axial direction, as function of time.

4 Conclusion

Multiple approaches of low-thrust satellite trajectory design incorporating machine learning have been developed already, however, they have not been thoroughly compared yet. In this thesis is shown how the ANN and GP compare in terms of accuracy of predicting the optimal transfer to Mars, where the difference comes from, and how different factors influence the prediction accuracy of each algorithm. In the paper in Chapter 2, three important conclusions are drawn: The GP provides more accurate results than the ANN, the dataset itself defines the effectiveness of the ML-algorithms trained on it, and scaling the datasets improves the results of training both ML-algorithms on the data.

Based on the additional results given in the thesis, an additional conclusion can be drawn. The ANN performs better on predicting the ΔV budget of trajectories outside the training range than the GP. This is most likely to the cyclic or constant nature of some of the training features. The phase angle, θ, between Earth and Mars repeats itself every synodic period of Mars, and the distances between Earth and the Sun, and Mars and the Sun are almost constant in relative terms. Only the departure date is not cyclic, or with very small deviations, so with an improved set of training features, the performance of the ML-algorithms outside the training range might become even better. To finalize the conclusion, the research questions posed in the introduction are answered.

4.1. Answers to the research questions

1. *How can machine learning algorithms be used for optimization of low-thrust trajectories?*
   There are multiple ways ML-algorithms can be used for optimization of low-thrust interplanetary satellite trajectories. There are two main approaches to be distinguished. The first method is to use the ML-algorithm to define the optimal control sequence. This approach is very suitable for autonomous on board steering and performing corrections. The the second approach is to perform regression analysis and use a surrogate approach to find the optimal trajectory, of which the evolutionary neurocontroller was one of the first applications. This method is very suitable for finding optimal trajectories since evaluations of the ML algorithm are quick and computationally cheap. The latter of the methods is used in this thesis.

2. *Why is a surrogate method a viable approach?*
   The justification for the surrogate approach is twofold. On one hand, it can be used to determine the optimal trajectory, on the other hand it can be used to verify that a sub-optimal result is close to the optimum. In terms of ΔV budget gained by this approach, it is shown that the best trajectory determined using the surrogate approach only required 0.02 km/s less than the best outcome of the training grid. In this problem, the gain in ΔV is very small, less than 0.5%, so taking steps of 40 days in the departure dates and time of flight is justified. However, this is only known after the surrogate approach is performed.

   Not only the accuracy of the surrogate approach is important, the required time to obtain the answer should be considered as well. A comparison between the surrogate approach and an optimization using the original fitness function has not been made, but it can be deduced that a grid search is not a suitable solution to determine the optimal trajectory with a margin of 1 day in terms of the departure date and the TOF. Designing a single low-thrust trajectory using the hodographic shaping method took on average just under 1.5 seconds. For every departure date and time of flight evaluated for the training dataset, 6 trajectories were designed (one for every value of N), of which only the best was selected for training the ML algorithm. Obtaining the dataset of 2432 trajectories, with a step size of 40 days for the departure date and TOF, took about 6 hours. Training the GP took about half an hour, and performing the optimization using the DE took half a minute on average. Verifying this result using the hodographic shaping method required generating another small grid of trajectories, for which the value of N could be estimated based on the TOF determined by the surrogate approach. Searching a grid of 121 trajectories (11x11 surrounding the outcome of the surrogate approach) required 3 minutes, so in total, obtaining the optimal trajectory took 6.5 hours. When a grid with step size 1 day would be used to find the optimal trajectory, determining all trajectories in the grid would take 1600 times (40x40) as much time, or roughly 400 days.

3. *How well do the ML-algorithms predict the outcome of the test dataset?*
The accuracy of the prediction of the ΔV budgets is measured using the standard deviation of the prediction error. The values for the standard deviation of the error of dataset 3 are as low as 0.16 km/s for the ANN and 0.07 km/s for the GP. For both the other datasets, the standard deviation is about 12 km/s, which exceeds the minima of these datasets. This renders the ML-algorithm unusable for two of the three datasets. Whether the standard deviation of the prediction error is low enough depends on the problem. In order to find the optimal trajectory using the DE-strategy, the accuracy of the ANN is not good enough. An incorrect local minimum (8486,1213) was assigned as location to search for the global minimum by some of the runs using ANNs. There were 9 points generated by the DE-algorithm in the vicinity of this local minimum, so it cannot be considered an outlier. The accuracy of the GP is good enough for this application. The correct local minimum was located as the most promising to look further for the optimal trajectory. This is impressive since the difference between the optimal trajectory in the training range and the best training sample is about 0.02 km/s. In conclusion, the accuracy of the ML-algorithm is dependent on the training dataset.

4. *How does the training data influence the accuracy of the output?*
There are multiple properties of the data that influence the accuracy of the ML-algorithms. These are mainly the smoothness of the output value of the test data, the distribution of the test data samples over the output codomain, and to lesser extent, the value of the data. To show this, the three different trajectory datasets were compared. The most accurate predictions were performed on dataset 3. There are two reasons this dataset is easiest to train on. The first reason is that the amount of jumps in the data is smallest. For dataset 1, there are large clusters of data with large jumps in between, for dataset 2 there are no clusters and many smaller jumps, and for dataset 3, there are no jumps. This resulted in a smoother porkchop plot, and more accurate predictions by the ML-algorithm. The second reason is that the distribution of the points is very suitable for accurate training. An issue that might occur for ML is the class imbalance problem. If the distribution of datapoints is skewed, the accuracy of the predictions in regions where the density of samples is low, is lower than for regions with higher density, because the ANN is not trained well enough. In the thesis is verified that this is not an issue for the prediction of the ΔV budget of the trajectories using dataset 3. Since samples with a higher ΔV value are more sparse, the standard deviation of the error in the important regions is lower than the mean values. The standard deviation of the error for the best performing ANN and GP on the whole dataset are 0.164 km/s and 0.071 km/s respectively. For the prediction of the minimum values, the standard deviations of the error are 0.032 km/s for the ANN, and 0.020 for the GP. Finally, the value of the data could be a factor. Data with higher values is harder to predict. However, scaling prevents most of the issues.

5. *How can the optimal hyperparameters for the ML-algorithm be determined?*
The way to determine the optimal hyperparameters depends on the algorithm. For GPs, the optimal hyperparameters are inferred based on the maximum likelihood. For ANNs, the optimal hyperparameters cannot be determined directly. In papers on trajectory optimization using ANNs, grid searches, random searches, and a tree structured Parzen estimators approach were used

to find them. In this thesis, the author decided to use a grid search, because not only does it give relatively good hyperparameters, it also gives insight in the effect of changing them, and what the relation between hyperparameters is.

6. *Why is there a difference between the performance of the algorithms?*
   Throughout this thesis, the GP outperformed the ANN on multiple tests. Mathematically, it is hard to show why, if not impossible. However, in terms of specifications of the algorithm, a comparison can be made. A GP is non-parametric regression algorithm, which means it follows the data perfectly, and gives an estimate of the behavior in between data points with a mean value and a probability distribution. The hyperparameters are optimized by the algorithm using the maximum likelihood, and the kernel contains information of the expected behavior of the function that is to be fitted. The hyperparameters of the ANN are not dependent on the problem. Any number of layers, nodes, or epochs can be chosen to find the best fitting regression function. Even for the activation function is shown that the ReLu function outperforms other functions most of the time [23]. The factor that improves the ANN the most consistent is the number of training samples. This is also the main difference between the two algorithms. Because the GP consists of a matrix inversion, the computational effort increases with $\mathcal{O}(n)^3$, where the computational effort of the ANN increases with $\mathcal{O}(n)$, given n the number of training samples. There is no clear boundary of training samples above which the ANN is preferred, since this is more a time issue than a numeric issue, but for dataset 3, this boundary is higher than the maximum number of training samples used. To summarize, it is not surprising that the GP outperformed the ANN on this problem, since the non-parametric nature of the GP makes it possible to perfectly simulate the training data, and the hyperparameters are determined in a more efficient way.

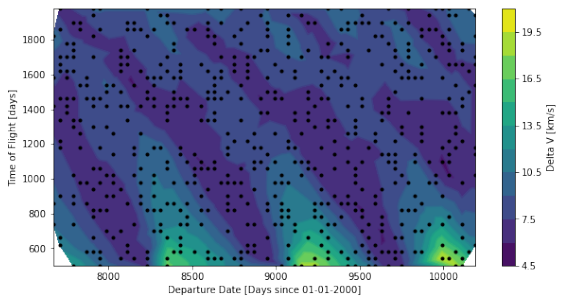
7. *How do the ML-algorithms compare with each other in terms of computational speed and accuracy?* In the paper, it is shown that the GP is more accurate and the ANN is faster. The average duration of the training of both algorithms is given in this thesis, with 28 minutes for the GP, and 10 minutes for the ANN. There is no reason to assume a significant improvement can be made by training deeper or wider ANNs. On average, the ANNs improve when more layers are added, more nodes per layer are used, when a smaller batch size is used, or when the number of epochs trained over is increased, however, for the best performing ANNs, this improvement is too small to expect that a higher accuracy can be obtained than for GPs.

8. *How do the different ML-algorithms perform outside of the training range?*
   The performance of the ANN and GP was tested outside the training range for different feature sets. There is a clear improvement in the predictions when the two algorithms are trained on parameters that repeat themselves after a period of time (such as the position and velocity of Earth and Mars). Improvement might be obtained when testing a suitable feature set with bounded limits for all features, however, since the departure date is very important for the accuracy of the ML-algorithms, such a feature set has not been found yet.

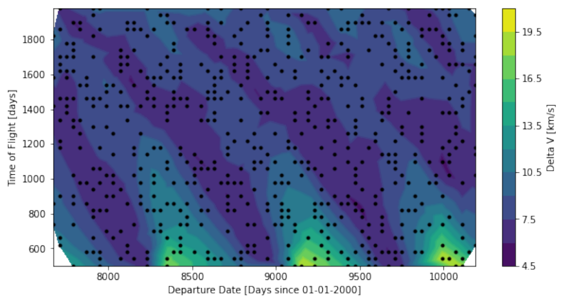
5 Recommendations

Based on the research performed in this thesis, a collection of recommendations is given to the reader and fellow students for further research.

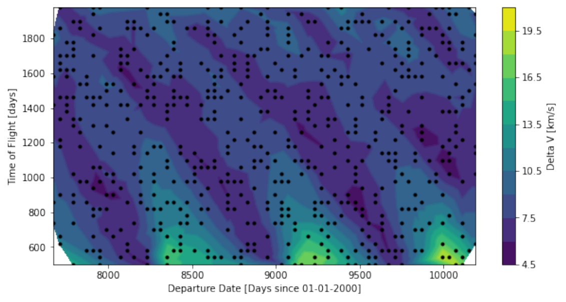
1. *Test different machine learning algorithms to learn as much about the data as possible.*
   ANNs and GPs are not the only tools to perform ML-regression. Two other algorithms that were tested as example are the support vector machine (SVM) and extreme gradient boosting (XGB). SVMs were successfully used to design low-thrust trajectories from Earth to Vesta [26]. XGB is interesting since it combines gradient boosting with decision trees, and decision-based algorithms seem very promising for combinatorial problems such as multi-flyby trajectories [8], which are more difficult compared to the problem investigated in this thesis. Furthermore, this algorithm can be used to determine the importance of the input features, which shows what features are important for the regression analysis. In Figure 5.1, the output of the different ML-algorithms on the same test dataset is shown.



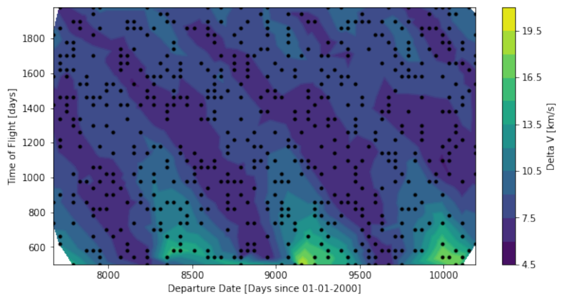
(a) Porkchop plot of the test output data of the ANN.



(b) Porkchop plot of the test output data of the GP.



(c) Porkchop plot of the test output data of the SVM.



(d) Porkchop plot of the test output data of the XGB.

Figure 5.1: Comparison of different machine learning algorithms on the same test dataset. The black dots depict the test data samples.

As can be seen in Figure 5.1, all four ML-algorithms are capable of predicting the test data, although the accuracy differs between the algorithms. Table 5.1 shows that the best performance is obtained by the GPR and the ANN, which is not surprising since the hyperparameters of these algorithms were tuned in this thesis. Between the XGB and the SVM, not much can be said, except that the general trend of the test data is estimated reasonably well. When the hyperparameters for these algorithms are optimized as well, the performance can be assessed more accurately.

ML-algorithms	**ANN**	**GP**	**SVM**	**XGB**
Mean of the prediction error [km/s]	0.011	0.005	-0.002	-0.010
Std. of the prediction error [km/s]	0.155	0.075	0.338	0.371

Table 5.1: Comparison of the performance of different machine learning algorithms on the dataset shown in Figures 5.1.

2. *Optimize the feature set.*
In this thesis, it was shown that the features the ML-algorithms are trained on are very important. They largely influence the accuracy of the output of both the ANN as the GP. The process of optimizing the feature set used to train ML-algorithms on is called feature engineering.

The first step in feature engineering is to come up with a large pool of features that might have an influence on the outcome of the ML-algorithm. This is done by reconsidering features selected for training and design them as good as possible. When taking a closer look at the problem tackled in this thesis, the arrival date might be a convenient feature. Replacing the departure date with the arrival date for the porkchop plot of the trajectories of dataset 3 results in Figure 5.2. As can

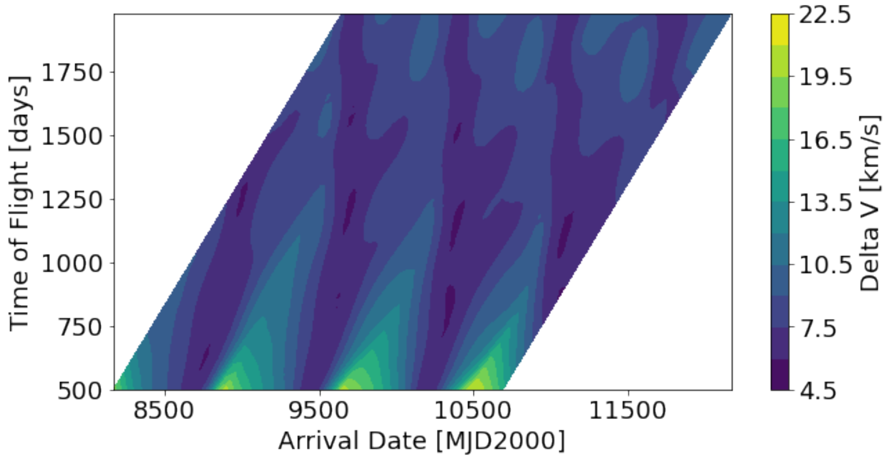


Figure 5.2: Porkchop plot of dataset 3, using the arrival date instead of the departure date.

be seen, the regions with low or high ΔV are almost only dependent on the arrival date. The pattern of trajectories that require a low ΔV budget repeats itself every 700 arrival days, about the sidereal period of Mars with respect to Earth. Since this pattern is repeating as well, substituting the departure or arrival date with the date since the last passing of Mars, i.e. the moment the phase angle between Earth and Mars equals 0, might improve the performance both within as outside the training range of departure dates. This information is already available in the data, since the polar angle of both Mars and the Earth is used for inference.

Not only new features should be investigated, also, the way features are described should be considered. Instead of training on θ, it might be beneficial to train on both the sine as the cosine of this angle, since the angle jumps from 2π to 0 every rotation, while the sine and cosine remain smooth. Due to the nature of ML-algorithms, adding more features does only influence the outcome if there is a trend between the input and the output, and when the features are not correlated to other features.

The second step is to find the feature set that performs best. The approach depends on the algorithm used for inference. For ANNs and GPs, the approach is to test the full set designed in step 1, and use the accuracy of this set as benchmark. By leaving features out one by one, the important features can be identified, and the feature set can be reduced to decrease the time required to train the ML-algorithm. For the XGB algorithm, the importance of every feature can be determined. The feature importance is both dependent on the training data as on the algorithm, so small changes in the value of the feature importance are seen every new instance the XGB algorithm is trained. However, the general trends are clear: the most important features for the XGB algorithm are the time of flight, the departure date, and some of the state variables of Mars. The radial velocity of both Mars and the Earth does not contribute to the prediction of the XGB-algorithm at all. Even though the feature importances of the XGB-algorithm do not translate to the ANN and GP one to one, it is expected that the time of flight is indeed a very important feature, and the radial velocity of Earth and Mars are not the most important features for the inference.

3. *Compile a dataset of different missions.*
   In this thesis, different ML-algorithms were tested for a mission to Mars. Examples of test cases described in other theses are the use of:

   - Both GPR and classification to find trajectories to Mars, Ceres and from Mars back to Earth [2].
   - Online training of ANNs for a mission to both Ceres and Vesta, and a mission to visit multiple asteroids [21].
   - Extreme Learning Machines for multiple-flyby missions to Vesta, Jupiter and Mercury [4].

   To obtain a truly valuable tool, these mission could all collected and combined in the same dataset, some more inward missions are added, and one ML-algorithm is trained on the whole collection. In theory this enables a single ML-algorithm to design many different low-thrust trajectories in the Solar system.

4. *Improve the details of the selected approach.*
   Within the thesis, there are many small points that can be improved upon. There are various reasons they were not tackled in this thesis. An overview of possibilities is:

   (a) *Repeat experiment with different hyperparameters for each layer of the ANN*
   A combination of activation functions, or layers of different width might influence the result.

   (b) *Test the Mean squared error as loss function for the ANN*
   There is a lot of similarity between the standard deviation of the prediction error and the mean squared error. When aiming for the lowest standard deviation, a loss function that aims at exactly that might benefit the results.

   (c) *Generate trajectories using a more accurate method*
   Indirect methods, using Pontryagin's minimum principle for instance, that define the optimal control sequence, are more accurate than shaping methods.

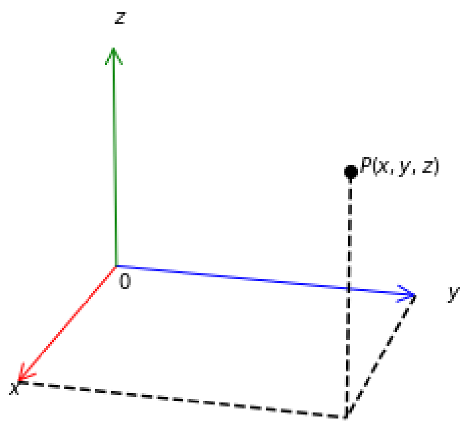
   (d) *Apply Latin hypercube sampling for selecting the training dataset*
   Sampling was performed randomly. Especially for GPs, where the number of training samples can not exceed a couple of thousand, making sure the distribution of training samples over each feature is uniform will improve the performance of the ML-algorithm.

A Shaping Methods

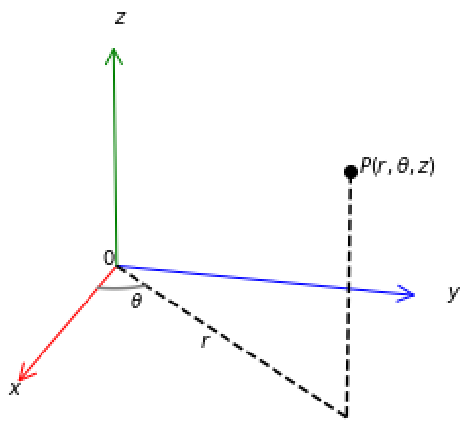
The datasets of low-thrust trajectories used in this thesis were designed using shaping methods: attractive, low fidelity, direct methods for trajectory design. The advantage over indirect methods is that shaping methods are easier to implement, the radius of conversion is larger and they have a reduced problem size. The downside is that the accuracy is lower [18]. The idea behind shaping methods is to shape the trajectory according to a predefined shape, and match the parameters of the equations defining this shape such that all boundary conditions are met. Before the two shaping methods used in this thesis are described in more details, some basic concepts and physics need to be defined.

A.1. Coordinate Frame Transformations

Throughout this thesis, the heliocentric reference frame is used, since trajectories from Earth to Mars orbit the Sun. To define these trajectories, two different coordinate systems are used: the Cartesian coordinate system and cylindrical coordinate system. In the Cartesian coordinate system, the axes are linear and perpendicular. These are called x-, y- and z-axis and the configuration is given in Figure A.1(a). The x-, y- and z-coordinate are defined along these axis and can be ranged in the domain $(-\infty, \infty)$.



(a) Cartesian coordinate system.



(b) Cylindrical coordinate system.

Figure A.1: The two coordinate systems used in this thesis.

The coordinates in the cylindrical coordinate system are called r, θ and z. r is the radial distance from the origin, θ is the angle with respect to the x-axis, counterclockwise defined positive, and z is the same as in the Cartesian coordinate system. This system is shown in Figure A.1(b). The coordinates can be ranged in the domains $[0,\infty), [0, 2\pi]$, and $(-\infty,\infty)$. When only 2 dimensions are used, these coordinates are called polar coordinates and the system only consists of the radius, r, and the polar angle, θ. The transformation from the Cartesian coordinate frame to cylindrical coordinates is performed using the following three equations:

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

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

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

The transformation from cylindrical coordinates back to Cartesian coordinates is the following:

$$x = r\cos\theta \tag{A.4}$$

$$y = r\sin\theta \tag{A.5}$$

$$z = z\,. \tag{A.6}$$

A.2. Equations of motion

The equations of motion describe the position, velocity and acceleration of the satellite during the trajectory. In cylindrical coordinates, they are defined the following [5]:

$$f_r = \ddot{r} - r\dot{\theta}^2 + \frac{\mu}{s^3}r \tag{A.7}$$

$$f_\theta = r\ddot{\theta} + 2\dot{r}\dot{\theta} \tag{A.8}$$

$$f_z = \ddot{z} + \frac{\mu}{s^2}z\,, \tag{A.9}$$

where $s = \sqrt{r^2 + z^2}$.

A.3. Exposin Shaping

The exposin shaping method is a 2-dimensional shaping method, developed by Petropoulos and Longuski [14]. The implied shape for the exposin shaping method is the exponential sinusoid, which is defined by

$$r = k_0 e^{k_1 \sin(k_2\theta+\phi)}, \tag{A.10}$$

using polar coordinates r and θ. In this equation, tangential thrust is assumed. The equations of motion that need to be satisfied in this equation are Eqns. (A.7) and (A.8). Given the angles and vectors in Figure A.2, the equations of motion can be written as:

$$\ddot{r} - r\dot{\theta}^2 + \frac{\mu}{r^2} = f\sin\alpha \tag{A.11}$$

$$r\ddot{\theta} + 2\dot{r}\dot{\theta} = f\cos\alpha. \tag{A.12}$$

Because of the assumption of tangential thrust, the thrust angle α is given by:

$$\alpha = \gamma + n\pi, \tag{A.13}$$

where n is 0 for acceleration, and 1 for deceleration. $\tan(\gamma)$ can be defined using:

$$\tan\gamma = \frac{1}{r}\frac{\mathrm{d}r}{\mathrm{d}\theta} = k_1k_2\cos(k_2\theta+\phi). \tag{A.14}$$

Filling in the definition for the exposin shape for the radius gives the following terms for the rotation rate $\dot{\theta}$ and the normalized thrust acceleration $a = \frac{f}{\mu/r^2}$:

$$\dot{\theta}^2 = \left(\frac{\mu}{r^3}\right)\frac{1}{\tan^2\gamma + k_1k_2^2\sin(k_2\theta+\phi)+1} \tag{A.15}$$

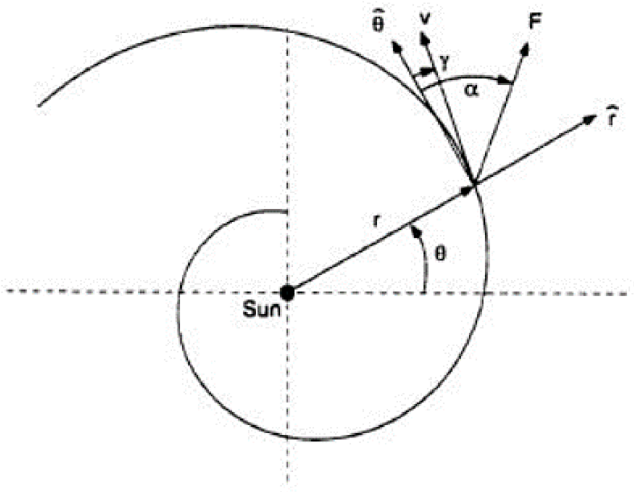


Figure A.2: Definition of variables by Petropoulos [13].

$$a = \frac{(-1)^n \tan\gamma}{2\cos\gamma}\left[\frac{1}{\tan^2\gamma + k_1k_2^2\sin(k_2\theta+\phi)+1} - \frac{k_2^2\,(1-2k_1\sin(k_2\theta+\phi))}{\left(\tan^2\gamma + k_1k_2^2\sin(k_2\theta+\phi)+1\right)^2}\right]. \quad \text{(A.16)}$$

To prevent $\dot{\theta}^2 < 0$, the condition $|k_1k_2^2| < 1$ is applied.

Based on this approach, Izzo applied Lambert targeting to allow the exposin shaping method to be used in a global optimization scheme [7]. The chosen approach is the following. Assuming θ at departure equals zero, matches the coordinate frame with the departure position. The initial and final position can then be written as:

$$r_1 = k_0 e^{k_1\sin\phi} \quad \text{(A.17)}$$

$$r_2 = k_0 e^{k_1\sin(k_2\bar{\theta}+\phi)}, \quad \text{(A.18)}$$

where $\bar{\theta} = \psi + 2\pi N$ to allow for N revolutions around the center body. Assuming values for k_2, N and γ_1, leaves three unknowns, k_0, k_1 and ϕ. Finding k_1 is done in two steps: at first the magnitude is determined using

$$k_1^2 = \left(\frac{\ln\left(\frac{r_1}{r_2}\right) + \frac{\tan\gamma_1}{k_2}\sin k_2\bar{\theta}}{1-\cos k_2\bar{\theta}}\right)^2 + \frac{\tan^2\gamma_1}{k_2^2}, \quad \text{(A.19)}$$

and then the sign using

$$\text{sign}(k_1) = \text{sign}\left(\ln\left(\frac{r_1}{r_2}\right) + \frac{\tan\gamma_1}{k_2}\sin k_2\bar{\theta}\right). \quad \text{(A.20)}$$

Now k_1 is determined, finding k_0 and ϕ is straightforward:

$$\phi = \arccos\left(\frac{\tan\gamma_1}{k_1k_2}\right) \quad \text{(A.21)}$$

$$k_0 = \frac{r_1}{e^{k_1\sin\phi}}. \quad \text{(A.22)}$$

Based on these equations, it can be shown that

$$\forall k_2 : \exists S_{K_2}[r_1, r_2, \psi, N]. \quad \text{(A.23)}$$

However, this is without applying the dynamic feasibility condition $|k_1 k_2^2| < 1$. Substituting Eqn. (A.19) into this condition gives

$$\left(\frac{k_2^2 \ln\left(\frac{r_1}{r_2}\right) + k_2 \tan\gamma_1 \sin k_2\bar{\theta}}{1 - \cos k_2\bar{\theta}} \right)^2 + k_2^2 \tan^2\gamma_1 < 1. \tag{A.24}$$

From this equation, the boundaries for γ_1 that give feasible trajectories can be determined:

$$\tan\gamma_{1_{min,max}} = \frac{k_2}{2}\left[-\ln\frac{r_1}{r_2}\cot\frac{k_2\bar{\theta}}{2} \pm \sqrt{\frac{2(1-\cos k_2\bar{\theta})}{k_2^4} - \ln^2\frac{r_1}{r_2}} \right]. \tag{A.25}$$

These trajectories satisfy the dynamic feasibility condition given the term below the square root is positive, however, they do not meet the required time of flight (TOF) condition yet. The condition on the required time of flight makes certain that the arrival body is present at the predetermined phase angle ψ. The TOF can be determined using Eqn.(A.15):

$$TOF = \int_{t_0}^{t_f} \mathrm{d}t = \int_{\theta_0}^{\theta_f} \frac{\mathrm{d}t}{\mathrm{d}\theta}\,\mathrm{d}\theta = \int_{\theta_0}^{\theta_f} \sqrt{\frac{r^3}{\mu}\tan^2\gamma + k_1 k_2^2 \sin(k_2\theta + \phi) + 1}\,\mathrm{d}\theta. \tag{A.26}$$

The value of γ_1 can be varied within the allowed range in order to make the TOF determined using Eqn. (A.26) match the required TOF.

Since the exposin shaping method is only 2-dimensional, additional plane changes need to be defined at the beginning and end of the low-thrust trajectory. The same approach is used as by Bouwman [2], modelling the plane change after the low-thrust trajectory is calculated, using impulsive shots.

A.4. Hodographic Shaping

The hodographic shaping method was developed by Gondelach and Noomen [5]. The main difference between this shaping method and most others are that most shaping algorithms try to fit the trajectory to reach the destination, but the hodographic shaping method uses the velocity of the satellite and the mission target to match the trajectory. The idea behind the method is to connect the hodograph of the departure body with the hodograph of the arrival body, and apply boundary conditions such that the trajectory is fulfilling all conditions. An intuitive way of displaying a trajectory is by showing the position of the satellite in a Cartesian reference frame (Figures 3.12(a) and 3.12(b)), or using cylindrical coordinates r, θ and z (Section A.1). A hodograph gives the velocity $\dot{r}$ and $r\dot{\theta}$ instead of the position. The hodograph of the Earth, Mars and the optimal low-thrust transfer between the two is shown in Figure A.3

As can be seen in Figure A.3, the hodograph shows what the velocity of a satellite should be to go from the trajectory of the departure body to the trajectory of the arrival body. In general, it does not show where the bodies are, so in order to make sure the departure and arrival body are at the desired positions, boundary conditions are required. Furthermore, there are infinitely many lines connecting the two hodographs, so these need to be constrained as well. This is done using the velocity function, V. The velocity function consists of multiple base functions, v, which should be relatively easily be integrated. They are multiplied by a coefficient, c, and added such that

$$V(t) = \sum_{i=1}^{n} c_i v_i(t). \tag{A.27}$$

The number of base functions should at least equal the number of boundary conditions. Suitable choices for base functions are listed by Gondelach [5]. The position and acceleration of the satellite can now be defined as function of the velocity functions. The position is obtained by integrating the

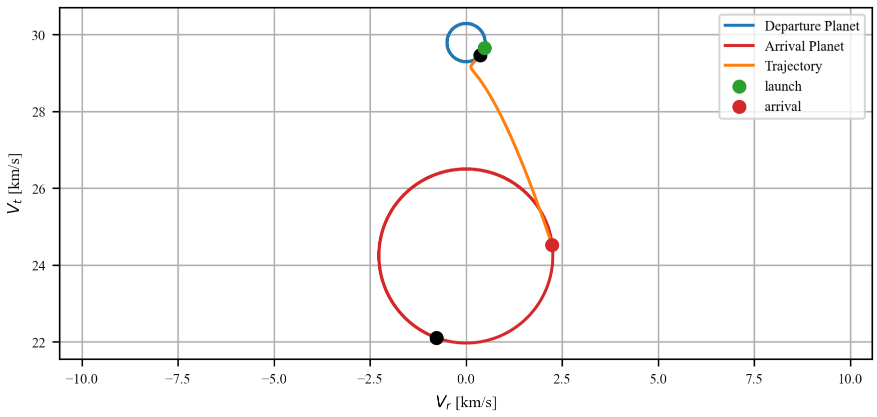


Figure A.3: Velocity hodograph of the optimal trajectory designed using the hodographic shaping method. Method implemented by Stubbig [21].

velocity functions, and the acceleration by differentiating them, as the following equations:

$$r(t) = r_0 + \int_0^{t_f} V_r \,\mathrm{d}t, \qquad \theta(t) = \theta_0 + \int_0^{t_f} V_\theta \,\mathrm{d}t, \qquad z(t) = z_0 + \int_0^{t_f} V_z \,\mathrm{d}t, \tag{A.28}$$

$$\dot{r}(t) = \dot{V}_r, \qquad \ddot{\theta}(t) = \frac{\dot{V}_\theta}{r} - \frac{V_\theta V_r}{r^2}, \qquad \ddot{z}(t) = \dot{V}_z. \tag{A.29}$$

For a three-dimensional problem, the following boundary conditions apply:

$$V_r(0) = V_{r,0}, \qquad V_r(t_f) = V_{r,f}, \qquad \int_0^{t_f} V_r \,\mathrm{d}t = r_f - r_0 \tag{A.30}$$

$$V_\theta(0) = V_{\theta,0}, \qquad V_r(\theta_f) = V_{\theta,f}, \qquad \int_0^{t_f} \frac{V_\theta}{r} \,\mathrm{d}t = \theta_f = \psi + 2\pi N \tag{A.31}$$

$$V_z(0) = V_{z,0}, \qquad V_z(t_f) = V_{z,f}, \qquad \int_0^{t_f} V_z \,\mathrm{d}t = z_f - z_0. \tag{A.32}$$

In this set of equations, the first condition is to make sure the initial velocity of the trajectory matches the velocity of the departure body. The second column does the same for the arrival body. The third column is to make sure the distance covered by the trajectory is indeed from the departure body to the arrival body. For the displacement condition in tangential direction in Eqn. (A.31), the same conditions can be recognized as for the exposin shaping method: $\theta_0 = 0$ and N defines the number of revolutions around the Sun. For the velocities in all three directions, the following system of equations is solved to make sure all boundary conditions are met:

$$\begin{bmatrix} v_1(0) & v_2(0) & v_3(0) \\ v_1(t_f) & v_2(t_f) & v_1(t_f) \\ \int_0^{t_f} v_1 \,\mathrm{d}t & \int_0^{t_f} v_2 \,\mathrm{d}t & \int_0^{t_f} v_3 \,\mathrm{d}t \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} V_0 - \sum_{i=4}^{n} c_i v_i(0) \\ V_f - \sum_{i=4}^{n} c_i v_i(t_f) \\ \int_0^{t_f} V \,\mathrm{d}t - \sum_{i=4}^{n} \int_0^{t_f} c_i v_i \,\mathrm{d}t \end{bmatrix} \tag{A.33}$$

When suitable constants and velocity functions are determined, the velocities and accelerations can be entered into the equations of motion (Eqns. (A.7), (A.8) and (A.9)). The ΔV budget can be obtained by integrating the resulting accelerations (f_r, f_θ and f_z) over the TOF:

$$\Delta V = \int_0^{t_f} f \, \mathrm{d}t. \tag{A.34}$$

B
Machine Learning Algorithms

In the paper in Chapter 2 and Chapter 3, four different machine learning algorithms are shown. These algorithms have in common that they perform supervised learning, which makes regression analysis possible. Both for supervised as unsupervised learning, an algorithm is trained on a set of training samples $\mathbf{x}$. The difference is that for supervised learning, the output y is known. The outputs $\mathbf{y}$ are also called the observations. The goal for the regression analysis is to find a function $f(\mathbf{x})$ that matches the observations, such that the value of new observations can be estimated with the regression function. In terms of conditional probability, this can be written as [12]:

$$f(\mathbf{x}_*) = p(y_*|\mathbf{x}, \mathbf{y}, \mathbf{x}_*). \tag{B.1}$$

So in order to find the best estimate of the new observation, the function of new input $\mathbf{x}^*$ should equal the maximum likelihood of the new output y^* given previous input samples, $\mathbf{x}$, observations, $\mathbf{y}$, and the new input values of which the function is to be estimated.

B.1. Artificial Neural Networks

The artificial neural networks is a ML-algorithm that consists of multiple connected layers existing out of nodes. When trained, these layers are capable of predicting the right outcome for every input, within bounds of accuracy, which was shown by Hornik, by means of the universal approximation theorem [6].

In the first step, the input vector is multiplied with the first layer of weights (nodes), and a bias is added:

$$\mathbf{z} = W\mathbf{x} + \mathbf{b}. \tag{B.2}$$

This output is activated with an activation function,

$$\mathbf{a} = \sigma(\mathbf{z}), \tag{B.3}$$

and the activated output serves as input for the subsequent layer in case of a feedforward ANN. The output of the final layer is not activated for regression analysis. When performing the regression, this is enough to obtain the answer. However, before the ANN can give an accurate answer, it requires training, also called backpropagation. When the ANN is trained, not only the input is known, but also the output. The difference between the predicted output and the desired output is assigned a value by means of the loss function. For the MAE, the loss function for a single sample is

$$L = |y_{train} - y_{pred}|. \tag{B.4}$$

The target is to reduce the mean of the losses for the whole training dataset as much as possible. To this end, the derivative of the cost function with respect to the weights and biases is determined.

$$\frac{\mathrm{d}L}{\mathrm{d}\mathbf{b}} = \boldsymbol{\delta}, \quad \frac{\mathrm{d}L}{\mathrm{d}W} = \mathbf{a_{-1}}\boldsymbol{\delta}, \tag{B.5}$$

where $\mathbf{a_{-1}}$ is the activated output of the previous layer, and $\boldsymbol{\delta}$ is defined by

$$\boldsymbol{\delta} = \nabla_a L \odot \frac{\mathrm{d}\sigma(\mathbf{z})}{\mathrm{d}\mathbf{z}}. \tag{B.6}$$

In this equation, $\odot$ is the Hadamard product. The derivative of the weights and biases in the previous layers can be determined similarly by multiplying the derivatives. The derivative of the weights in the first layer can be traced through the different layers to the loss function, hence the name backpropagation. During training, the weights and biases are changed by multiplying the negative derivative of the cost function with the learning rate, η, and possibly adding a momentum term, v. The learning rate defines by how much the weight or bias is changed, and the momentum function increases this value when the sign of the derivative remains the same over a couple of epochs, and decreases it when the sign changes:

$$v \to v' = \mu v - \eta \nabla C \tag{B.7}$$
$$W \to W' = W + v. \tag{B.8}$$

The momentum term both increases the learning speed, as making more accurate predictions possible by effectively decreasing the learning rate when the loss value is decreased. The downside of adding momentum is that a new hyperparameter, μ, is introduced.

B.2. Gaussian Processes

The Gaussian process (GP) is the other ML algorithm used in the paper in Chapter 2. The Gaussian process (GP) is a non-parametric inference algorithm. It can be used for classification and regression analysis, the latter also called Kriging. Rasmussen states: ”A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution” [15]. The probability density function of the multivariate Gaussian distribution is defined by

$$N(\mu, K) = \frac{1}{\sqrt{(2\pi)^n |K|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T K^{-1} (\mathbf{x}-\boldsymbol{\mu})}, \tag{B.9}$$

where μ is the mean value and K the covariance matrix. Similarly. the GP is defined using a mean function and kernel function:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right). \tag{B.10}$$

In this equation, $\mathbf{x}$ and $\mathbf{x}'$ are both samples in the input domain, so when $\mathbf{x}' = \mathbf{x}$, $k(\mathbf{x}, \mathbf{x}')$ gives the variance of $\mathbf{x}$. The mean function of the GP is usually assumed to be 0, since the GP is flexible enough to correct for this when training [15]. The kernel functions combined form the covariance matrix:

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}') \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}', \mathbf{x}_1) & k(\mathbf{x}', \mathbf{x}_2) & \cdots & k(\mathbf{x}', \mathbf{x}') \end{bmatrix}. \tag{B.11}$$

When training the GP, the covariance matrix is selected such that the training output $\mathbf{y}$ is matched as good as possible. In order to do that, the hyperparameters of the kernel functions need to be optimized. The optimal hyperparameters are selected when the best a posteriori estimate occurs, i.e. at the maximum of $p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$. Using Bayes' theorem and the marginal likelihood, it can be shown that this is equal to maximizing $\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$:

$$p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{x})}. \tag{B.12}$$

Both $p(\boldsymbol{\theta})$ and $p(\mathbf{y}|\mathbf{x})$ are not dependent on $\boldsymbol{\theta}$. Since for $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$, a multivariate Gaussian distribution ($N(\mathbf{0}, K)$) was assumed by the definition of the GP, maximizing the a posteriori gives

$$\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}' K^{-1} \mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi, \tag{B.13}$$

and the maximum value for this equation can be determined using an optimization algorithm.

In order to perform inference on unseen data, two other matrices have to be generated,

$$K_* = \left[k(\mathbf{x}_*, \mathbf{x_1}), k(\mathbf{x}_*, \mathbf{x_2}), \cdots, k(\mathbf{x}_*, \mathbf{x}')\right] \qquad \text{and} \qquad K_{**} = [k(\mathbf{x}_*, \mathbf{x}_*)]\,, \tag{B.14}$$

which together form one large covariance matrix. The joint distribution of the training data output, $\mathbf{y}$, and the predicted output of the new input, $\mathbf{y}_*$, can be defined as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{x}) \\ \mu(\mathbf{x}_*) \end{bmatrix}, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right). \tag{B.15}$$

From this equation, the conditional probability $p(\mathbf{y}_*|\mathbf{y})$ can be computed, which gives an estimate for the output value of $\mathbf{x}_*$:

$$\mathbf{y}_*|\mathbf{y} \sim \mathcal{N}\left(K_* K^{-1}\mathbf{y}, K_{**} - K_* K^{-1} K_*^T\right). \tag{B.16}$$

Proof for this step can for example be found in Section 4.3.4 of [12].

C Differential Evolution

Differential evolution is an evolutionary algorithm devised by Storn and Price in 1995 [20]. The algorithm is commonly used for optimization in the design of satellite trajectories [18]. DE utilizes collections of vectors, $\mathbf{x}$, to find the best solution in the search space. These are called generations, and every generations consists of NP vectors. The DE-algorithm consists of three steps in order to perform the optimization: mutation, crossover and selection.

C.1. The algorithm

In the mutation process, the vector that will create diversity for next generation are obtained. It is generated by adding random perturbations to a vector in the current generations. The perturbations are linear combinations of two random vectors in this generation, $\mathbf{y}$ and $\mathbf{z}$. Given $\mathbf{x}$ a vector in the current generation, the mutated vector is

$$\mathbf{v} = \mathbf{x} + F(\mathbf{y} - \mathbf{z}). \tag{C.1}$$

In this equation, F is the scaling factor that defines the rate at which the population evolves, and it is ranged between 0 and 2. For every vector $\mathbf{v}'$, other vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are used.

In the crossover process, the new candidate vector for the subsequent generation is obtained. For the crossover process, a vector of uniform distributed random numbers between 0 and 1 is generated, with the same length as the input vectors. Furthermore, a random integer, J, between 1 and the number of features is generated. The vector with random numbers defines whether a feature for the new vector, $\mathbf{u}$, comes from the original vector $\mathbf{x}$ or the mutated vector $\mathbf{v}$. When the random number is smaller than the crossover factor, CR, the mutated value is picked. When the random number is larger than CR, the value of the vector $\mathbf{x}$ is crossed over to the next generation. The random integer will define what feature of $\mathbf{v}$ will be selected irrespective of the crossover factor, to guarantee mutation. So for every feature, j, goes

$$u_j = \begin{cases} v_j & \text{if } rand(j) \leq CR \text{ or } j = J \\ x_j & \text{if } rand(j) \geq CR \text{ and } j \neq J, \end{cases} \tag{C.2}$$

and together, all scalars u_j make up $\mathbf{u}$, the candidate vector for the next generation.

In the selection process, the function output of parent $\mathbf{x}$ is compared to the function output of $\mathbf{u}$, and the best performing vector is selected for the next generation.

C.2. Optimization strategies

There are many different variations to the DE algorithm described in the section above. The general notation of the different strategies is [20]

$$DE/x/y/z.$$

In this notation, x specifies what vector is chosen for the mutation. It can either be a ”rand” for a random vector, or ”best” when the vector is chosen with the best function evaluation. y is the number of difference vectors used, and z describes the crossover scheme. The algorithm described in the example above is called DE/rand/1/bin, and is the example case given by the developers of the algorithm. The algorithm used in this thesis is DE/best/1/bin, so the only difference is the vector selected for offspring.

List of References

This list of references lists all the references used in the thesis, except for the references only used in the paper in Chapter 2. All references used in this paper are included in the list of references of the paper.

[1] C. Ampatzis and D. Izzo. Machine learning techniques for approximation of objective functions in trajectory optimisation. In *Proceedings of the ijcai-09 workshop on artificial intelligence in space*, pages 1–6, 2009.

[2] L. Bouwman, Y Liu, and K. J. Cowan. Gaussian process models for preliminary low-thrust trajectory optimization. In Edit Islam I. Hussein, Kenneth R. Horneman, Christopher Scott, and Brian W. Hansen, editors, *Proceedings of the AAS/AIAA Astrodynamics Conference Aug. 11-15, 2019, Portland*, Advances in the Astronautical Sciences Series. AAS, 2020. ISBN 978-0-87703-665-4. 2019 AAS/AIAA Astrodynamics Specialist Conference ; Conference date: 11-08-2019 Through 15-08-2019.

[3] Bernd Dachwald. Evolutionary neurocontrol: a smart method for global optimization of low-thrust trajectories. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 5405, 2004.

[4] P. Gómez Pérez, Y. Liu, and K. J. Cowan. *Global optimization of low-thrust interplanetary trajectories using a machine learning surrogate*. Master thesis, Delft University of Technology, 2020.

[5] D. J. Gondelach and R. Noomen. Hodographic-shaping method for low-thrust interplanetary trajectory design. *Journal of Spacecraft and Rockets*, 52(3):728–738, 2015. ISSN 0022-4650.

[6] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. ISSN 0893-6080.

[7] D. Izzo. Lambert's problem for exponential sinusoids. *Journal of guidance, control, and dynamics*, 29(5):1242–1245, 2006. ISSN 0731-5090.

[8] D. Izzo, M. Märtens, and B. Pan. A survey on artificial intelligence trends in spacecraft guidance dynamics and control. *arXiv preprint arXiv:1812.02948*, 2018.

[9] D. Izzo, C. I. Sprague, and D. V. Tailor. *Machine learning and evolutionary techniques in interplanetary trajectory design*, pages 191–210. Springer, 2018.

[10] H. Li, S. Chen, D. Izzo, and H. Baoyin. Deep networks as approximators of optimal low-thrust and multi-impulse cost in multitarget missions. *Acta Astronautica*, 2019. ISSN 0094-5765.

[11] A. Mereta, D. Izzo, and A. Wittig. Machine learning of optimal low-thrust transfers between near-earth objects. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 543–553. Springer, 2017.

[12] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. ISBN 0262304325.

[13] A. E. Petropoulos. *A shape-based approach to automated, low-thrust, gravity -assist trajectory design*. Ph.d. dissertation, Purdue University, 2001.

[14] A. E. Petropoulos and J. M. Longuski. Shape-based algorithm for automated design of low-thrust gravity-assist trajectories. *AAS/AIAA Astrodynamics Specialists Conference*, 41:10, 2004.

[15] C. E. Rasmussen. *Gaussian Processes in Machine Learning*. the MIT Press, Cambridge, 2006.

[16] A. Rubinsztejn, R. Sood, and F. E. Laipert. Neural network based optimal control: Resilience to missed thrust events for long duration transfers. In *2019 Astrodynamics Specialists Conference*, 2019.

[17] H. Shang and Y. Liu. Assessing accessibility of main-belt asteroids based on gaussian process regression. *Journal of Guidance, Control, and Dynamics*, 40(5):1144–1154, 2017. ISSN 0731-5090.

[18] A. Shirazi, J. Ceberio, and J. A. Lozano. Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions. *Progress in Aerospace Sciences*, 2018. ISSN 0376-0421.

[19] C. I. Sprague, D. Izzo, and P. Ögren. Learning a family of optimal state feedback controllers. *arXiv preprint arXiv:1902.10139*, 2019.

[20] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997. ISSN 0925-5001.

[21] L. Stubbig and K. J. Cowan. *Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories*. Master thesis, Delft University of Technology, 2019.

[22] L. Stubbig and K. J. Cowan. Improving the evolutionary optimization of interplanetary low-thrust trajectories using an artificial neural network surrogate model. 2020. 2020 AAS/AIAA Astrodynamics Specialist Conference ; Conference date: 9-08-2020 Through 12-08-2020.

[23] C. Sánchez-Sánchez and D. Izzo. Real-time optimal control via deep neural networks: Study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41(5):1122–1135, 2018. ISSN 0731-5090 1533-3884. doi: 10.2514/1.G002357.

[24] C. Sánchez-Sánchez, D. Izzo, and D. Hennes. Optimal real-time landing using deep networks. In *Proceedings of the Sixth International Conference on Astrodynamics Tools and Techniques, ICATT*, volume 12, pages 2493–2537, 2016.

[25] L. Torgo, P. Branco, R. P. Ribeiro, and B. Pfahringer. Resampling strategies for regression. *Expert Systems*, 32(3):465–476, 2015. ISSN 0266-4720.

[26] L. Xu, H. Shang, and X. Qin. A rapid estimation for interplanetary low-thrust trajectories using support vector regression. In *IOP Conference Series: Materials Science and Engineering*, volume 449. IOP Publishing, 2018. ISBN 1757-899X.