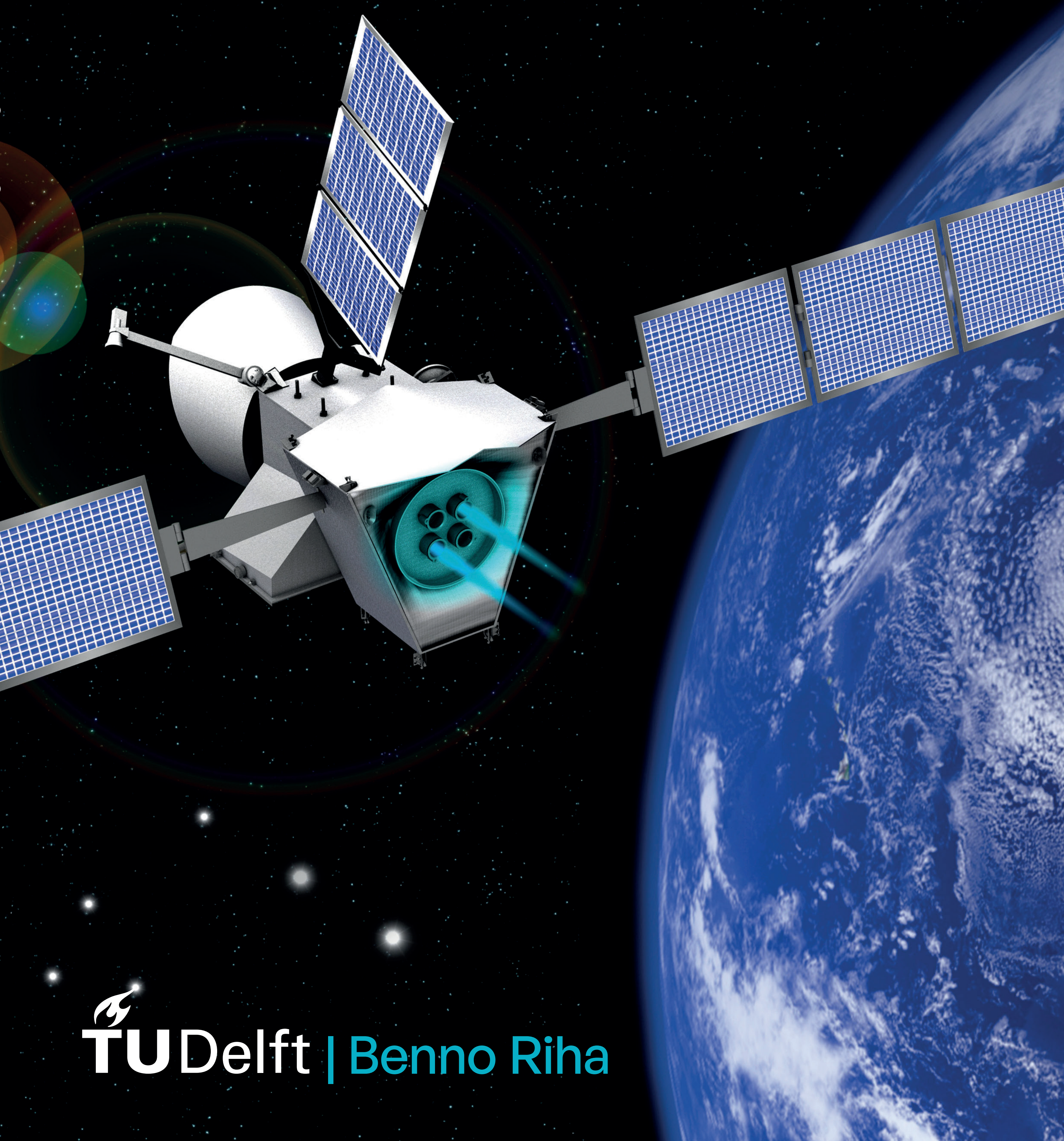
Feature Engineering for Low-Thrust Trajectory Optimization

A Systematic Analysis Using Pontryagin Fuel-Optimal Earth-Mars Transfer Trajectories



TU Delft | Benno Riha

Feature Engineering for Low-Thrust Trajectory Optimization

A Systematic Analysis Using Pontryagin Fuel-Optimal Earth-Mars Transfer Trajectories

by

Benno Riha

in partial fulfilment of the requirements for the degree of

Master of Science

at the Astrodynamics and Space Missions Group,
Faculty of Aerospace Engineering,
Delft University of Technology,

to be publicly defended on 30 August 2023 at 09:30 CEST.

Preface

Well, this is it. Finally. This thesis single-handedly kept me in Delft for a little longer than I expected, and I would be lying if I said all those months spent working on it were just fun and games. The difficult parts made the resolutions of the problems keeping me busy all the more satisfying, though.

Throughout it all, I was lucky enough to have Kevin Cowan as supervisor on my side – always finding positive words, even after the nths week being stuck on the same problem. Thank you. The entire university staff was always super helpful when anything was needed. From all those people, I would like to thank Dominic Dirkx, who was always available when anything Tudat related needed solving, even going so far as to implement was is needed for any thesis project. Additionally, I also want to thank the staff of the Delft High Performance Computing Centre. Whenever one has a question, they are super quick and helpful in their answers. I have not seen any problem they could not solve in a quick interaction on Mattermost. Finally, I want to thank Ekin Öztürk. They were super quick to answer my unsolicited email and even went as far as to look at my code to try and figure out the origin of the discrepancy between their published results and my implementation.

Finally, I want to thank everyone that was there and kept me going. My friends that made my stay in Delft what it was and always were there to share my thesis frustration. My friends not in Delft that also listened to some of it and kept me sane, sometimes more, sometimes less, remotely. And, of course, my family, who supported me throughout this journey and beyond.

Enjoy the read!

Benno Riha
Delft, August 2023

Abstract

Using low-thrust propulsion for interplanetary space missions has the potential to allow for more payload for the same mass put into orbit compared to what impulsive propulsion would allow for. The disadvantages are found in mission planning, however, as the continuous nature of the thrust yields a more complex problem. One potential solution to help in the early planning and discovery phase of mission design is to employ artificial neural networks (ANNs). This has been done in the past, yet only in a limited capacity. Specifically, the engineering of the feature space used with the neural network has never been investigated. This thesis attempts to provide a first look at the influence of different feature space compositions. This includes the use of nine different state representations but also an analysis of additional values in the feature space. Additionally, the effect of extraneous variables, one of them notably being the target of interest, on the neural network performance is analyzed. The dataset used is generated using indirect optimization, and the case investigated is a set of minimum fuel Earth-Mars transfer trajectories.

Low-thrust spacecraft trajectories are for space exploration what neural networks are for computer science: The vanguard of current trends with a lot of potential. Only recently have the two ideas, trajectory optimization and machine learning, been combined. In all the publications making use of machine learning for low-thrust optimization, a clear gap exists, however: Feature engineering has never been investigated. This thesis attempts to provide a first patch for that gap, limiting itself in scope to interplanetary Earth-Mars trajectories and feedforward neural networks. The data used as the basis to evaluate the performance of a number of factors having a potential influence on the choice of feature is obtained through indirect optimization. A novel method to generate those trajectories is implemented. The data is then used to investigate the effect different targets and network parameters have on the choice of features. On the feature side, a significant number of state representations are analyzed, both in dimensional and nondimensionalized form. Additionally, the feature space is expanded by additional variables, and various transformations are attempted.

Over the course of this work, the importance of properly scaled data has been demonstrated. It is also shown that using Keplerian state and costate as feature and target, respectively, reliably yields good results. When mass is estimated, fuel mass is preferable over total spacecraft mass. Finally, none of the additional parameters or transformations (besides nondimensionalization) attempted resulted in reliable improvements and are thus best avoided.

Contents

Nomenclature

Abbreviations

AAS	American Astronautical Society
Adam	stochastic machine learning optimizer, name derived from ‘adaptive moment estimation’
ANN	Artificial Neural Network
CAS	Computer Algebra System
ELM	Extreme Learning Machine
GEO	Geostationary Orbit or Geosynchronous Equatorial Orbit
GTO	Geostationary Transfer Orbit
IQR	Interquartile range
KS	Kustaanheimo-Stiefel Elements
MEE	Modified Equinoctial Elements
MISO	Multiple Input Single Output
MSE	Mean Squared Error
PINN	Physics Informed Neural Networks
PoNN	Pontryagin Neural Networks, i.e., PINNs for indirect method optimal control
POP	Position-on-Orbit Parameter
RAAN	right ascension of ascending node, also known as Ω
RSW	Radial-Tangential-Normal body-fixed frame – also known as RTN
RTN	Radial-Tangential-Normal body-fixed frame – also known as RSW
SIMO	Single Input Multiple Output
TFC	Theory of Functional Connections
USM-EM	Unified State Model – Exponential Mapping
X-TFC	Extreme Theory of Functional Connections

Scalar Symbols

β	costate scaling factor
$\Delta\mathcal{M}$	euclidean norm of a difference in ‘concise orbit descriptor’ $\mathcal{M}$
$\mathcal{L}$	running cost function
ℓ	mean longitude
η_p	eccentric Poincaré variable η
γ	flight-path angle
$\hat{\phi}$	fourth component of the 5D hyperspherical angles
$\hat{\theta}_{\{1,2,3\}}$	first three components of the 5D hyperspherical angles
Λ	longitude
λ	costate element
$\lambda_{\text{est},\square}$	element of the estimate for the costate vector
$\mathcal{H}$	Hamiltonian
μ	gravitational parameter
Ω	right ascension of ascending node
ω	argument of periapsis
Φ	latitude

ψ	heading angle
SF	switching function
θ	true anomaly
θ_{cyl}	polar angle element of the cylindrical element set
ε	homotopy (or continuation) parameter
$\vartheta_{\{1,2,3,4\}}$	Kustaanheimo-Stiefel element of given index
$\vartheta'_{\{1,2,3,4\}}$	Kustaanheimo-Stiefel derivative element of given index
ξ	vis-viva orbital energy
ξ_p	eccentric Poincaré variable ξ
a	semi-major axis
C	first velocity component USM-EM
e	eccentricity
$e_{usm,\{1,2,3\}}$	exponential mapping component USM-EM
f_e	Modified Equinoctial Element, related to the orbit's eccentricity
f_n	normal component of the thrust force
f_r	radial component of the thrust force
f_t	tangential component of the thrust force
g_0	Earth sea-level acceleration
g_e	Modified Equinoctial Element, related to the orbit's eccentricity
h	specific angular momentum, scalar
h_a	Homotopy path characterisation parameter
H_d	Delaunay canonical H
h_e	Modified Equinoctial Element, related to the orbit's orientation
i	inclination
I_{sp}	specific impulse
J	cost functional (to be minimized)
k_e	Modified Equinoctial Element, related to the orbit's orientation
L	true longitude
L_d	Delaunay canonical L
M	mean anomaly
m	mass
m_{dry}	dry mass, i.e., mass after the interplanetary transfer is completed
m_{fuel}	fuel mass needed for interplanetary transfer
N_{coverage}	number of data points per trainable parameter
N_{dataset}	number of data points in the dataset
N_{layers}	number of hidden layers
N_{nodes}	number of neurons per hidden layer
N_{params}	number of trainable parameters
p	semi-latus rectum
r_{cyl}	radius element of the cylindrical element set
R_{f1}	second velocity component USM-EM
R_{f2}	third velocity component USM-EM
s^2	MEE dynamics helper variable
s_{x_1}	pseudo-signum of the first Cartesian element, used in the Kustaanheimo-Stiefel definition
t	time

T_{max}	maximum thrust force
t_{transfer}	time taken for the interplanetary orbit transfer
u	throttle setting
u_p	oblique Poincaré variable u
v_p	oblique Poincaré variable v
$v_{\theta,cyl}$	polar velocity element of the cylindrical element set
$v_{\{1,2,3\}}$	Cartesian velocity of given index, equivalent to $v_{\{x,y,z\}}$
$v_{r,cyl}$	radial velocity element of the cylindrical element set
w	MEE dynamics helper variable
$x_{\{1,2,3\}}$	Cartesian position of given index, equivalent to $\{x, y, z\}$
Vector and Matrix Symbols	
$\hat{\boldsymbol{e}}_z$	unit-vector in Cartesian z direction
$\hat{\boldsymbol{i}}_\tau$	thrust direction vector
$\mathcal{M}$	concise orbit descriptor
$\boldsymbol{x}_{\vec{e}}$	vector of the MEE elements f_e, g_e, h_e, and k_e.
$\boldsymbol{A}$	matrix relating the state to the state dynamics
$\boldsymbol{B}$	matrix relating the control to the state dynamics
$\boldsymbol{D}$	control independent dynamics matrix
$\boldsymbol{\lambda}$	costate vector
$\boldsymbol{\lambda}_{\text{est}}$	estimate for the costate vector
$\boldsymbol{\Phi}$	boundary conditions function
$\varrho(\cdot)$	state space mapping from b to a, i.e., $\boldsymbol{x}_a(t) = \varrho(\boldsymbol{x}_b(t))$
$\boldsymbol{c}$	weighting vector, for the individual elements of the ‘concise orbit descriptor’ $\mathcal{M}$
$\boldsymbol{h}$	specific angular momentum, vector
$\boldsymbol{r}$	Cartesian position vector, elements x, y, z.
$\boldsymbol{u}$	control vector (thrust)
$\boldsymbol{v}$	Cartesian velocity vector, elements v_x, v_y, v_z.
$\boldsymbol{x}$	state vector
Sub- and Superscripts	
$\square^*$	optimal
$\square^\mathsf{T}$	transpose of $\square$
$\square_0$	initial
$\square_f$	final
$\square_t$	target
$\square_{\text{cart}}$	$\square$ in Cartesian elements
$\square_{\text{kep}}$	$\square$ in Keplerian elements
$\square_{\text{mee}}$	$\square$ in Modified Equinoctial Elements
$\dot{\square}$	derivative w.r.t. time
$\lVert\cdot\rVert_2$	Euclidean norm, suffix $_2$ is optional

List of Figures

List of Tables

1 Introduction

The idea of using low-thrust propulsion (Stuhlinger, 1957; Tsien, 1953) is roughly as old as the study of neural networks (McCulloch & Pitts, 1943) itself – and yet, it would take around 70 years before those two pieces found together. The first research combining artificial intelligence techniques and low-thrust trajectory optimization was published at the start of the 21st century (Dachwald & Seboldt, 2002). Roughly around the same time, low-thrust propulsion gained the technological readiness level to be deployed (Oh et al., 2014) – and has been a great success on missions such as BepiColombo Novara, 2002, Dawn Brophy et al., 2003, and Hayabusa Kuninaka et al., 2006. Now, more than ever, methods to find the best low-thrust trajectories allowing for the highest science return are of great interest.

Trajectory optimization is, of course, nothing new. Not even the optimization of low-thrust – and thus, constant thrust – trajectories is new. Long before low-thrust propulsion could be realized, the first shape-based method to describe trajectories in a way allowing for optimization was proposed: The logarithmic spiral (Bacon, 1959). Since then, countless more methods have been proposed – some of them analytical, some of them numerical. The analytical solutions do lose out on accuracy and on possible optimality. They are also not of significant interest in the context of this work, for a different reason, however: They can usually be evaluated way quicker than a neural network could be trained – or even evaluated, for that matter.

The trajectory-generation part will be introduced shortly, yet taking a step back first. The combination of machine learning and trajectory optimization has seen quite a bit of development since the first aforementioned publication. Several publications describing use-cases for interplanetary transfers (H. Li, Baoyin, & Topputo, 2019; H. Li et al., 2020; Yin et al., 2020), trajectory to and between Near-Earth Asteroids (H. Li, Baoyin, & Topputo, 2019; Mereta et al., 2017; Viavattene & Ceriotti, 2022; Xie & Dempster, 2021), orbit raising maneuvers (Arora & Dutta, 2020; H. Li, Topputo, & Baoyin, 2019), or trajectories in cis-lunar space (LaFarge et al., 2021; Miller & Linares, 2019). And these were only examples using neural networks rather than other parts of the artificial intelligence toolbox.

While those are all very divergent examples, they do have one thing in common: They barely do any feature engineering. Feature engineering is selecting the set of inputs (features) so that the performance of the neural network (or, more broadly, the algorithm used) is maximized. Most of them pick one or two different state representations, try both, and take the resulting best – if even that (see Section C.1 for an overview of the relevant literature). The issue with this approach is that no insight whatsoever is gained regarding what does and does not work. This means that future researchers are bound to just do that too in perpetuity. It is also rather expensive, as training large networks takes time and large amounts of computational power to be trained. This approach necessitates training multiple networks, none of which might actually be optimal. This leads to the research question sought to be answered within this work:

How to select the feature set leading to the best possible performance of a feedforward neural network used for interplanetary Earth-Mars low-thrust fuel-optimal trajectories?

This report's main content is found in the form of a paper manuscript, using the conference template of the American Astronautical Society (AAS)*. Chapter 2 consists of said manuscript. The paper is then followed by a wider look at the indirect trajectory-generation method and implementation used to generate the data that was the basis for the training of the networks evaluated, found in Chapter 3. Thereafter, additional background on the state representations and the statistical framework used is provided in Chapter 4. This is followed by Chapter 5, where the verification and validation of the methods used takes place, demonstrating that what was implemented actually matches both the intention and the physical reality. Finally, the research question and its sub-questions are formally answered, and recommendations are formulated in Chapter 6.

*Available online: http://www.univelt.com/FAQ.html#SUBMISSION [visited 12-05-2023]

(Preprint) AAS XX-XXX

FEATURE ENGINEERING OF FEEDFORWARD NEURAL NETWORKS FOR LOW-THRUST INTERPLANETARY TRAJECTORY OPTIMIZATION

Benno Riha[*] and Kevin Cowan[†]

Using low-thrust propulsion for interplanetary space missions has the potential to allow for more payload, yet the resulting continuous nature of the thrust yields a more complex mission-planning problem. Artificial neural networks (ANNs) are promising to help solve the problem. This paper attempts to provide a first look at the influence of different feature space compositions of the ANN, limiting itself in scope to interplanetary Earth-Mars trajectories and feedforward neural networks. The effects of several factors potentially influencing the choice of features is evaluated. The analysis is based on data obtained through indirect optimization. A novel iterative shooting method to generate the trajectories is implemented. The data is then used to investigate the effect different targets and network parameters have on the choice of features. On the feature side, nine different state representations are analyzed, both in dimensional and nondimensional form. Additionally, the feature space is expanded by additional parameters, such as the angular momentum vector and the vis-viva energy. Furthermore, a transformation of the angles is attempted, and the target vector itself is varied between three costate representations, the transfer time, and two fuel mass representations.

Over the course of this work, the importance of properly scaled data has been demonstrated. It is also shown that using Keplerian state and costate as feature and target, respectively, reliably produces good results. When mass is estimated, fuel mass is preferable over total spacecraft mass. Finally, none of the additional parameters or feature vector transformations (besides nondimensionalization) resulted in reliable improvements and are thus best avoided.

Electric low-thrust propulsion is an idea as old as spaceflight itself[1–3], and so is the general concept of Artificial Neural Networks (ANNs)[4,5]. Only recently have the two been combined in pursuit of an improvement in the performance of solutions attempting to solve the intractable set of nonlinear differential equations that the low-thrust optimization problem boils down to[6–9]. The possible uses of artificial intelligence in trajectory optimization are wide-ranging: Recent works have used them in a surrogate function setting[7,10–13], to optimize landing trajectories[14–16], for orbital docking[17], as neurocontrollers[6,18,19], and more[20,21]. A comprehensive overview of the field can be found in Shirobokov et al.[22] and Izzo et al.[23,24].

In spite of the wide range of related publications, one aspect has, so far, not been explicitly considered: Feature engineering – i.e., the way the data is formatted before it is fed to the neural network. When done in relevant literature, if at all, feature engineering is limited to using two or three state representations, training networks for all of them, and picking the one yielding the best results[25–29]. That approach is cumbersome, computationally expensive, and, while it does lead to a tailored solution, hardly any insight can be gained from it.

This work aims to systematically investigate several features and what influence other non-feature-related parameters have on the choice of the feature set. The data used to train the reference networks is generated using the indirect method, giving an additional meaningful option for targets. The scope of this work is

[*]M.Sc. Student, Astrodynamics & Space Missions, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands.
[†]Education Fellow + Lecturer, Astrodynamics & Space Missions, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands.

limited to Earth-Mars fuel-optimal transfer trajectories in combination with fully connected feedforward artificial neural networks.

The basis for all neural network training is data. The first section of this paper hence describes the methodology used to generate that data. From there, the data transformations to be evaluated – transforming both feature and target – are introduced. This does include different state representations but also additional features which may or may not influence the networks' learning performance. The different factors to be investigated are outlined next, in addition to a brief summary of the Design of Experiments analysis framework used. Finally, the results are presented and discussed.

DATA GENERATION

At a high level, generating orbital transfers can be done using two distinct methods: the direct method and the indirect method. The direct method converts the optimal control problem into a non-linear programming problem, which can then be solved by any of the widely available solvers[30]. Conversely, the indirect method transforms the optimal control problem into a Two-Point Boundary Value Problem (TPBVP), which then boils down to a shooting problem.

In this work, the latter approach is followed. The use of the indirect approach has two main reasons: (1) The formulation allows, within the simplified problem space, a mathematically optimal solution of the problem. The trajectory found by solving the problem, if any, is, by definition, an optimal minimum-fuel trajectory. (2) To solve this problem, the state space is extended from the seven base state elements (six kinematic and one mass element) by seven more: One Lagrange multiplier, or costate element, for every state element. Once the initial extended state (seven state and seven costate elements) is known, the trajectory is fully determined. This latter fact makes these costates – that are otherwise difficult to ascertain for a given trajectory – a metric that would greatly benefit from being accurately estimated through a neural network.

This section starts by defining the optimal control problem. From there, the optimal throttle setting and thrust direction are derived based on Pontryagin's minimum principle. This results in a shooting problem, the possible formulations of which are discussed after that. At last, the strategy used to obtain meaningful trajectories is introduced.

Optimal Control Problem

The definition of the dynamical system is following Izzo and Öztürk[31], which is itself a reformulation of Jiang et al.[32] in Modified Equinoctial Elements (MEE) instead of Cartesian elements. MEE are chosen as they are non-singular and allow for meaningful boundary conditions[29]. The dynamics assume negligible mass for the spacecraft, a single point-mass orbited body (here: the Sun), and a thrust vector expressed in Radial-Tangential-Normal (RTN or RSW) elements.

The optimal control problem itself is a standard Lagrange problem and is given by

$$\begin{aligned} \text{minimize} \quad J(\boldsymbol{x}(t), \boldsymbol{u}(t), t) &= \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{u}(t))\,\mathrm{d}t \quad \text{where} \quad u \in [u_{\min}, u_{\max}], \\ \text{subject to} \quad \dot{\boldsymbol{x}}(t) &= \boldsymbol{B}(\boldsymbol{x}(t))\boldsymbol{u}(t) + \boldsymbol{D}(\boldsymbol{x}(t)), \\ \boldsymbol{0} &= \boldsymbol{\Phi}_0(\boldsymbol{x}(t_0)), \\ \boldsymbol{0} &= \boldsymbol{\Phi}_f(\boldsymbol{x}(t_f)), \end{aligned} \tag{1}$$

where J is the cost functional that is to be minimized; $\mathcal{L}$ is the running cost; $\boldsymbol{x}$ is the state vector; $\boldsymbol{u}$ is the control function; $\boldsymbol{B}$ and $\boldsymbol{D}$ are matrices describing the dynamics; and the $\boldsymbol{\Phi}_\square$ describe the boundary conditions at t_0, initial time, and t_f, final time.

For a fuel-optimal trajectory, the running cost is just the integrated mass flow or, equivalently, the integrated throttle u. The optimal control for a fuel-optimal trajectory is a so-called bang-bang control, which is not ideal for numerical optimization. For this reason, it will be smoothed out here using a logarithmic barrier[33], thus making the running cost

$$\mathcal{L}(\boldsymbol{u}(t)) = u - \varepsilon \ln\left[u(1-u)\right], \tag{2}$$

where $u \in [0,1]$ is the throttle setting, and ε is a homotopy – or continuation – parameter. When $\varepsilon = 1$, J describes the energy-optimal problem, and as $\varepsilon \to 0$, the problem turns into the fuel-optimal one.

The dynamics are described in Modified Equinoctial Elements and are as follows[31,34]

$$\boldsymbol{B}(\boldsymbol{x}) = \sqrt{\frac{p}{\mu}}\frac{1}{w}\begin{bmatrix} 0 & 2p & 0 \\ w\sin L & f_e + (1+w)\cos L & -g_e\left(h_e \sin L - k_e \cos L\right) \\ -w\cos L & g_e + (1+w)\sin L & f_e\left(h_e \sin L - k_e \cos L\right) \\ 0 & 0 & \frac{s^2}{2}\cos L \\ 0 & 0 & \frac{s^2}{2}\sin L \\ 0 & 0 & h_e \sin L - k_e \cos\left(L\right) \end{bmatrix}, \tag{3}$$

and

$$\boldsymbol{D}(\boldsymbol{x}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{\frac{\mu}{p^3}}w^2 \end{bmatrix}^\top, \tag{4}$$

with

$$w = 1 + f_e \cos L + g_e \sin L, \quad \text{and} \quad s^2 = 1 + h_e^2 + k_e^2 \tag{5}$$

where p is the semi-latus rectum; f_e and g_e are MEE elements relating to the orbit eccentricity; k_e and h_e are MEE elements relating to the orbit orientation; L is the true longitude; and μ is the gravitational parameter of the orbited body – in this case, the Sun. The control – here thrust – vector has its direction and magnitude expressed separately, thus

$$\boldsymbol{u}(t) = \frac{T_{\max} u(t)}{m}\hat{\boldsymbol{i}}_\tau = \left[f_r, f_t, f_n\right], \tag{6}$$

where $u \in [0,1]$ is the throttle setting, $T_{\max}$ is the maximum thrust, m is the instantaneous mass, and $\hat{\boldsymbol{i}}_\tau$ is the thrust direction. f_r is the thrust in radial direction; f_t is positive in the direction of the velocity vector and tangential to the orbit; and f_n completes the right-handed coordinate system and is thus normal to the orbital plane.

The dynamics are given by

$$\dot{\boldsymbol{x}}_e = \frac{T_{\max} u(t)}{m}\boldsymbol{B}(\boldsymbol{x})\hat{\boldsymbol{i}}_\tau + \boldsymbol{D}(\boldsymbol{x}), \qquad \text{and} \qquad \dot{m} = -\frac{T_{\max} u(t)}{I_{sp} g_0}, \tag{7}$$

where I_{sp} is the thruster's specific impulse and g_0 is Earth's standard sea-level gravitational acceleration. The $\square_e$ subscripts denote that the state vector is in MEE.

Applying Pontryagin's Minimum Principle

The control function minimizing the cost functional defined in Equation 1 and 2 can be found by applying Pontryagin's minimum principle[35]. The resulting Hamiltonian is then

$$\mathcal{H} = \frac{T_{\max} u}{m}\boldsymbol{\lambda}_e^\top \boldsymbol{B}\hat{\boldsymbol{i}}_\tau + \lambda_L \sqrt{\frac{\mu}{p^3}}w^2 - \frac{T_{\max}}{I_{sp} g_0}\lambda_m u + \left\{u - \varepsilon \ln\left[u(1-u)\right]\right\}, \tag{8}$$

where $\lambda_\square$ is the costate associated with the element specified, and $\boldsymbol{\lambda}_e$ is the costate vector associated with the MEE state. For simplicity, the dependencies on time and state are omitted from this point on.

Pontryagin's minimum principle states that the optimal control $\boldsymbol{u}^*$ is the one that minimizes $\mathcal{H}$ for all t along the optimal trajectory. The optimal control is thus given by

$$\boldsymbol{u}^* = \underset{\boldsymbol{u}\in\mathcal{U}}{\arg\min}\left[\frac{T_{\max} u}{m}\boldsymbol{\lambda}^\top \boldsymbol{B}\hat{\boldsymbol{i}}_\tau - \frac{T_{\max}}{I_{sp} g_0}\lambda_m u + \left\{u - \varepsilon \ln\left[u(1-u)\right]\right\}\right], \tag{9}$$

and, hence,

$$\frac{\partial}{\partial \boldsymbol{u}^*}\left[\frac{T_{\max} u}{m}\boldsymbol{\lambda}^\top \boldsymbol{B}\hat{\boldsymbol{i}}_\tau - \frac{T_{\max}}{I_{sp} g_0}\lambda_m u + \{u - \varepsilon \ln [u(1-u)]\}\right] = 0. \tag{10}$$

The optimal thrust direction is given by Lawden's Primer Vector Theory[36]. In Cartesian elements, this is the negative direction of the velocity costate $\boldsymbol{\lambda}_v$, while in MEE it is

$$\hat{\boldsymbol{i}}_\tau^* = -\frac{\boldsymbol{B}^\top \boldsymbol{\lambda}}{\|\boldsymbol{B}^\top \boldsymbol{\lambda}\|}. \tag{11}$$

Combining Equation 10 and 11, the ideal throttle setting can be found,

$$u(t)^* = \frac{2\varepsilon}{\mathrm{SF} + 2\varepsilon + \sqrt{\mathrm{SF}^2 + 4\varepsilon^2}}, \quad \text{with} \quad \mathrm{SF}(t) = -\frac{T_{\max}}{m}\|\boldsymbol{B}^\top \boldsymbol{\lambda}\| - \frac{T_{\max}}{I_{sp} g_0}\lambda_m + 1, \tag{12}$$

where SF is a switching function, which is the part of Equation 10 that does not vanish as $\varepsilon \to 0$.

Furthermore, the dynamics for the costates are given by

$$\dot{\boldsymbol{\lambda}}_e = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}, \quad \text{and} \quad \dot{\lambda}_m = -\frac{\partial \mathcal{H}}{\partial m}. \tag{13}$$

The full derivative matrices for Equation 13 can be found in Izzo and Öztürk[31].

The Shooting Problem

With the dynamics defined by Equation 7 and 13, the missing components are the boundary conditions. The departure condition is necessarily the MEE state and the mass at the point of departure. For arrival, the mass m and the true longitude L (i.e., the position on the arrival orbit) are kept free. This implies $\lambda_L|_{t=t_f} = 0$ and $\lambda_m|_{t=t_f} = 0$ – which are the so-called transversality conditions. Additionally, as the final time t_f is also kept free, $\mathcal{H}|_{t=t_f} = 0$. In actuality, the Hamiltonian is not only zero at the final time, but because the dynamics are time-invariant, it is constant along the entire optimal trajectory and must thus always be zero.

At the starting point of the trajectory, the entire costate part of the extended state vector is thus unknown. At the final time, however, only the direction of the first five costate elements and the final mass are unknown – the magnitude of that five-element vector is dictated by $\mathcal{H} = 0$. The transfer time is not known at either point, but the spacecraft's mass bounds limit it. Not only is the departure mass constrained to a given wet mass, but the arrival mass is also not allowed to go below 60 % of the departure mass.

The two shooting functions representing the forward and backward propagation are thus

$$\boldsymbol{\Phi}_{t^+}(\boldsymbol{\lambda}_{0,6}, \lambda_{m_0}, t_f) = \left\{[p, f_e, g_e, h_e, k_e]^\top - [p, f_e, g_e, h_e, k_e]^\top_{\text{final}}, \mathcal{H}, \lambda_L, \lambda_m\right\}\Big|_{t=t_f} = \boldsymbol{0} \tag{14}$$

and

$$\boldsymbol{\Phi}_{t^-}(\boldsymbol{\lambda}_{f,5}, m_f, t_0) = \left\{ [p, f_e, g_e, h_e, k_e, m]^\top\Big|_{t=t_0} - [p, f_e, g_e, h_e, k_e, m]^\top\Big|_{\text{initial}}\right\} = \boldsymbol{0}. \tag{15}$$

Note that in the subscript of $\boldsymbol{\lambda}_{\square,i}$ here, i stands for the number of unknown elements. The subscripts 'initial' and 'final' designate the mandated target departure and arrival states, respectively. Given the presence of fewer unknowns at the destination, this work will use the backward propagating shooting function. In other words, the trajectories are propagated in negative time direction.

Solving the Shooting Problem

The previous section showed that backward shooting is a problem of lower dimension and is thus expected to be easier to solve. To be able to do backward shooting, scaling the initial (initial referring to the starting point of propagation throughout this work) guess such that $\mathcal{H} = 0$ is necessary.

This is done utilizing an iterative process, which substitutes Equation 11 into Equation 8, and then separates the costate vector's magnitude from its direction, i.e., $\boldsymbol{\lambda} = |\beta|\boldsymbol{\lambda}_{\text{est}}$, where $\boldsymbol{\lambda}_{\text{est}}$ is the initial guess. The resulting expression for β,

$$\beta = \left(\frac{T_{\max} u}{m} \left\| \boldsymbol{B}^\top \boldsymbol{\lambda}_{\text{est}} \right\| - \lambda_{\text{est},L} \sqrt{\frac{\mu}{p^3}} w^2 \right)^{-1} \left[-\frac{T_{\max}}{I_{sp} g_0} \lambda_m u + u - \varepsilon \ln \left[u(1-u) \right] \right], \tag{16}$$

is recursive, as u is a function of $\boldsymbol{\lambda}$. An iterative relaxation method converges to an acceptable value of $\mathcal{H}$ (10^{-5}) within a handful of iterations.

In order to assess how close a given state is to the final orbit, a metric expressing how close two orbits are, independently of the position on said orbit, is devised. The metric used is the norm of the difference of the following 'concise orbit descriptor,'

$$\mathcal{M} = p \left[\; c_f f_e, \quad c_g g_e, \quad c_h h_e, \quad c_k k_e, \quad p^{-2} + (c_f f_e)^2 + (c_g g_e)^2 + (c_h h_e)^2 + (c_k k_e)^2 \; \right]^\top, \tag{17}$$

which contains all MEE elements scaled by p so that their magnitudes are comparable. This also gives the metric and its ℓ_2-norm a length dimension. The last element is present to ensure it remains expressive for (near) circular orbits. Additionally, the factors $c_\square$ are used case-by-case to ensure it accurately reflects the values encountered during transfer. For a (backward) Earth$\to$Mars transfer, those values are $\boldsymbol{c} = [1, 1, 10, 10]$.

When using $\boldsymbol{c} = \mathbf{1}$, a state-of-the-art non-linear optimizer's best result only matched the eccentricity (f_e and g_e) and semi-latus rectum, while the orientation itself was still very much off.

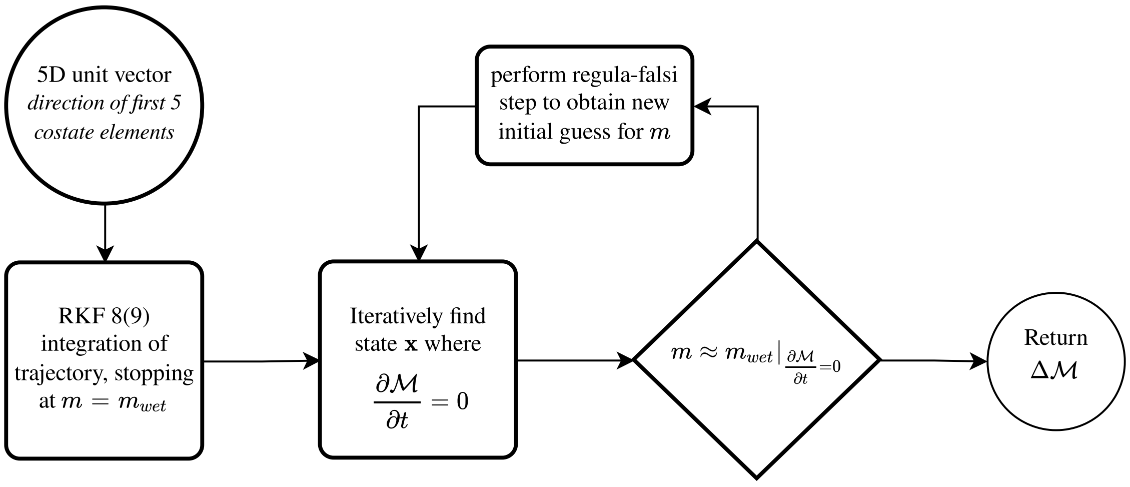

5D unit vector
direction of first 5 costate elements
RKF 8(9) integration of trajectory, stopping at $m = m_{wet}$
Iteratively find state $\mathbf{x}$ where $\frac{\partial \mathcal{M}}{\partial t} = 0$
perform regula-falsi step to obtain new initial guess for m
$m \approx m_{wet}|_{\frac{\partial \mathcal{M}}{\partial t}=0}$
Return $\Delta\mathcal{M}$

Figure 1. The shooting function, as seen by the optimizer: Five inputs, one cost output.

From here on, the backward shooting problem is reduced to a single function with one output – the ℓ_2 norm of the $\Delta\mathcal{M}$ (the Δ is w.r.t. the target orbit) – and one 5D unit vector as input. This simplified shooting function is achieved through two iterative layers. The outer layer iterates on the initial mass (which is the dry mass for backward propagation), such that the mass is $m = m_{\text{wet}}$ at the point where $\Delta\mathcal{M} = \|\mathcal{M} - \mathcal{M}_{\text{initial}}\|_2$ is minimal – i.e., where the orbit is closest to the target. Targeting the minimum point is done in the inner layer, which uses derivative-based adaptive stepping to target $\partial\Delta\mathcal{M}/\partial t = 0$. The outer loop uses a *regula falsi* algorithm of the Illinois[37] variant (the tolerance used here is $\mathcal{O}(10^{-6}) \approx 1.5 \times 10^{-3}$ kg) to iterate on the mass. A schematic representation of the just described shooting function is seen in Figure 1.

Finally, now that this function only takes a 5D unit vector as input, it can be solved by a non-linear optimizer such as SNOPT[38], given an appropriate initial guess. Initial guesses are found by shooting several trajectories – drawing the required 5D unit vectors from a Sobol[*] distribution (transformed to a normal distribution using the quantile function) using Muller's method[39]. Many guesses need to be taken until a guess with $\|\Delta\mathcal{M}\|_2 \leq 10^{-2}$ is found. This initial guessing method does not guarantee convergence, the threshold of which is taken to be $< 10^{-5}$ of the norm of $\Delta\mathcal{M}$. Convergence does occur for about $33\,\%$ of initial guesses taken that way. This results in the dataset of 811 534 trajectories used in this work, which took about 60 CPU seconds per viable trajectory[†] to obtain.

The spacecraft considered has a wet mass of $1500\,\mathrm{kg}$, a specific impulse of $3800\,\mathrm{s}$, and a max-thrust of $0.33\,\mathrm{N}$ (following Izzo and Öztürk[31]). The target Earth ephemeris is taken from Tudat's SPICE interface at Julian Year 2030. The Mars orbit used has its timestamp drawn from a Sobol distribution for a full orbit (approximated as $687\,\mathrm{Julian\ days}$), starting from Julian Year 2030. The integration is done using Tudat's implementation[41] of the Runge-Kutta-Fehlberg 8(9) variable step-size integrator, using absolute and relative tolerances of 10^{-12}.

DATA TRANSFORMATIONS

Optimizing the feature space cannot be done in a vacuum, as the best option is likely dependent on numerous external factors. One important factor influencing the feature choice is the target to be estimated, yet other factors may influence the feature selection, too. The costate target can be transformed similarly to the orbital state description in the feature. This section will look at two kinds of transformations: (1) Transformations that do not add or remove any information yet express the data differently for both the feature and the target; and (2) transformations that add data that may be redundant in a mathematical sense, yet have the potential to benefit the network performance. In essence, this second type of transformation tries to lower the theoretical barrier in network size implied by the Universal Approximation Theorem[42].

While the data generation is done nondimensionally, dimensionality is considered as a factor, too. The nondimensionalization used is such that the gravitational parameter $\mu = 1$, the spacecraft wet mass $m = 1$, and the unit distance is 1 AU. Note that the unit of a costate is time divided by the unit of its state counterpart.

Transformation of Orbital States

For reasons laid out in the section introducing the optimal control problem, the data generation is done in Modified Equinoctial Elements. There is no reason to assume that this representation works best as part of a neural network feature set, however, and thus nine different state representations are investigated. The ones that do not require additional explanation are the Cartesian, Keplerian, spherical, and cylindrical elements. Of the less common state representations used, the first one is the Unified State Model[43] of Exponential Mapping kind (USM-EM), the conversion of which is done using Tudat[41]. Delaunay and Poincaré elements are also included, thereby adding two canonical representations to the set of candidates considered. The definitions of the latter two are taken from Hintz[44][‡]. Finally, the Kustaanheimo-Stiefel (KS) elements conversion used is based on Fukushima[46]. Instead of having eight elements where one is always zero, depending on the sign of the first Cartesian element, that information is encoded in the sign of the first KS element. This change brings forth the set of seven KS elements used. The redundant orbital energy included as ninth element in Fukushima[46] is left out, too.

Transformation of Orbital Costates

Due to the MEE nature of the data generation, the costates are MEE costates, too. The costates are not contained in the feature vector but are part of the estimated target space. Even though feature engineering is the focus of this work, the target representation needs to be treated as a nuisance factor (a factor that can be controlled and measured yet is not directly investigated), at the very least.

[*]Implementation based on `https://web.maths.unsw.edu.au/~fkuo/sobol/` [visited 24-01-2023]

[†]The computations were done on Intel XEON E5-6248R 3.0 GHz CPUs[40].

[‡]Note that the conversion from MEE to Poincaré is wrong in Hintz[44], see Lyon[45] for the correct one.

In order to map between different costate representations, the following relation, found in Taheri et al.[47], is used,

$$\boldsymbol{\lambda}_{\boldsymbol{x}_a}(t)^{\mathrm{T}} \left[\frac{\partial \boldsymbol{\varrho}(\boldsymbol{x}_b)}{\partial \boldsymbol{x}_b}\right]\Bigg|_t = \boldsymbol{\lambda}_{\boldsymbol{x}_b}(t)^{\mathrm{T}}, \tag{18}$$

where $\boldsymbol{x}_i$ is the state in a given state representation; $\boldsymbol{\lambda}_{\boldsymbol{x}_i}$ are the corresponding co-states; and $\boldsymbol{\varrho}(\cdot)$ is the state space mapping from b to a, i.e., $\boldsymbol{x}_a(t) = \boldsymbol{\varrho}(\boldsymbol{x}_b(t))$. The inverse mapping can be found by inverting the Jacobian matrix or by swapping a and b in Equation 18, depending on what is more convenient. The Jacobian matrix in Equation 18 is derived using a computer algebra system*. The costate representations considered are the ones corresponding to Cartesian and to Keplerian elements.

Additional Features

Additional features used are metrics that are not directly part of the dynamics but may allow for convergence where the network does not have enough data or not enough nodes (due to a limited amount of data). These transformations include: (1) the sine-cosine transformation of angles, (2) the vis-viva energy as an additional feature, and (3) the three Cartesian components of the specific angular momentum as additional features. Those parameters are all evaluated in multiple variants: (i) their value at the initial point, (ii) their value at the final point, (iii) the inclusion of both the starting and final value, (iv) the difference between the initial and final value, and (v) exclusion.

STATISTICAL EXPERIMENT AND LEARNING SETUP

To isolate the effects of a given feature set, many factors potentially influencing the result have to be analyzed. Considering all combinations of factor levels (a factor being a variable that is considered, while a level is one of the possible values of said variable) is not feasible, at least if a considerable number of factors is to be analyzed. For that reason, a tool from the realm of Design of Experiments is used in this work: The fractional factorial experiment design.

A full factorial design is a set of "experiments" (in this case, neural network configurations) containing all possible combinations. Running a full factorial design has the advantage that it does not only enable insight into the effect of every factor itself but also includes the effect of all combinations of factors. These effects of factor combinations are called interaction effects. When linearly approximating the result as a function of the factor levels, the interaction effects would be the cross terms. The higher-order interaction terms likely do not contain any meaningful information, and the number of experiments necessary can thus be reduced without losing substantial understanding[48]. This reduction in configurations considered does mean that some (interaction) effects can no longer be distinguished from each other. Those effects that can no longer be separated are said to be confounded with each other. By carefully selecting the experimental runs to include, the confounding can be limited to higher-order effects, i.e., the effects of the factor themselves are uniquely identifiable, yet interactions of multiple factors may be confounded with other multifactor interactions. A fractional factorial design minimizing the lower-order interactions is called a minimum aberration design.

Table 1. Screening design numerical factors (hyperparameters) and their levels.

N_{dataset}	Batch Size	N_{layers}	N_{coverage}
$81\,153$	$1\,\%$	6	5
$811\,534$	$5\,\%$	11	20

Note: 84 runs for each numerical level.

The Screening Design

In Design of Experiments, a screening design is a design where only the highest order effects are considered, thereby allowing to focus on factors and levels of those factors that are most impactful in subsequent

*Maple (2022). Maplesoft, a division of Waterloo Maple Inc., Waterloo, Ontario.

Table 2. Screening design categorical factors, their levels, and the number of experiments per level.
i+f: initial and final, Δ: difference, $\boldsymbol{x}_{\text{Feat.}}$: feature set, ξ: vis-viva energy, $\boldsymbol{h}$: specific angular momentum, -: exclude

Direction	State Repr.	$\boldsymbol{x}_{\text{Feat.}}$	POP[*]	Angles	Dim.	ξ	$\boldsymbol{h}$	Target
t^+ 84	MEE 29	i 55	include 111	sin-cos 76	non. 84	- 24	- 34	$\boldsymbol{\lambda}_{\text{kep}}$ 30
t^- 84	KS 7	f 63	- 57	raw 92	dim. 84	i 37	i 27	$\boldsymbol{\lambda}_{\text{mee}}$ 29
	Cartesian 7	i+f 50				f 34	f 30	$\boldsymbol{\lambda}_{\text{cart}}$ 26
	Keplerian 27					i+f 32	i+f 39	t_{transfer} 31
	Spherical 14					$\Delta_{\text{i,f}}$ 41	$\Delta_{\text{i,f}}$ 38	m_{dry} 28
	USM-EM 15							m_{fuel} 24
	Cylindrical 14							
	Delaunay 28							
	Poincaré 27							

number of neural networks trained having a given factor level

experimental runs. The screening design is found in Table 1 and 2, where the different factors, their levels, and the number of network configuration containing each factor level are presented. The terms 'hyperparameter' and 'numerical factor' are used interchangeably in this work, as they happen to both describe the parameters found in Table 1, albeit from a different point of view. Note that some combinations of factor levels do not exist and were thus not considered. Notably, the position-on-orbit parameter[*] (POP) is not present for all orbits, and not including it is hence not always possible. The same applies to the angle transformation, in which all angles are replaced by their sine and cosine values. This transformation is also only applicable for element sets that do contain angles. Those two limitations brought the number of different networks to be trained for the screening design down to 168 (starting with a 256-run fractional factorial design).

The factors were selected to cover the most encountered ones in literature, on top of which additional parameters were added, that have anecdotally shown potential. One might notice that the factors considered do contain some, yet not all, hyperparameters one would expect for a neural network training setup. Specifically, the different activation functions within the networks were explicitly not considered, nor were any learning hyperparameters contemplated. Both were omitted to keep the scope of the analysis manageable. The activation functions were left out, as performance benefits over the ReLU function found in preliminary trials were not significant. The learning hyperparameters were neglected for three reasons: (1) They are too numerous to realistically consider, (2) the default settings already proved performant, and (3) they are deemed unlikely to influence conclusions regarding the choice of feature vector.

$\text{N}_{\text{coverage}}$ in Table 1 is the ratio of data points to learnable parameters, where the latter refers to the set of weights and biases that are adjusted during training. This ratio is computed by using the dataset size (after a test-split of 0.9 and a validation-split of 0.85, called N_{params}), and then solving the equation for the number of parameters of a fully connected feedforward neural network. Given that the number of weights from and to the input and output layer is small compared to the hidden-layer connections, those parameters are not included in the count. The number of neurons per hidden layer is thus taken as

$$N_{\text{nodes}} = \frac{-N_{\text{layers}} + \sqrt{N_{\text{layers}}^2 + 4N_{\text{params}}N_{\text{layers}}/N_{\text{coverage}} - 4N_{\text{params}}/N_{\text{coverage}}}}{2(N_{\text{layers}} - 1)}. \tag{19}$$

One last parameter not present in the first iteration of the screening design, yet used thereafter, is the 'costate scaling'. Two meaningful options exist for scaling the costates. Either, the costate vector is in its 'final' scaled length, which is the one that yields $\mathcal{H} \approx 0$. Or, the costates are in a normalized form (i.e., $||\boldsymbol{\lambda}||_2 = 1$). Both are equivalent, as the normalized ones can be rescaled for $\mathcal{H} = 0$ (note, however, that that solution may not be unique).

[*]'Position-on-orbit,' the only fast-changing parameter. Describes merely the position on the orbit, where the orbit itself is fully defined without it. Present in MEE, Keplerian, Delaunay, and Poincaré elements.

Final Design

Given the complex nature of mixed-level (i.e., factors having different numbers of levels or possible values) designs and that the screening design allows adequate insight into what does and what does not work, all non-binary parameters are reduced to the two most promising options (see the results section) for final analysis. The already binary parameters are kept in, as is. The reduction to binary is done on a per-target basis, resulting in three distinct statistical experiments. Those new experiments can be found in Table 3, where every level is present in 64 experiments. Note that the new values for the numerical factors were found by stepping the derivative of the linear least-squares MSE fit in the negative direction (direction of decreasing MSE). The experimental designs used are fractional factorials with only confounding between two-factor interactions and higher (this is known as a resolution IV design), which results in 128 runs for the number of binary factors investigated here. The column order is optimized during a second run to ensure that the two-factor interactions in the 20 highest magnitude linear fit factors are not confounded.

Table 3. Final design factors and their two levels, per target type.

i+f: initial and final, $\boldsymbol{x}_{\text{Feat.}}$: feature set, ξ: vis-viva energy, $\boldsymbol{h}$: specific angular momentum, -: exclude

Target Type	State Repr.	$\boldsymbol{x}_{\text{Feat.}}$	ξ	$\boldsymbol{h}$	Target	Batch Size	N_{layers}	N_{coverage}
costate	Poincaré	i	-	f	$\boldsymbol{\lambda}_{\text{kep}}$	1	8	10
	Keplerian	f	i	$\Delta_{\text{i,f}}$	$\boldsymbol{\lambda}_{\text{mee}}$	3	10	13
mass	MEE	i	-	-	m_{dry}	2	6	11
	Keplerian	f	f	i	m_{fuel}	5	9	14
time	MEE	i	-	f	t_{transfer}	1	7	11
	Keplerian	i+f	$\Delta_{\text{i,f}}$	$\Delta_{\text{i,f}}$		3	9	15

Learning Setup

The networks are trained using TensorFlow v2.4.1[49]. The networks are fully connected, with ReLU activation functions for the hidden and input layers and linear activation functions for the output layer. All networks are initialized using 'He normal'[50] for the input layer and 'Lecun normal'[51] for the hidden layers and the output layer. The loss function is the Mean Squared Error (MSE). Additionally, an early stopping condition of 10^{-3} with a patience of 15 epochs is used. The optimizer is the Adam optimizer[52].

All choices made regarding training setup can also be viewed as factors, at the very least as nuisance factors. The reasons why they have nonetheless been fixed fall into two categories.

The activation functions and initialization schemes were fixed because preliminary results indicated that their impact was minimal. The design space could thus be reduced by leaving them out.

The optimizer, on the other hand, may very well make a significant difference. This factor is, nevertheless, not considered in this work. The reason for that is that varying the optimizer is everything but straightforward. Other optimizers (like, e.g., RMSprop, Stochastic Gradient Descent, Adagrad) need fine-tuning of parameters specific to them to yield usable results, which was not feasible within the bounds of this work.

RESULTS

The resulting performance of the different factors analyzed in the experimental designs, defined earlier, are presented and discussed within this section. It starts with the initial screening design, which contains dimensionality as a factor. Given the effects of the dimensionality, an iteration of the screening design was needed, which is discussed next. From there, the two most promising options for each factor are extracted on a per-target basis, which constitutes the final design and is discussed last. Note that the Mean Squared Error (MSE) within this section always refers to the error between the target values and the values predicted by the network for the corresponding features in the test set unless indicated otherwise.

Initial Screening Design

The initial iteration of the screening design, as seen in Table 2, includes the dimensionality of the data as a parameter. Given the rather significant difference in magnitude of various elements, it is not entirely unexpected for that performance to be abysmal. This can be seen in Figure 2, where only very few dimensionalized configurations converge to a stable minimum value. Following this development, the dimensionality is dropped as a parameter. Instead, the costate scaling (with values 'normalized' or 'scaled,' where the latter is scaled for $\mathcal{H} = 0$) is introduced. This is discussed in the next section.

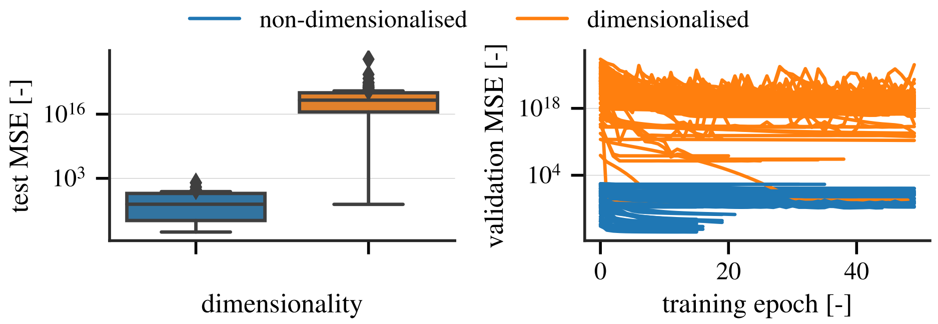


Figure 2. Distribution of the test MSE of the test data w.r.t. dimensionality (left), and the path followed by the MSE of the validation data during learning w.r.t. dimensionality (right).

Iterated Screening Design

The iterated design is only marginally different, as it no longer includes the dimensionality of the data (the data in all variants is now nondimensionalized). In contrast, it does now include the costate scaling. At this stage, the Design of Experiments framework would attempt to fit a linear model to explain how the different factors contribute to the MSE. This is not done yet, as outliers would dominate this linear model. For that reason, the distribution of the effects of the individual factor levels on the MSE will be looked at instead. By looking at the distribution resulting from the factors and their values individually, one can assess what impact that factor has irrespective of all other factors analyzed. This is possible because the different variations of the other factors are present in all variants of the factor analyzed, resulting in the distributions seen.

Notes on Statistical Interpretation When looking for the best feature set, one is looking for the one minimizing the MSE, not one that, on average, results in the lowest result. While that is the case, focusing

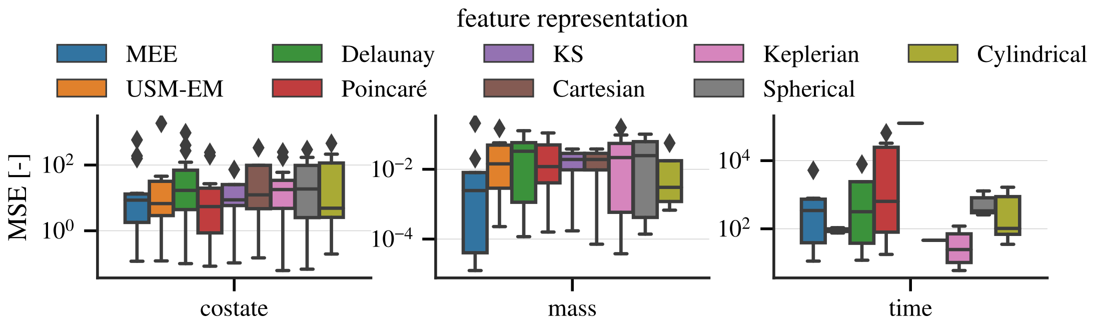


Figure 3. Statistical distribution of the MSE of the screening design for the different levels of the feature representation used.

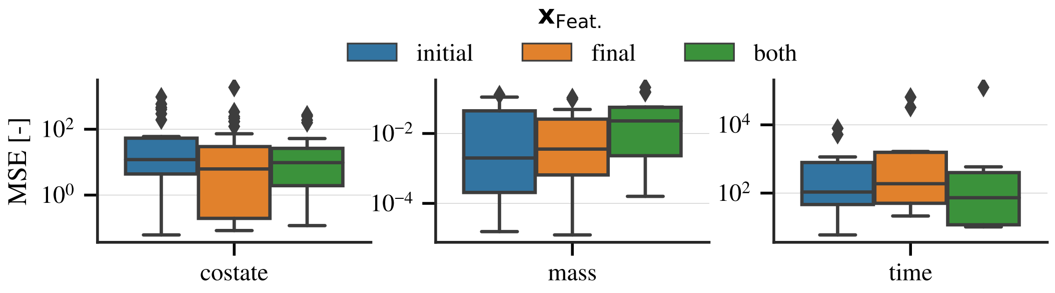


Figure 4. Statistical distribution of the MSE for the iteration of the screening design for the different sampling points included in the feature set.

only on the minimum is not necessarily helpful, as the minimum point is one specific combination of all factors involved. It does not allow drawing conclusions beyond that particular case. Therefore, the minimum will be kept in mind in the following analysis, yet the location of the interquartile range (IQR, 25th to 75th percentile of the data) and that of the median remain the focus.

The Impact of the Choice of State Representation The distribution of the MSE per target per feature set used is plotted in Figure 3. Additionally, also see Figure 4, which shows the effect of including either the initial (works best when estimating mass), the final (best when estimating the costates), or both (best when estimating the transfer time) states in the feature set.

When overlooking the entire space, it seems using Keplerian elements, or MEE are both strong choices, regardless of any other design decision. Neither is always the best, yet both reliably score well in their lower IQR half (i.e., the half with the lowest MSEs). Modified Equinoctial Elements result in a lower median than Keplerian elements when targeting the costates or the mass, whereas Keplerian elements are the clear choice when targeting time of flight.

Focusing on the different targets individually, it seems the impact the choice of feature set has is different for the various cases. When costates are estimated, all element sets seem to perform roughly similarly, with similar interquartile ranges and minima. Noteworthy here is that the Poincaré elements seem to perform slightly better than the others – even though Poincaré costates are not part of the targets considered. On the other hand, the MEE seem to be the clear choice for mass estimation, as the corresponding IQR is below the median of all but one of the remaining options. Finally, the choice of state representation seems most impactful when predicting the transfer time.

Some element sets only yield very limited ranges of results: The USM-EM elements, the Kustaanheimo-Stiefel elements, and the Cartesian elements. All of those are suboptimal choices, but Kustaanheimo-Stiefel specifically is orders of magnitude worse than any of the others. The limited range in those items may partially be due to little variability (no POP and no angles), resulting in only 7 runs for KS and Cartesian (see Table 2). That fact still does not negate the seemingly identical performance of all options that were tried, however.

To summarize, whether it is worth spending time on optimizing the choice of state representation depends on the target approximated. One must be careful in selecting orbital elements when approximating the time of free-time fuel-optimal trajectories, can probably get away with most choices when targeting the costate, and should likely use MEE when the mass is the object of interest. Keplerian elements or MEE are likely a good starting point when in doubt or pressed for time.

The Impact of the Target Representation The impact of the broad target (i.e., time, costate, or mass) itself is already inherently discussed throughout this section, as it directly influences the results obtained for the other factors themselves. However, the impact of the different options chosen to represent the same target types is not discussed here. Figure 5 demonstrates the distributions described hereafter have.

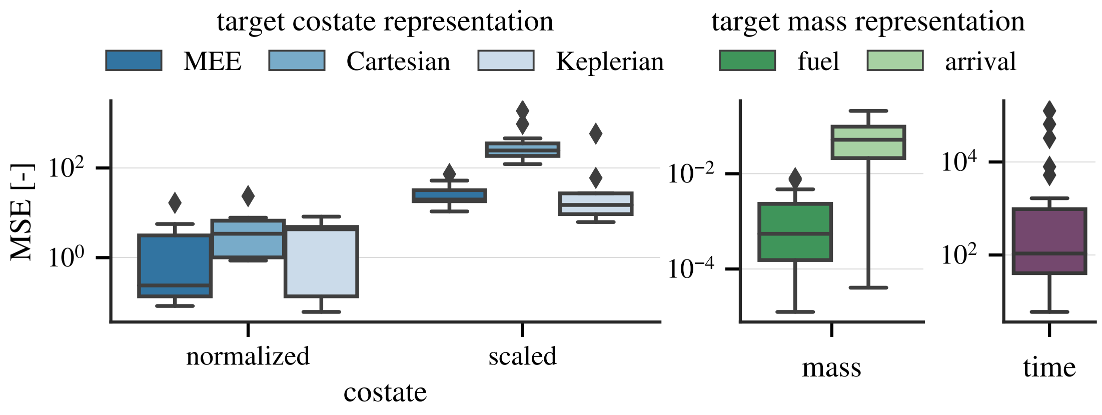


Figure 5. Statistical distribution of the MSE of the screening design for the different levels of the target factor.

Starting with the costates, one must first note that the normalized and scaled MSEs are not comparable as the costate magnitude is different, and so are thus their error magnitudes. Comparisons within the groups can be made, however, and there, Cartesian costates as targets yield worse results for both scaling options. When normalizing the costates, MEE costates have a significantly lower median than Keplerian costates. The scaled costates option has the Keplerian costates exceeding the MEE costates' performance. This is a surprising result as, to the authors' knowledge, Keplerian costates have not been exploited before.

Finally, the mass target, too, had two possible estimated values: The arrival mass (or dry mass) and the fuel mass. Estimating the fuel mass rather than the dry mass (or arrival mass) seems to be the obvious choice here. Given that the relation between the dry mass and the fuel mass is constant, the networks' bias appears less well estimated than the fuel mass relation itself.

The Impact of the Direction of Time As expected, the backward propagated trajectories, having costate elements λ_L and λ_m as zero, perform best. The reduction in target space seems to have an impact of about an order of magnitude MSE, across the board (see Figure 6).

The Impact of the Hyperparameters The dataset size (see Figure 7 for the numerical factors/hyperparameters) seems most impactful for costate and time estimation. Interestingly, both reach better median results for the smaller dataset. It is unclear why this is, as the node coverage is the same regardless of dataset size. This may be a sign of overfitting, yet clearly determining that needs further investigation.

The batch size seems to be of limited impact. For costate and mass estimation, the median of both the

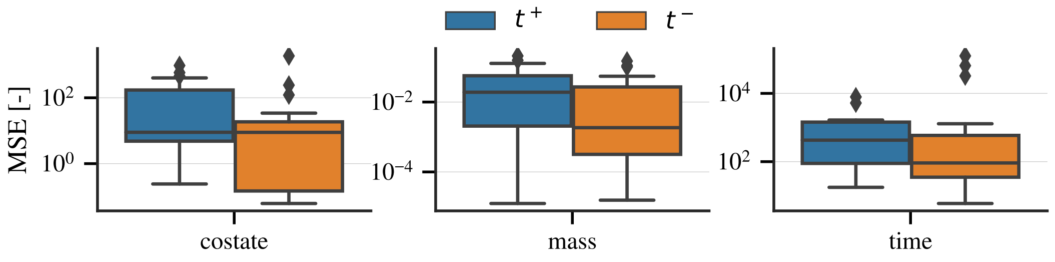


Figure 6. Statistical distribution of the MSE of the screening design for the different levels of the time direction factor.

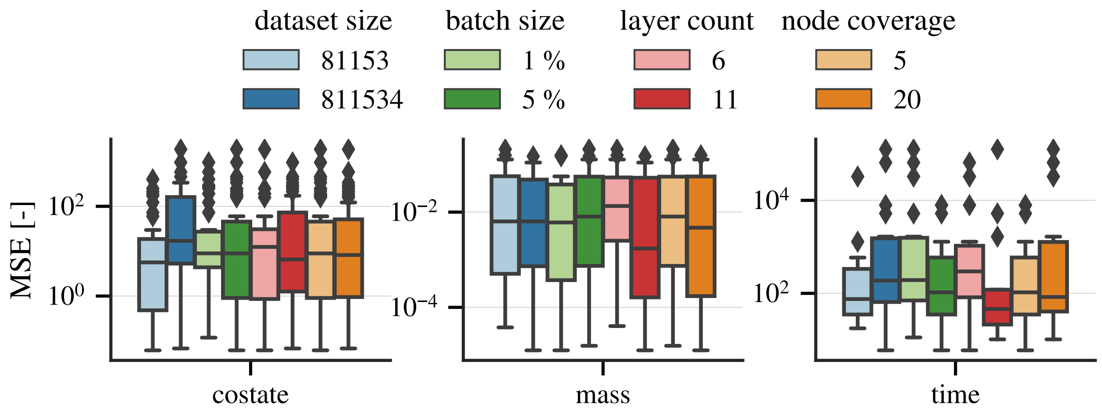


Figure 7. Numerical factors/hyperparameters and their effect as function of the target of the screening design.

results using the batch sizes with 1 % and 5 % of data points are effectively identical. For time estimation, the larger batch sizes perform better on average and in the minimum case. Interestingly, the training time does not actually decrease but increases slightly when going from a 1 % to a 5 % batch size while targeting the costates. The opposite is expected, as larger batch sizes mean fewer iterations to cover the entire dataset.

The layer count is, in this work, a factor describing the architecture of the network, where a higher number means more depth (more hidden layers) yet less width (fewer neurons in each layer). Keeping that in mind, one can clearly see that, in most cases, a deeper network is favorable. However, the best minimum for time estimation is reached with the shallower network. The setup used that keeps the number of trainable parameters roughly constant nonetheless has a downside that may influence the results. The size of both the feature and target vectors are varied between the different networks, thereby having more or fewer connections going to and from the same number of nodes in the hidden layers.

Also known as the data-to-parameter ratio, the node coverage factor embodies the number of data points that are used for every trainable parameter in the network (see the section on the sing design for the definition). Generally, the impact of the node coverage appears to be minimal. While both the median MSEs and the minima reached are very close, the median MSEs of the higher node coverage are always slightly lower. These results align with what is observed in other fields using statistical experiments, where a long-standing rule of thumb[53] has been for that ratio to be around 10. The range and even effect of this parameter are debated[54], however, which is also in line with the results achieved here.

The Impact of Additional Feature Parameters and Feature Modifications The specific angular momentum vector and the orbital energy are added as optional additional parameters. It is worth noting already that adding both the initial and final values to the feature space simultaneously never leads to a good result. The distribution of the results with different variants of the vis-viva energy and the specific angular moment vector can be found in Figure 8.

Not including the orbital energy at all leads to close to the best results in most cases. However, including the final orbital energy for mass estimation or the initial one for costate estimation led to slight improvements of the results. Its inclusion is thus rarely, if ever, worth considering.

The specific angular momentum ($\boldsymbol{h}$) vector consists of three orthogonal Cartesian components. It thus adds directional information, too. Including the $\boldsymbol{h}$ delta gives slightly better results when estimating time, while the final angular momentum has an order of magnitude improvement over the baseline for costate estimation. The mass estimation is not majorly influenced by the presence or absence of the specific angular momentum vector.

The effect POP exclusion and sine-cosine transforms have on the MSE can be seen in Figure 9. The

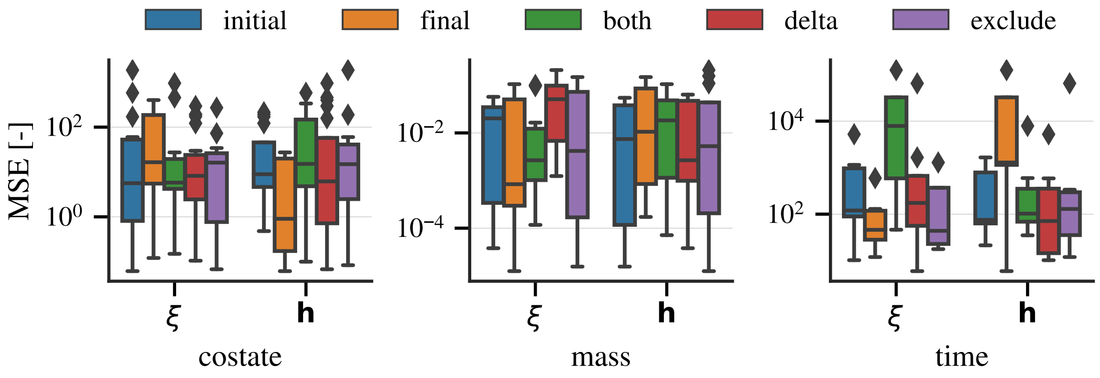


Figure 8. Effect of inclusion or exclusion of the specific angular momentum vector h and the vis-viva energy ξ on MSE of the screening design, plotted as a function of target type.

costates' value depends on the position on a given orbit, yet apparently, the POP does not seem to influence the performance either way. On the other hand, excluding the POP does result in lower MSE for mass and higher MSE for time estimation. The difference between using the angles directly and their sine and cosine instead is most beneficial when estimating the costates. While the transformation seems to aid the time estimation on average, it appears slightly disadvantageous when mass is estimated.

Follow-Up Experiments, A Closer Look at the Best Option Pairs

Using the best two levels from the screening design for every factor, this follow-up experiment aims to see whether the results obtained before hold up under more scrutiny (i.e., more data-point, more combinations). Additionally, by reverting to a two-level design, the potential presence of two-factor (or even three-factor) interaction effects can be investigated. It must be noted here, however, that by looking at the effects on a per-target basis, all effects described are already the interaction of the other factor and the target factor (or, at least, target type).

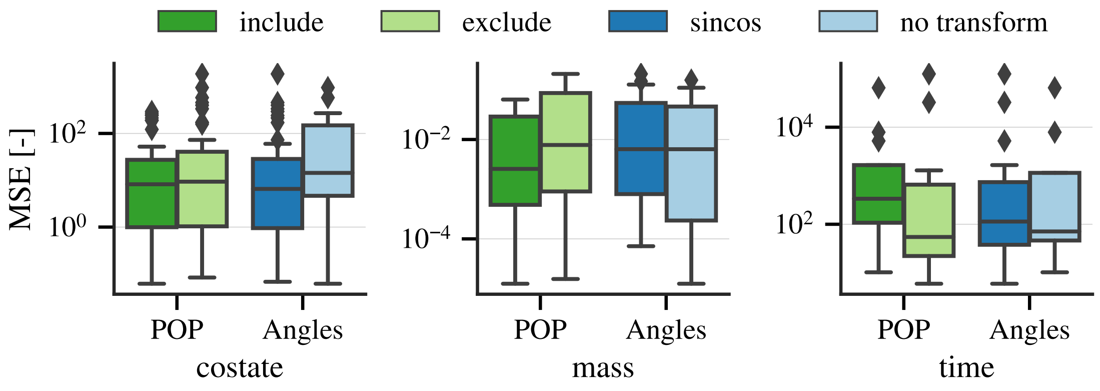


Figure 9. Effect of inclusion or exclusion of the angle sine-cosine transformation, and the inclusion or exclusion of the position-on-orbit parameter of the screening design, plotted as a function of target type.

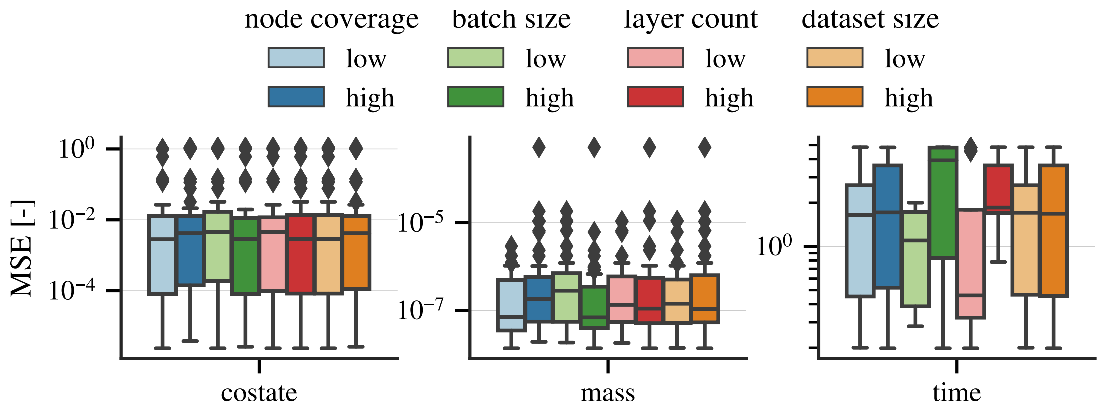


Figure 10. Final experiment results in terms of MSE, as a function of target type and numerical parameters. See Table 3 for the respective values for 'low' and 'high' used.

Changes in Factor Effects w.r.t. Screening Design By picking the two most promising options for all factors, the difference in results – both in terms of bulk of the results, and minima – has been considerably reduced. The results have also improved overall, as is to be expected. The numerical elements were iterated toward decreasing MSE, though they have yet to fully converge. The mostly-converged nature of the hyper-parameters is visible in Figure 10, where predominantly minim differences remain between the 'high' and 'low' levels. When time-targeting, one can still see significant differences, indicating that another iteration would likely be beneficial and, thus, that the chosen step size is not yet optimal. The remaining differences in main-factor effects are few.

For the costate estimation, the choice of dataset (negative rather than positive time) and the choice of target (Keplerian costates over MEE costates) make up the set of variables that still make a significant difference. The effects of these factors are plotted in Figure 11. As before, the costate scaling (normalizing or scaling for $\mathcal{H} = 0$) cannot be assessed here, as the magnitude of the values is different, at least partly resulting in the difference in MSE.

When targeting the mass, the choice of target (fuel mass rather than arrival mass) and the choice of feature state representation are the main remaining differences (visible in Figure 12).

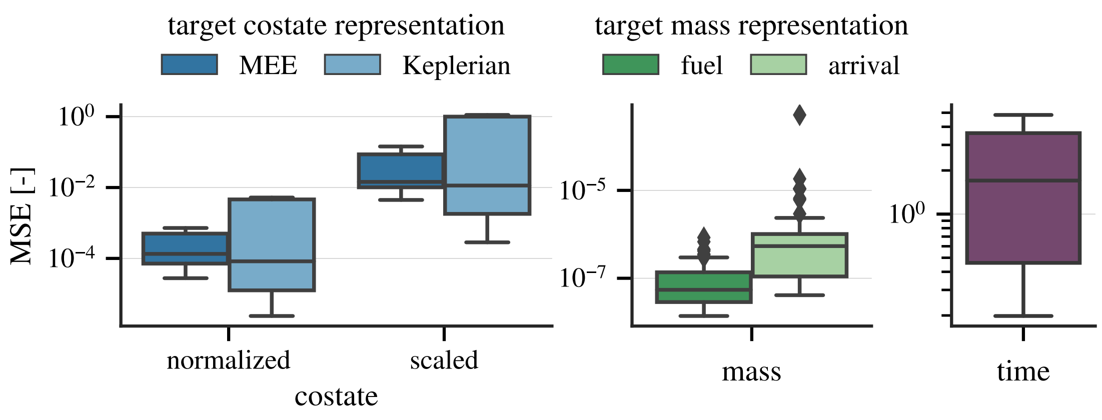


Figure 11. Final experiment results in terms of MSE, as a function of target type and specific target used.

The time approximation is still quite volatile and particular in its choices, partly due to some factors already having had distinctly favorable levels in the screening design. Nonetheless, even for those factors a second level was still included for consistency between the three sub-experiments. When looking at Figure 12, it can be seen that the choice of state representation in the feature set is most impactful for time estimation, where the Keplerian elements perform significantly better than the Modified Equinoctial Elements do.

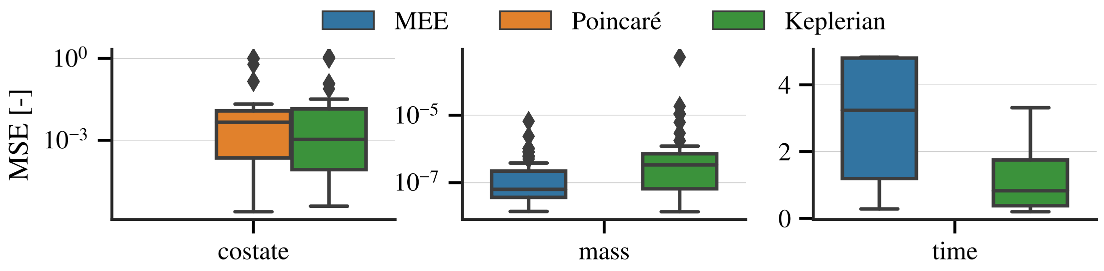


Figure 12. Final experiment results in terms of MSE, as a function of target type and state representation used in the feature vector.

Two-Factor Interactions The fact that the final design consists of only two-level factors can be exploited, as it trivially allows insight into two-factor effects, i.e., increased or decreased performance due to specific combinations of factor levels. A linear model is least-squares fit to the MSE of the different runs to gauge the effect of the individual factors and two-factor effects. The linear model contains all factors and all interactions between factors, and, as before, the design matrix is such that every 'low' value is -1, and every 'high' value is 1. This coding ensures that the coefficients of that linear model can be compared. The ten most significant (i.e., with the greatest absolute magnitude) coefficients, by target, can be found in Table 4. As before, the MSE that the coefficients act on is not comparable between target types as they describe different things.

Table 4. The most significant linear least-squares factor and two-factor interaction coefficients.

costate target	$\times 10^{-1}$	mass target	$\times 10^{-6}$	time target	$\times 10^{-1}$
$\lVert\boldsymbol{\lambda}\rVert$	1.04	Target	4.75	$\boldsymbol{x}_{\text{Feat.}} \times \text{N}_{\text{dataset}}$	9.08
Dir.	0.94	State Repr. × Dir.	4.65	$\text{N}_{\text{dataset}}$	9.04
Dir. × $\lVert\boldsymbol{\lambda}\rVert$	0.93	$\boldsymbol{x}_{\text{Feat.}}$	4.54	$\boldsymbol{x}_{\text{Feat.}}$	9.00
Dir. × Target	0.91	Batch Size	4.54	N_{layers}	5.07
$\text{N}_{\text{coverage}} \times \text{N}_{\text{layers}}$	0.90	POP × $\lVert\boldsymbol{\lambda}\rVert$	4.54	$\boldsymbol{x}_{\text{Feat.}} \times \text{N}_{\text{layers}}$	1.60
Target	0.83	Angles × $\text{N}_{\text{dataset}}$	4.52	$\text{N}_{\text{layers}} \times \text{N}_{\text{dataset}}$	1.53
$\lVert\boldsymbol{\lambda}\rVert$ × Target	0.82	Target × Batch Size	4.51	Angles	0.91
Target × State Repr.	0.37	$\boldsymbol{x}_{\text{Feat.}}$ × Target	4.47	$\boldsymbol{x}_{\text{Feat.}}$ × POP	0.79
Dir. × State Repr.	0.36	State Repr. × N_{layers}	4.39	Dir.	0.47
Batch Size × POP	0.36	$\boldsymbol{x}_{\text{Feat.}}$ × Batch Size	4.38	$\boldsymbol{x}_{\text{Feat.}}$ × Batch Size	0.46

When targeting the costate vector, the direction of integration appears to have strong interactions with several other factors. While the costate scaling itself and two-factor interactions appear in this table, those cannot be taken into account, as at least part of that effect is due to the different magnitude of the scaled and the unscaled costates. When going backwards in time, those second-order effects are thus favorable for Keplerian costates with the final state (i.e., Earth's orbit) as features. The final state as a feature, however, has a negative second-order effect when used to target Keplerian costates. Finally, looking at the numerical factors, a higher node coverage seems to go together with a lower layer count and vice versa, while a bigger batch size favors excluding the POP.

When mass is the estimation target, the coefficients all seem to have very similar magnitudes. The reason may be that all variants explored result in relatively low MSEs already, making those differences less meaningful. These coefficients are mostly noise, as can also be seen from the fact that the factor 'costate scaling' is included, although no costates are used anywhere in the data.

Moving on, the MSE least-squares fit as a function of time-targeting data differs from the other two, as the factors here have vastly differing magnitudes. Making the top of the list is, for the first time, an interaction effect rather than a main effect. It would thus seem that representing the features in MEE works better with a smaller dataset, whereas Keplerian works better with the larger one. It is not entirely clear why this is, but it is likely related to the fact, that the choice of Keplerian over MEE already has a significant benefit w.r.t. performance. The shortcomings of MEE could thus get emphasized with a larger dataset. The opposite effect is seen when looking at the layer count (although the magnitude of the impact is already almost an order of magnitude smaller). Here, MEE features have a positive interaction with an increased layer count. The interaction between layer count and dataset size is of similar magnitude to the aforementioned one. It implies that the smaller dataset favors a shallower network (and vice-versa), showing that the feature-coverage metric is not the be-all-end-all and could likely be surpassed by an alternative solution. The remaining interactions have coefficients with a magnitude significantly smaller than the top ones and will thus not be considered further.

Two two-factor interactions have some effect, and it seems those get some relevance once one has iterated toward an already well-performing design. Some of them are very small in magnitude, yet the options considered at this stage do not differ significantly anymore either, which explains at least part of it. Additionally, there are some contradictions in the data, making the case that three-factor interactions are potentially interesting. Given the high confounding that three-factor interactions experience at the current number of experimental runs, those have not been investigated further – yet doing just that could lead to some additional insight to maximize the potential of the neural networks.

CONCLUSION

The goal of this work was to find a recipe for when to use what feature when fitting feedforward neural networks to interplanetary transfer orbit data. Having trained 552 neural networks while carefully varying several factors, this question still cannot be conclusively answered. Yet, a step towards that answer was made, as the specific case considered within this work was investigated. One recommendation can be unequivocally made, irrespective of use-case: dimensional orbit data can hardly be learned by a neural network.

Now to the more specific conclusions. Generally, when in doubt, both Modified Equinoctial Elements (MEE) and Keplerian elements for feature data are an effective choice. More specifically, MEE appear to have a slight edge when estimating mass and time, while Poincaré elements are marginally better when estimating costates – the differences are minute, however. Including additional parameters, specifically the specific angular momentum vector and the vis-viva energy, is, on average, more likely to reduce the quality of the results for a similar network layout. They are thus best avoided. Other modifications of the feature vector, like only describing the orbit itself rather than the position on said orbit, do not help. Replacing the angles by their sine and cosine gives slightly more reliably low results only when estimating costates, yet even then is its impact not hugely significant.

Although feature engineering was the goal set at the onset of this work, the process revealed that the impact of the target choice could not be neglected either. When estimating costates, mapping them over to Keplerian costates before training the network is seemingly well worth it (about an order of magnitude of estimation mean squared error (MSE) can be gained). If that is not possible, MEE costates are a second valid option, while Cartesian costates should be avoided as targets. A similar observation can be made when approximating the mass, where a significant gain can be achieved (one order of magnitude MSE) by targeting the fuel mass rather than the dry mass. Finally, it can be noted costates are best approximated in terms of direction only and at the final rather than the initial state.

When it comes to the point of time to include in the feature, the final (or targeted) state seems to perform better when targeting the costates or the fuel mass. The transfer-time estimation, on the other hand, seems to benefit from having both the initial and final state in the feature space.

Other factors investigated, such as node count, layer count, data-to-parameter ratio, batch size, and dataset size, do not seem to have a momentous influence. Good results can be achieved with most settings, assuming that those are chosen reasonably. The results not majorly depending on the data set size is a positive sign, as it implies that the performance is likely consistent as the network is scaled up or down.

Finally, two-factor (or higher) interaction effects only seem to matter once one approaches a reasonably well-optimized network, as even there, they do not appear to have a notable influence.

BIBLIOGRAPHY

[1] H. S. Tsien, "Take-Off from Satellite Orbit," *Journal of the American Rocket Society*, vol. 23, no. 4, pp. 233–236, 1953.

[2] E. Stuhlinger, "Possibilities of Electrical Space Ship Propulsion," in *Bericht Über Den V. Internationalen Astronautischen Kongreß: Innsbruck, 5. Bis 7. August 1954*, F. Hecht, Ed. Berlin, Heidelberg: Springer, 1955, pp. 100–119.

[3] ——, "The Flight Path of an Electrically Propelled Space Ship," *Journal of Jet Propulsion*, vol. 27, no. 4, pp. 410–414, Apr. 1957.

[4] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[5] M. Haenlein and A. Kaplan, "A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence," *California Management Review*, vol. 61, no. 4, pp. 5–14, Aug. 2019.

[6] B. Dachwald and W. Seboldt, "Optimization of Interplanetary Rendezvous Trajectories for Solar Sailcraft Using a Neurocontroller," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, ser. Guidance, Navigation, and Control and Co-located Conferences. Monterey, California: American Institute of Aeronautics and Astronautics, Aug. 2002, AIAA 2002-4989.

[7] C. Ampatzis and D. Izzo, "Machine learning techniques for approximation of objective functions in trajectory optimisation," in *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI09)*. Pasadena, California, USA: Association for the Advancement of Artificial Intelligence, Jul. 2009.

[8] I. Carnelli, B. Dachwald, and M. Vasile, "Evolutionary Neurocontrol: A Novel Method for Low-Thrust Gravity-Assist Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 2, pp. 616–625, Mar. 2009.

[9] S. Pourtakdoust, F. Pazooki, and M. Fakhri Noushabadi, "A neuro-optimal approach for thrust-insensitive trajectory planning," *Aircraft Engineering and Aerospace Technology*, vol. 81, no. 3, pp. 212–220, Jan. 2009.

[10] A. Cassioli, D. Di Lorenzo, M. Locatelli, F. Schoen, and M. Sciandrone, "Machine learning for global optimization," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 279–303, Jan. 2012.

[11] P. Gómez Pérez, Y. Liu, and K. Cowan, "Global optimization of low-thrust interplanetary trajectories using a machine learning surrogate," in *AAS/AIAA Astrodynamics Specialist Conference*, ser. Advances in the Astronautical Sciences, vol. 175. Virtual, Online: AAS/AIAA, Aug. 2020, pp. 5147–5166, AAS 20-663.

[12] S. da Graça Marto and M. Vasile, "Many-objective robust trajectory optimisation under epistemic uncertainty and imprecision," *Acta Astronautica*, vol. 191, pp. 99–124, Feb. 2022.

[13] N. Ozaki, K. Yanagida, T. Chikazawa, N. Pushparaj, N. Takeishi *et al.*, "Asteroid Flyby Cycler Trajectory Design Using Deep Neural Networks," Feb. 2022.

[14] 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, pp. 1122–1135, May 2018.

[15] B. Gaudet, R. Linares, and R. Furfaro, "Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Powered Descent and Landing," Oct. 2018.

[16] P. Solano-López, A. B. Vilar, R. Gutiérrez-Ramon, and H. U. Cereijo, "Assessment of Machine Learning Techniques for Time-Optimal Landing Trajectories," in *Proceedings of the 9th European Conference for Aerospace Sciences*, Lille, France, Jun. 2022, p. 15 pages.

[17] C. E. Oestreich, R. Linares, and R. Gondhalekar, “Autonomous Six-Degree-of-Freedom Spacecraft Docking with Rotating Targets via Reinforcement Learning,” *Journal of Aerospace Information Systems*, vol. 18, no. 7, pp. 417–428, 2021.

[18] B. Dachwald, “Optimization of Interplanetary Solar Sailcraft Trajectories Using Evolutionary Neurocontrol,” *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 1, pp. 66–72, Jan. 2004.

[19] ——, “Optimization of very-low-thrust trajectories using evolutionary neurocontrol,” *Acta Astronautica*, vol. 57, no. 2, pp. 175–185, Jul. 2005, IAC-04-A.6.03.

[20] M. Pérez-Cutiño, F. Rodríguez, L. Pascual, and J. Díaz-Báñez, “Neural networks algorithms for ornithopter trajectory optimization,” in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2021, pp. 1665–1670.

[21] L. Stubbig and K. Cowan, “Improving the evolutionary optimization of interplanetary low-thrust trajectories using a neural network surrogate model,” in *AAS/AIAA Astrodynamics Specialist Conference*, ser. Advances in the Astronautical Sciences, vol. 175. Virtual, Online: AAS/AIAA, Aug. 2020, pp. 5127–5146, AAS 20-658.

[22] M. Shirobokov, S. Trofimov, and M. Ovchinnikov, “Survey of machine learning techniques in spacecraft control design,” *Acta Astronautica*, vol. 186, pp. 87–97, Sep. 2021.

[23] D. Izzo, C. Sprague, and D. Tailor, “Machine learning and evolutionary techniques in interplanetary trajectory design,” Sep. 2018.

[24] D. Izzo, M. Märtens, and B. Pan, “A survey on artificial intelligence trends in spacecraft guidance dynamics and control,” *Astrodynamics*, vol. 3, no. 4, pp. 287–299, Dec. 2019.

[25] R. Xie and A. G. Dempster, “An on-line deep learning framework for low-thrust trajectory optimisation,” *Aerospace Science and Technology*, vol. 118, p. 107002, Nov. 2021.

[26] G. Viavattene and M. Ceriotti, “Artificial Neural Networks for Multiple NEA Rendezvous Missions with Continuous Thrust,” *Journal of Spacecraft and Rockets*, vol. 59, no. 2, pp. 574–586, Mar. 2022.

[27] 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*, vol. 166, pp. 469–481, Jan. 2020.

[28] H. Li, F. Topputo, and H. Baoyin, “Autonomous Time-Optimal Many-Revolution Orbit Raising for Electric Propulsion GEO Satellites via Neural Networks,” Sep. 2019.

[29] H. Li, H. Baoyin, and F. Topputo, “Neural Networks in Time-Optimal Low-Thrust Interplanetary Transfers,” *IEEE Access*, vol. 7, pp. 156 413–156 419, 2019.

[30] G. Tang, F. Jiang, and J. Li, “Fuel-Optimal Low-Thrust Trajectory Optimization Using Indirect Method and Successive Convex Programming,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 4, pp. 2053–2066, Aug. 2018.

[31] D. Izzo and E. Öztürk, “Real-Time Guidance for Low-Thrust Transfers Using Deep Neural Networks,” *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 2, pp. 315–327, 2021.

[32] F. Jiang, H. Baoyin, and J. Li, “Practical Techniques for Low-Thrust Trajectory Optimization with Homotopic Approach,” *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 1, pp. 245–258, Jan. 2012.

[33] R. Bertrand and R. Epenoy, “New smoothing techniques for solving bang–bang optimal control problems—numerical results and statistical interpretation,” *Optimal Control Applications and Methods*, vol. 23, no. 4, pp. 171–197, 2002.

[34] M. J. H. Walker, B. Ireland, and J. Owens, “A set of modified equinoctial orbit elements,” *Celestial mechanics*, vol. 36, no. 4, pp. 409–419, Aug. 1985.

[35] L. S. Pontryagin, *The Mathematical Theory of Optimal Processes*, english ed ed., ser. Classics of Soviet Mathematics. New York: Gordon and Breach Science Publishers, 1986, no. v. 1.

[36] D. F. Lawden, *Optimal Trajectories for Space Navigation*, ser. Butterworths Mathematical Texts. London: Butterworths, 1963.

[37] M. Dowell and P. Jarratt, “A modified regula falsi method for computing the root of an equation,” *BIT Numerical Mathematics*, vol. 11, no. 2, pp. 168–174, Jun. 1971.

[38] P. E. Gill, W. Murray, and M. A. Saunders, “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization,” *SIAM Review*, vol. 47, no. 1, pp. 99–131, Jan. 2005.

[39] M. E. Muller, “A note on a method for generating points uniformly on n-dimensional spheres,” *Communications of the ACM*, vol. 2, no. 4, pp. 19–20, Apr. 1959.

[40] Delft High Performance Computing Centre, “DelftBlue Supercomputer (Phase 1),” https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022. [Online]. Available: https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1

[41] TU Delft, “Tudat – TU Delft Astrodynamics Toolbox,” 2022. [Online]. Available: https://github.com/tudat-team/tudat

[42] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[43] V. Vittaldev, E. Mooij, and M. C. Naeije, “Unified State Model theory and application in Astrodynamics,” *Celestial Mechanics and Dynamical Astronomy*, vol. 112, no. 3, pp. 253–282, Mar. 2012.

[44] G. R. Hintz, “Survey of Orbit Element Sets,” *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 785–790, May 2008.

[45] R. H. Lyon, “Geosynchronous Orbit Determination Using Space Surveillance Network Observations And Improved Radiative Force Modeling,” Master’s thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, Jun. 2004. [Online]. Available: https://core.ac.uk/display/80014985

[46] T. Fukushima, “Efficient Orbit Integration by Linear Transformation for Kustaanheimo-Stiefel Regularization,” *The Astronomical Journal*, vol. 129, no. 5, p. 2496, May 2005.

[47] E. Taheri, V. Arya, and J. L. Junkins, “Costate mapping for indirect trajectory optimization,” *Astrodynamics*, vol. 5, no. 4, pp. 359–371, Dec. 2021.

[48] D. C. Montgomery, *Design and Analysis of Experiments*, ninth edition ed. Hoboken, NJ: John Wiley & Sons, Inc, 2017.

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” Mar. 2016.

[50] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[51] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017.

[52] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Jan. 2017.

[53] E. Vittinghoff and C. E. McCulloch, “Relaxing the Rule of Ten Events per Variable in Logistic and Cox Regression,” *American Journal of Epidemiology*, vol. 165, no. 6, pp. 710–718, Mar. 2007.

[54] M. van Smeden, J. A. H. de Groot, K. G. M. Moons, G. S. Collins, D. G. Altman *et al.*, “No rationale for 1 variable per 10 events criterion for binary logistic regression analysis,” *BMC Medical Research Methodology*, vol. 16, no. 1, p. 163, Nov. 2016.

3 Trajectory Generation

The Section Paper/Data Generation (p. 4) already gives an overview of the method used to generate the data the networks are eventually trained on. This chapter expands upon that in three ways: Section 3.1 provides an overview of the literature and how it could and could not be used in this work. A complete derivation from the objective function of the optimal control problem to the throttle function found in Izzo and Öztürk (2021) is done in Section 3.2, including a break-down of the solutions' optimality. Finally, several numerical methods were employed to obtain a shooting function ready to be optimized by an existing non-linear optimizer. Those methods are only mentioned by name in Paper/Solving the Shooting Problem (p. 7), yet are fully expanded upon in Section 3.3. Lastly, Section 3.4 provides a closer look at the relationship between costates representing a valid transfer orbit and the initial guess converging to those costates.

3.1. Indirect Problem Solution Strategies in Literature

As established, the fuel-optimal Earth-Mars transfer cannot be solved analytically and must thus be solved numerically. While numerical solutions are never arbitrary, they do allow for a lot of creativity. In the following, a number of solutions available in literature will be briefly summarized. For an overview of the literature described in the first two subsections, please see Table C.6 (p. 77).

The methods described can be split into homotopic and non-homotopic. Section 3.1.1 walks through the methods that do use homotopy, while Section 3.1.2 briefly summarizes the ones that do not. Finally, Section 3.1.3 summarizes methods using Physics Informed Neural Networks, the Theory of Functional Connections, and combinations of the two.

3.1.1. Using Homotopy to Solve an Easier Problem First

One of the main factors contributing to the difficulty of convergence of fuel-optimal trajectories is the fact that the resulting control is bang-bang (either on or off, without anything in between). A prevalent method to get around this difficulty is to devise a homotopy path that allows to transform the solution of a smoother problem into the solution of the fuel-optimal problem.

A common starting point for this is the energy-optimal problem, but other ideas have been suggested too. T. Li et al. (2021) and Zhu et al. (2017) start from the minimum-thrust (i.e., constant-thrust) trajectory, for example, and Jawaharlal Ayyanathan and Taheri (2022) apply a smoothing on the throttle function itself. When the homotopy is applied, different approaches can be taken to get from the initial problem to the final one. The two main ideas here are quadratic smoothing, where a continuation parameter is directly used to go from fully quadratic (i.e., u^2) to a fully linear (i.e., u) running cost. Another approach found in literature with similar frequency is the logarithmic barrier. This is the one applied in this work (though merely in a smoothing capacity, as homotopy was not needed in the end; see Appendix D for an attempt at homotopy for the Earth-Venus problem). Note that while several cited publications broadly use homotopy, they still solve the problem in a variety of ways.

Of all methods considered in Table C.6, Izzo and Öztürk (2021) is the only one allowing for free final-time and free position-on-orbit parameter, which is why none of the others was attempted outright. However, aspects of those methods were tried on the path toward the algorithm eventually found. One of these aspects is a change of variable first found in Wang and Grant (2018) and subsequently used by T. Li et al. (2021) and Morelli et al. (2022). In this change of variables, the mass and the throttle are replaced by the logarithm of the mass and the acceleration, respectively. The initial costate of the mass logarithm is then just the fuel mass used – which can be found for the minimum-thrust case, as minimum-thrust implies constant thrust. This approach does not translate to a free-time solution.

The energy-optimal problem is more readily solvable due to its continuous nature, yet a good initial costate estimate is still needed – which is, once again, obtained in diverse ways. The costates can be estimated from the fact that an infinite thrust is the impulsive thrust case with a known solution (Zhu et al., 2017) or limiting the costate range by mapping them to a body-fixed state representation (Jawaharlal Ayyanathan & Taheri, 2022), for example. More often than not, these do rely on the fact that the transfer time is set a priori, however, or they reduce the feature set while also adding new, yet different, unknowns.

3.1.2. Non-Homotopic Methods Considered

Rosa Sentinella and Casalino (2006) uses a genetic algorithm to solve the minimum-fuel problem instead. Unfortunately, it is unclear what is actually being optimized for in that work, as the Hamiltonian does not contain a running cost of any kind (which would be 1 when optimizing for time, and u or $\dot{m}$ when optimizing for fuel-usage). All attempts trying to solve the shooting function outlined in Izzo and Öztürk (2021) using a genetic algorithm (or similar) were fruitless when using a minimum-fuel Hamiltonian, however.

Wang and Grant (2018) choose to instead apply a lossless convexification to the problem, allowing it to be solved by known methods. Using that method, a convergence within 2.5 s can be achieved, yet there is no clear path to adapt it to the problem at hand.

Finally, Epenoy (2023) transforms the problem by applying constraints on the thrust and coast arcs. This simplification strays from the mathematical optimality, yet when limiting the number of coast arcs to one, it can be solved in $\sim$1 s. This time quickly increases for more coast arcs, reaching up to $\sim$700 s for trajectories with four thrust and three coast arcs.

3.1.3. Physics-Informed Neural Networks and the Theory of Functional Connections

Other methods used in literature that were briefly considered here include Physics Informed Neural Networks (PINN) or, more specifically, Pontryagin Neural Networks (PoNN). This is sometimes combined with the Theory of Functional Connections (TFC) by using Extreme Learning Machines (ELM), which results in the Extreme Theory of Functional Connections (X-TFC) (D'Ambrosio et al., 2021).

PINNs have been attempted for optimal planar orbit transfer problems by Schiassi et al. (2022), besides others. The idea here being that the control problem is solved through unsupervised learning by having the dynamics and the boundary conditions encapsulated in the loss function. While promising, Schiassi et al. (2022) do not manage to get the PINN to deal with the competing objectives satisfactorily and thus make use of the TFC.

The Theory of Functional connections works by discretizing the trajectory using orthogonal polynomials (e.g., Chebyshev or Legendre polynomials; see Johnston et al., 2020 and D'Ambrosio et al., 2022), in addition to a component ensuring the boundary conditions are met. The polynomials' free parameters are then tuned through iterative least-squares, yielding a solution to the optimal control problem. Instead of polynomials, one can also make use of an Extreme Learning Machine. This has been done both using Single Input Multiple Output (SIMO) ELMs (D'Ambrosio et al., 2021) and using Multiple Input Single Output (MISO) ELMs (Schiassi et al., 2021). While Schiassi et al. (2022) mention that an expansion of the framework to 3D problems is a yet-to-be-solved problem, this method was considered on the path towards the eventual solution found. Convergence could not be achieved when attempting to adapt the X-TFC framework to 3D, however. The 3D convergence issue may be mitigable by tuning the iterative least-squares method, yet this could not be resolved within a reasonable time frame and was thus not followed up on.

3.2. Optimal Control Theory Applied to Interplanetary Transfers

This derivation is filling in the blanks between the initial definition of the optimal control problem as provided in Izzo and Öztürk (2021) and the resulting throttle function. While this section is essentially just following Pontryagin's (1986) original work (summarized in Section 3.2.1) applied to Izzo and Öztürk (2021) (laid out in Section 3.2.2), it also draws on the lectures by Prof. Hurák* as secondary source.

3.2.1. The Definition of Pontryagin's Minimum Principle

The problem, in terms of calculus of variations, is to find the function $\boldsymbol{u}(t)$ that minimizes the objective. That problem has already been briefly outlined in Paper/Optimal Control Problem (p. 4), yet is repeated here to be built upon. The problem is

$$\begin{aligned} \text{minimize} \quad & J = \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{u}(t))\,\mathrm{d}t \quad \text{where} \quad u \in [u_{\text{min}}, u_{\text{max}}] \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{D} \qquad , \\ & 0 = \boldsymbol{\Phi}_0(\boldsymbol{x}(t_0)) \\ & 0 = \boldsymbol{\Phi}_f(\boldsymbol{x}(t_f)) \end{aligned} \tag{3.1}$$

where J is the objective or cost functional; $\boldsymbol{u}(t)$ is the thrust vector-function; $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are matrices describing the dynamics; $\boldsymbol{x}$ is the trajectory itself; and $\dot{\square}$ the time-derivative. $\mathcal{L}$ is the running cost. The boundary values of the Optimal Control Problem (OCP) are represented by $\boldsymbol{\Phi}$. The Hamiltonian is then given by Equation 3.2. Note that the Hamiltonian is a function of t, $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$, and $\boldsymbol{\lambda}$. These are omitted for sake of clarity.

$$\mathcal{H} = \boldsymbol{\lambda}^{\mathsf{T}} \left[\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{D}\right] + \mathcal{L}(u(t)). \tag{3.2}$$

Following Pontryagin, the optimal control $\boldsymbol{u}^*$ (where $\square^*$ designates $\square$ associated with the trajectory minimizing J, or, equivalently, $\mathcal{H}$) minimizes the Hamiltonian, and thus, an equation for the optimal control vector emerges.

$$\mathcal{H}(t, \boldsymbol{x}^*, \boldsymbol{u}^*, \lambda^*) \le \mathcal{H}(t, \boldsymbol{x}^*, \boldsymbol{u}, \lambda^*) \quad \forall \boldsymbol{u} \in \mathcal{U} \quad \Longrightarrow \quad \boldsymbol{u}^* = \operatorname*{arg\,min}_{\boldsymbol{u}\in\mathcal{U}} \left[\boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{B}\boldsymbol{u} + \mathcal{L}(u)\right], \tag{3.3}$$

where $\mathcal{U}$ is the set of valid control vector functions $\boldsymbol{u}$, in case of the specific problem here, this set is limited by the engine's capabilities. When considering solar-electric propulsion, the distance to the Sun would be another factor reducing the set of possible thrust vector functions.

The direction of the vector minimizing that expression is given by Lawden's (1963) Primer Vector Theory, and is

$$\hat{\boldsymbol{u}}^* = -\frac{\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda}}{\|\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda}\|}. \tag{3.4}$$

By separating $\boldsymbol{u}$ into direction and magnitude, i.e., $\boldsymbol{u} = u \cdot \hat{\boldsymbol{u}}$, one can also obtain the magnitude of $\boldsymbol{u}$ minimizing the aforementioned expression by having a look of how it changes as a function of u

$$\frac{\partial}{\partial u}\left[u\boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{B}\hat{\boldsymbol{u}}^* + \mathcal{L}(u)\right] = -\|\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda}\| + \frac{\partial\mathcal{L}(u)}{\partial u}. \tag{3.5}$$

There are two possibilities from here on, which depend on $\partial\mathcal{L}/\partial u$. Those two possibilities are:

(1) The resulting relation contains u. In that case, the extremum is found by setting this expression to 0 and solving for u.

(2) The resulting relation does not contain u. In that case, the sign of $\partial\mathcal{L}/\partial u$ can be used to determine the ideal u:

negative If the derivative of the expression to be minimized w.r.t. u is negative, this means that the greater u, the smaller the expression. If u has an upper-bound, that upper-bound, u_{max} is thus the optimal value.

positive A positive derivative means, the smaller u, the smaller the expression. Here, if u has a lower bound, that lower bound is thus the optimal value.

zero This is then a so-called singular trajectory. These are not treated here. The interested reader is pointed towards works such as Bonnans et al. (2008), or more recently, Foroozandeh et al. (2018), which discuss singular trajectories and related solution strategies.

*Online Lectures for B3M35ORR, "Optimal and Robust Control," of Czech Technical University, Prague, found at https://www.youtube.com/playlist?list=PLMLojHoA_QPmRiPotD_TnfdUkglTexuqm [visited 30-04-2023]

3.2.2. Applying Pontryagin's Principle to the Interplanetary Low-Thrust Problem

The dynamics are described by (Izzo & Öztürk, 2021)

$$\dot{\boldsymbol{x}} = \frac{Tu(t)}{m}\boldsymbol{B}(\boldsymbol{x})\hat{\boldsymbol{i}}_\tau + \boldsymbol{D}(\boldsymbol{x}), \qquad \text{and} \qquad \dot{m} = -\frac{Tu(t)}{I_{sp}g_0}, \tag{3.6}$$

where $\hat{\boldsymbol{i}}_\tau Tu(t)/m = \boldsymbol{u}(t)$, with $\hat{\boldsymbol{i}}_\tau$ being the unit direction vector, T the maximum thrust and m the mass. This definition allows for a throttle setting $u \in [0, 1]$. The Hamiltonian is then, following Equation 3.2, given by

$$\mathcal{H} = \frac{Tu}{m}\boldsymbol{\lambda}^\top\boldsymbol{B}(\boldsymbol{x})\hat{\boldsymbol{i}}_\tau + \lambda_L\sqrt{\frac{\mu}{p^3}}w^2 - \frac{T}{I_{sp}g_0}\lambda_m u + \{u - \varepsilon \ln[u(1-u)]\}. \tag{3.7}$$

Applying Pontryagin, as before, one obtains an expression that is minimized by the optimal control function,

$$\boldsymbol{u}^* = \underset{\boldsymbol{u}\in\mathcal{U}}{\arg\min}\left[\frac{Tu}{m}\boldsymbol{\lambda}^\top\boldsymbol{B}(\boldsymbol{x})\hat{\boldsymbol{i}}_\tau - \frac{T}{I_{sp}g_0}\lambda_m u + \{u - \varepsilon\ln[u(1-u)]\}\right]. \tag{3.8}$$

Taking the first-derivative yields an expression describing how that expression changes as a function of $\boldsymbol{u}$, which can then be substituted by the primer vector, described in Equation 3.4,

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}u}\left[\frac{Tu}{m}\boldsymbol{\lambda}^\top\boldsymbol{B}(\boldsymbol{x})\hat{\boldsymbol{i}}_\tau - \frac{T}{I_{sp}g_0}\lambda_m u + \{u - \varepsilon\ln[u\cdot(1-u)]\}\right]\\ &= -\frac{T}{m}\|\boldsymbol{B}^\top\boldsymbol{\lambda}\| - \frac{T}{I_{sp}g_0}\lambda_m + 1 - \frac{\varepsilon\cdot(1-2u)}{u\cdot(1-u)}.\end{aligned} \tag{3.9}$$

When taking the limit $\varepsilon \to 0$, i.e., going towards a fuel-optimal solution, only one part of the expression does not vanish. This part is extracted into a switching function,

$$\mathrm{SF}(t) = -\frac{T}{m}\|\boldsymbol{B}^\top\boldsymbol{\lambda}\| - \frac{T}{I_{sp}g_0}\lambda_m + 1. \tag{3.10}$$

Finally, one can solve for u at the extremum ($\mathrm{d}/\mathrm{d}u = 0$),

$$0 = \mathrm{SF} - \frac{\varepsilon\cdot(1-2u^*)}{u^*\cdot(1-u^*)} \quad \Longleftrightarrow \quad u^* = \frac{\mathrm{SF} + 2\varepsilon \pm \sqrt{\mathrm{SF}^2 + 4\varepsilon^2}}{2\,\mathrm{SF}}. \tag{3.11}$$

As there are two solutions, a closer look at those solutions is not unimportant: By taking the first-order derivative, there is no guarantee that a minimum is found. It could just as well be a maximum or even a saddle point. The + variation can be rewritten as

$$u_+^* = \frac{\mathrm{SF}+2\varepsilon+\sqrt{\mathrm{SF}^2+4\varepsilon^2}}{2\,\mathrm{SF}}\cdot\frac{\mathrm{SF}+2\varepsilon-\sqrt{\mathrm{SF}^2+4\varepsilon^2}}{\mathrm{SF}+2\varepsilon-\sqrt{\mathrm{SF}^2+4\varepsilon^2}} = \frac{2\varepsilon}{\mathrm{SF}+2\varepsilon-\sqrt{\mathrm{SF}^2+4\varepsilon^2}}. \tag{3.12}$$

The limit of Equation 3.12 as SF $\to 0^\pm$ goes to ($\pm$) infinity. This does not seem optimal for a bounded control variable. Additionally, when looking at

$$\lim_{\varepsilon\to 0} u_+^* = \lim_{\varepsilon\to 0}\frac{1}{\frac{\mathrm{SF}}{2\varepsilon}+1-\sqrt{\frac{\mathrm{SF}^2}{4\varepsilon^2}+1}} = \lim_{\varepsilon\to 0}\frac{1}{\frac{\mathrm{SF}}{2\varepsilon}+1-\left|\frac{\mathrm{SF}}{2\varepsilon}\right|} = \begin{cases}1 & \text{for SF} \geq 0\\ 0 & \text{for SF} < 0\end{cases}, \tag{3.13}$$

and keeping in mind that SF is the derivative of the function that is to be minimized w.r.t. u, one can see that this $u(t)$ will actually maximize the Hamiltonian instead. This is thus a maximum and not a minimum.

Looking at the $-$ variation next,

$$u_-^* = \frac{\mathrm{SF}+2\varepsilon-\sqrt{\mathrm{SF}^2+4\varepsilon^2}}{2\,\mathrm{SF}}\cdot\frac{\mathrm{SF}+2\varepsilon+\sqrt{\mathrm{SF}^2+4\varepsilon^2}}{\mathrm{SF}+2\varepsilon+\sqrt{\mathrm{SF}^2+4\varepsilon^2}} = \frac{2\varepsilon}{\mathrm{SF}+2\varepsilon+\sqrt{\mathrm{SF}^2+4\varepsilon^2}}, \tag{3.14}$$

this has a finite value for SF $\to 0^\pm$. Taking the limit once more,

$$\lim_{\varepsilon\to 0} u_-^* = \lim_{\varepsilon\to 0}\frac{1}{\frac{\mathrm{SF}}{2\varepsilon}+1+\sqrt{\frac{\mathrm{SF}^2}{4\varepsilon^2}+1}} = \lim_{\varepsilon\to 0}\frac{1}{\frac{\mathrm{SF}}{2\varepsilon}+1+\left|\frac{\mathrm{SF}}{2\varepsilon}\right|} = \begin{cases}0 & \text{for SF} > 0\\ 1 & \text{for SF} \leq 0\end{cases}, \tag{3.15}$$

where, this time, the result is one that actually minimizes Equation 3.9 – and thus also the Hamiltonian. This solution coincides with what is described in Izzo and Öztürk (2021).

3.3. Numerical Methods Used

Where Paper/Figure 1 already described the shooting function in broad strokes, this chapter will expand upon the numerical methods used within said algorithm. The shooting function itself contains three iterative numerical methods, which are outlined in this section.

The first one of those methods is the scaling of the initial costate guess, as described in Paper/Solving the Shooting Problem (p. 7). Section 3.3.1 describes that costate scaling, which ensures that the Hamiltonian is zero (within a given tolerance). The second one is the one ensuring that the point closest to the target orbit is targeted. Here, too, the integrator used is numerical and thus cannot be solved exactly. How it is solved instead is shown in Section 3.3.2. Finally, the loop ensuring that the optimizer does not need to find a fitting dry mass value, too, is described in Section 3.3.3. The initial mass (i.e., the mass at the start of propagation; here: the dry mass) is iterated upon so that the final mass (i.e., the mass at the end of propagation; here: the wet mass) reaches the desired value at the point of minimum $\Delta\mathcal{M}$.

3.3.1. Numerical Costate Scaling for Zero Hamiltonian

The conditions for the optimal control trajectory used in this work state that $\mathcal{H}|_{t=t_0} = 0$, but also that $\mathrm{d}\mathcal{H}/\mathrm{d}t = 0$. The latter is not really relevant when backward shooting, as the former is then directly accounted for – yet it is still useful when verifying the correctness of a given implementation.

This equation initially seems solvable with a straightforward numerical scheme: Start with $\beta = 1$, calculate the resulting u, and use Paper/Equation 16 to calculate the following guess for β. However, this scheme is unstable, and β does not always converge this way. To ensure its convergence, an adaptive under-relaxation method is used (Murty, 1983). As the ideal relaxation factor is not known, over-relaxation is not possible. The converge rate using under-relaxation turned out to be acceptable. The new β value is a weighted sum of the previous one and the new guess, where the weighting factor is the ratio of the previous two changes in β. The full algorithm is described in Algorithm 1, including the equation for β.

Algorithm 1: Scale costate for zero Hamiltonian.

```
Input  : State vector $\boldsymbol{x}$, mass $m$, costate guess that is to be scaled $\boldsymbol{\lambda}_{guess}$
Output : $\lambda$ scaled for $\mathcal{H} = 0$
Function ScaleCostate($\boldsymbol{x}_i, m, \boldsymbol{\lambda}_{\text{est}}$):
    $\beta \leftarrow 1,\ \omega_\beta \leftarrow 1,\ \delta_{\beta,0} \leftarrow \text{NaN},\ \beta_{raw,0} \leftarrow \text{NaN}$
    repeat
        $u \leftarrow$ ComputeThrottle($\boldsymbol{x}_i, m, \beta\boldsymbol{\lambda}_{\text{est}}$)   /* compute $u$ with current state and $\beta$ estimate */
        $\beta_{raw,1} \leftarrow \left(\frac{Tu}{m}\|\boldsymbol{B}^\top\boldsymbol{\lambda}_{\text{est}}\| - \lambda_{\text{est},L}\sqrt{\frac{\mu}{p^3}}w^2\right)^{-1}\left[-\frac{T}{I_{sp}g_0}\lambda_m u + u - \varepsilon\ln\left[u(1-u)\right]\right]$
        $\beta \leftarrow (1-\omega_\beta)\cdot\beta + \omega_\beta\cdot\beta_{raw,1}$   /* $\beta \leftarrow$ weighted sum of previous and current value. */
        $\delta_{\beta,1} \leftarrow \beta_{raw,0} - \beta_{raw,1}$
        if $\delta_{\beta,0}$ is not NaN then
            $\omega_\beta \leftarrow \left|\frac{\delta_{\beta,1}}{\delta_{\beta,0}}\right|$   /* update relaxation factor based on rate-of-change ratio. */
        $\delta_{\beta,0} \leftarrow \delta_{\beta,1}$   /* update elements for next iteration. */
        $\beta_{raw,0} \leftarrow \beta_{raw,1}$
    until $|\delta_{\beta,1}| < \epsilon_\beta$   /* stop when the change in $\beta$ is below the set threshold. */;
    return $\beta\cdot\boldsymbol{\lambda}_{\text{est}}$
```

A few more observations regarding Algorithm 1 and the costate scaling in general can be made.

Firstly, when $\epsilon \to 0$, the whole algorithm is not iterative anymore but can be solved exactly. The thrust when reaching the target orbit must be one, as the only element varying without thrust is the true longitude L, which is not part of the boundary conditions. Furthermore, $\epsilon \to 0$ implies a bang-bang thrust function which can only be zero or one and hence must be one in this case.

Secondly, the stability increase is achieved by one main assumption: The last guess is already close to the ideal value. Using that assumption, one can then successively reduce the amount that the new scaling factor is changed, thereby making appropriate steps to reduce the Hamiltonian. The tolerance used in this work is $\epsilon_\beta = 10^{-12}$.

3.3.2. Exact Targeting of the Closest Point

Once the costate vector has been scaled using the numerical method outlined in Section 3.3.1, it is then propagated. Usually, when propagating, one targets a certain value of the independent variable, which is most often time. In this case, however, the transfer time is unknown. The perfect transfer could exist with any value of t, and as backward propagation is done, only the state has to match – there are no constraints on the costates, and the Hamiltonian is constant. For that reason, the minimum $\Delta\mathcal{M}$ (w.r.t. the target orbit) visited by the integrator is used as a starting point. From there on, the derivative of $\Delta\mathcal{M}$ is iteratively taken to get the minimum within an acceptable tolerance. This is outlined in Algorithm 2. The iterative method used is a modified secant method (Dahlquist & Björck, 2008).

Algorithm 2: Propagate until minimum $\Delta\mathcal{M}$.

```
Input  : A starting extended state $\boldsymbol{x}_i$ (MEE state, mass, and corresponding costates), a final mass
         $m_f$ (i.e., the spacecraft's wet mass for backward propagation), a target state $\boldsymbol{x}_t$ (MEE, only
         first five elements), and an integrator.
Output : The closest state found up until $m = m_f$.

Function IntegrateUntilClosestToTarget($\boldsymbol{x}_i, m_f, \boldsymbol{x}_t$, integrator):
    $\boldsymbol{x}_{\text{running}} \leftarrow \boldsymbol{x}_i$, $\boldsymbol{x}_{\text{closest}} \leftarrow \boldsymbol{x}_i$, $\Delta\mathcal{M}_{\text{min}} \leftarrow \infty$
    repeat
        $\boldsymbol{x}_{\text{running}} \leftarrow$ integrator→doControlledStep()      /* assuming variable-step integrator */
        $\Delta\mathcal{M}_{\text{running}} \leftarrow \|\mathcal{M}(\boldsymbol{x}_{\text{running}}) - \mathcal{M}(\boldsymbol{x}_t)\|_2$
        if $\Delta\mathcal{M}_{\text{running}} < \Delta\mathcal{M}_{\text{min}}$ then
            $\Delta\mathcal{M}_{\text{min}} \leftarrow \Delta\mathcal{M}_{\text{running}}$
            $\boldsymbol{x}_{\text{closest}} \leftarrow \boldsymbol{x}_{\text{running}}$
    until $m_{\boldsymbol{x}_{\text{running}}} < m_f$;      /* until fuel is used/final mass is reached */;

    $t_{\text{step}} \leftarrow$ integrator→lastStepSize()
    $\delta_{\Delta\mathcal{M},0} \leftarrow \frac{\mathrm{d}}{\mathrm{d}t}\|\mathcal{M}(\boldsymbol{x}_{\text{closest}}) - \mathcal{M}(\boldsymbol{x}_t)\|_2$      /* see Equation A.16 */
    integrator→goTo($\boldsymbol{x}_{\text{closest}}$)

    while $\delta_{\Delta\mathcal{M},0} > \epsilon_{\Delta\mathcal{M}}$ do
        $\boldsymbol{x}_{\text{closest}} \leftarrow$ integrator→doStep($t_{\text{step}}$)
        $\delta_{\Delta\mathcal{M},1} \leftarrow \frac{\mathrm{d}}{\mathrm{d}t}\|\mathcal{M}(\boldsymbol{x}_{\text{closest}}) - \mathcal{M}(\boldsymbol{x}_t)\|_2$      /* see Equation A.16 */
        $t_{\text{step}} \leftarrow t_{\text{step}} \cdot \frac{\delta_{\Delta\mathcal{M},1}}{\delta_{\Delta\mathcal{M},0} - \delta_{\Delta\mathcal{M},1}}$
        $\delta_{\Delta\mathcal{M},0} \leftarrow \delta_{\Delta\mathcal{M},1}$
    return $\boldsymbol{x}_{\text{closest}}$;
```

Algorithm 2 also exhibits convergence performance. When generating trajectories, the $\delta_{\Delta\mathcal{M}}$ used was 10^{-8}. The derivative of $\Delta\mathcal{M}$, found in Section A.2, was derived using a Computer Algebra System (CAS).

3.3.3. Iterating the Initial Mass to Ensure the Closest Point is the Starting Point

Section 3.3.1 and 3.3.2 have set the baseline for this function, as both costate scaling and targeting of the closest point are required for this last part. Here the other pieces are put together so that the optimizer merely optimizes the costates themselves. The initial mass is chosen (i.e., dry mass for backward propagation) so that the final mass (i.e., wet mass for backward propagation) is reached exactly at the point closest to the target orbit. A *regula falsi* algorithm – also known as the false-position method (Dahlquist & Björck, 2008) – is employed for that purpose. Specifically, a variant of that algorithm is used, called the *Illinois Algorithm* (Dowell & Jarratt, 1971). The function whose root – or zero – is to be found is the amount of unused fuel left when reaching the point closest to the target. The specific algorithm used is outlined in Algorithm 3.

The base idea of the regula falsi algorithm follows a similar train of thought as the bisection method: The root of the function is found by iteratively approaching from both the positive and the negative side. The one change it makes to the basic regula falsi is that it forces both sides to get closer, whereas regula falsi may only improve one side for several iterations. The latter is not always problematic, as it can still result in fast convergence. In this specific case, however, it meant that the solution took a significant number of additional iterations. The additional iterations happened because the delta between the positive and negative solution was too large to accurately approximate the derivative, as only one of the two sides moved closer to the root while the other was kept constant. The Illinois Algorithm takes care of that issue by forcing the side of the opposite sign to move if it has not done so during two consecutive iterations.

Algorithm 3: Iterate mass for closest hit at $m = m_f$.

```
Input  : An initial state $\boldsymbol{x}_i$ (MEE elements), an initial mass $m_i$, a targeted final mass $m_f$, a target
         state $\boldsymbol{x}_t$ (MEE, only first five elements), an initial costate guess $\boldsymbol{\lambda}_{guess}$, and an integrator
Output : The initial state yielding the closest hit at $m = m_f$
Function IterateMassForClosestHit ($\boldsymbol{x}_i, m_i, m_f, \boldsymbol{x}_t, \boldsymbol{\lambda}_{guess}$):
    while $|m_f - m_{\boldsymbol{x}_{\text{closest}}}| > \delta_m$ do
        $\boldsymbol{x}_{\text{closest}} \leftarrow$ IntegrateUntilClosestToTarget(ScaleCostate($\boldsymbol{x}_i, m_i, \boldsymbol{\lambda}_{guess}$))
        $m_{closest} \leftarrow m_{\boldsymbol{x}_{\text{closest}}}$
        $m_{\text{fuel,unused}} \leftarrow m_f - m_{closest}$, $m_{\text{fuel,used}} \leftarrow m_{closest} - m_i$
                                                /* first run, or second with non-opposing signs */
        if $m_{i,A}$ is NaN and $m_{i,B}$ is NaN or ($m_{i,B}$ is NaN and $m_{\text{fuel,unused}} \cdot m_{\text{fuel,unused},A} > 0$) then
            $m_{i,A} \leftarrow m_f - (1 - 0.1 \cdot \text{sign}(m_{\text{fuel,unused}})) \cdot m_{\text{fuel,used}}$        /* encourage opposite sign */
            $m_{\text{fuel,unused},A} \leftarrow m_{\text{fuel,unused}}$
        else                                                                     /* regula falsi algorithm */
            if $m_{\text{fuel,unused}} \cdot m_{\text{fuel,unused},A} > 0$ then
                $m_{\text{fuel,unused},A} \leftarrow m_{\text{fuel,unused}}$
                $m_{i,A} \leftarrow m_i$
                if $m_{\text{fuel,unused},last} \cdot m_{\text{fuel,unused}} > 0$ then              /* Illinois algorithm */
                    $m_{\text{fuel,unused},B} \leftarrow 0.5 \cdot m_{\text{fuel,unused},B}$
            else
                $m_{\text{fuel,unused},B} \leftarrow m_{\text{fuel,unused}}$
                $m_{i,B} \leftarrow m_i$
                if $m_{\text{fuel,unused},last} \cdot m_{\text{fuel,unused}} > 0$ then              /* Illinois algorithm */
                    $m_{\text{fuel,unused},A} \leftarrow 0.5 \cdot m_{\text{fuel,unused},A}$
            $m_i \leftarrow \dfrac{m_{i,A} \cdot m_{\text{fuel,unused},B} - m_{i,B} \cdot m_{\text{fuel,unused},A}}{m_{\text{fuel,unused},B} - m_{\text{fuel,unused},A}}$
        $m_{\text{fuel,unused},last} \leftarrow m_{\text{fuel,unused}}$
    return $[\boldsymbol{x}_i, m_i, \boldsymbol{\lambda}_{guess}]$
```

3.4. Statistical Analysis of Convergence and Initial Guesses

The previous section demonstrated the effort necessary to achieve convergence. Yet, even then, only every third initial guess converges. This chapter attempts to paint a picture of why that may be, by looking at the relationship between the initial guess provided to the optimizer and the resulting trajectory. This convergence rate is within the range of what can be found in literature (see Table C.6).

When looking at the distribution of departure and arrival points, color-coded by their $\Delta\mathcal{M}$ at Earth's orbit (see Figure 3.1), one can see a clear pattern emerge: Convergence seems to be easier to achieve at certain regions along the orbit. Additionally, there seem to be parts of Earth's orbit that no minimum-fuel trajectory departs from. Furthermore, the meaning of a $\Delta\mathcal{M}$ that is not in the order of $\mathcal{O}(10^{-6})$ can be seen,

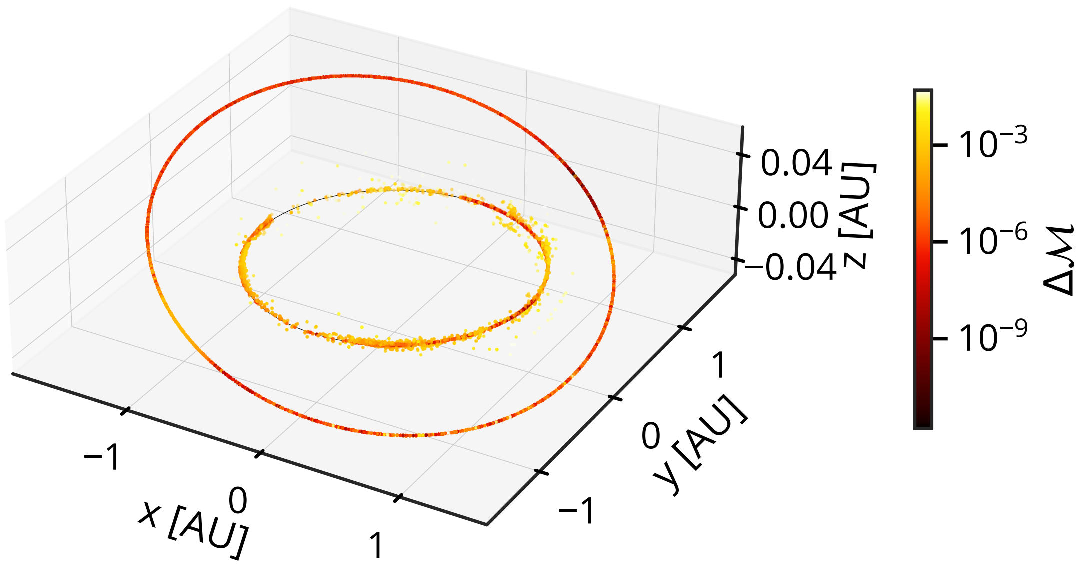


Figure 3.1: Visualization of Earth-Mars trajectory start and end-points, colored by their deviation from the target orbit.

as those dots show up as a yellow cloud next to Earth's orbit.

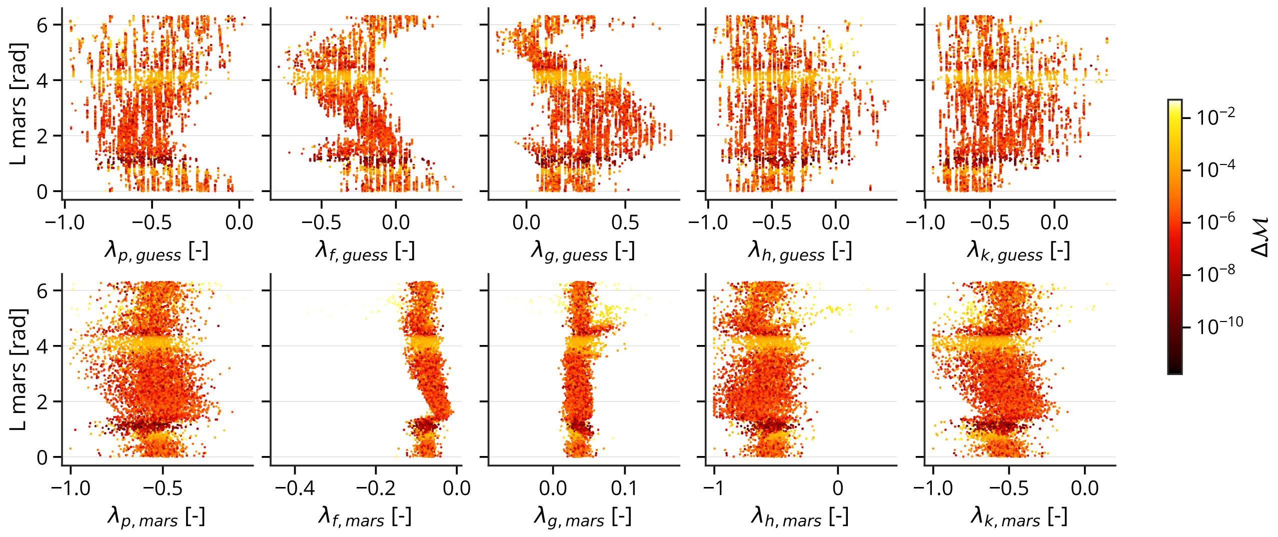


Figure 3.2: Visualization of initial guesses and the corresponding converged value, color-coded by their deviation from the target orbit.

Figure 3.2 is a visualization of the initial guesses for the different costate elements, compared to the value those guesses then converged to. Here, a clear relationship between the convergence potential and the value of true longitude is also visible. The converged values can be seen to be quite a bit narrower in bandwidth than the initial guesses that enable the optimizer to find those same values. When considering that the initial guesses range from -1 to 1, only specific ranges turn out to be feasible. While this is a valuable observation, this is all that can be seen from this data. There is no other clear relationship between the initial guess and what value it may converge to besides the true longitude.

Given that ostensibly only the direction of the initial costates is relevant, as they are scaled for $\mathcal{H} = 0$ in any case, looking purely at that direction is a logical next step. One way to map the points on the surface of a 4-sphere (a 4-sphere is a sphere in 5D space, the surface of which is 4D) to four variables is to use hyperspherical coordinates. These can be defined in different ways, yet this analysis uses the definition found in Equation 3.16.

$$\begin{bmatrix} \cos\hat{\theta}_1 \\ \cos\hat{\theta}_2 \\ \cos\hat{\theta}_3 \\ \tan\hat{\phi} \end{bmatrix} = \begin{bmatrix} \dfrac{\lambda_{g_e}}{\sqrt{\lambda_p^2 + \lambda_{f_e}^2 + \lambda_{g_e}^2}} \\ \dfrac{\lambda_{h_e}}{\sqrt{\lambda_p^2 + \lambda_{f_e}^2 + \lambda_{g_e}^2 + \lambda_{h_e}^2}} \\ \dfrac{\lambda_{k_e}}{\sqrt{\lambda_p^2 + \lambda_{f_e}^2 + \lambda_{g_e}^2 + \lambda_{h_e}^2 + \lambda_{k_e}^2}} \\ \dfrac{\lambda_{f_e}}{\lambda_p} \end{bmatrix} \tag{3.16}$$

Here, $\hat{\theta}_{\{1-3\}}, \hat{\phi}$ represent the four angles of the hyperspherical coordinates.

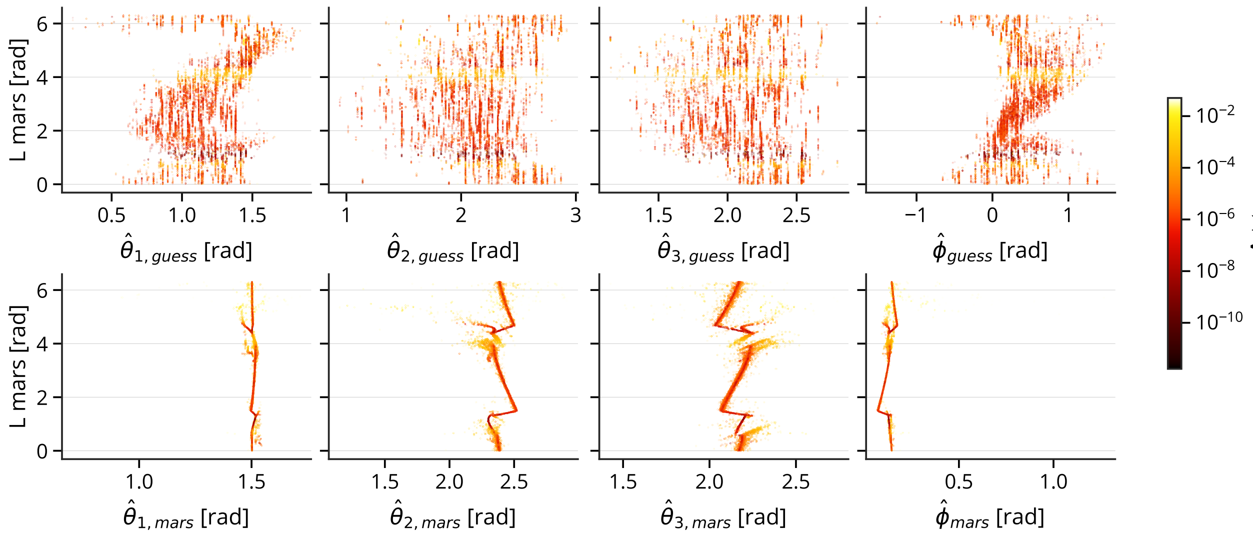


Figure 3.3: Visualization of the directions of several initial guesses and the corresponding converged direction (described employing hyperspherical elements, see Equation 3.16), color-coded by their deviation from the target orbit.

Applying that transformation to both the initial guess and the resulting costates yields noteworthy results, as seen in Figure 3.3. The converged costates collapse to a single line with two discontinuities, roughly π true longitudes apart. Even more interestingly, however, this pattern is not at all visible in the initial guesses. While the actual region of fuel-optimal trajectories appears to be very narrow, this still does not inform on successful initial guesses. Following the discovery of the directionality of the converged trajectories, an interpolator (as a function of the true longitude L) was fit through the lines seen in Figure 3.3 – yet convergence was not consistently achieved. The reason for this is unclear and would require further investigation. Additionally, using the hyperspherical elements instead of the raw costate vector did not improve the optimizer's convergence performance. That lack of improvement is unexpected, as that substitution represents a reduction in the input space of the shooting function from five to four variables.

4 Implementation Details

This chapter provides additional implementation details that are relevant yet not crucial to fit into the paper part of this work. For implementation details regarding the trajectory generation part of this work, see Chapter 3.

Section 4.1 provides an overview of the state-transformation functions used. This is followed by Section 4.2, which gives an overview of the Design of Experiments method, providing the analysis framework. Finally, Section 4.3 references the most relevant tools that made this work possible.

4.1. Overview of State Transformations Used

This section provides a concise overview of all orbital state transformations used, found in Equation 4.1 to 4.7. Most of the transformations are taken from existing literature, and in case they are not, the derivation is presented in Appendix A. Given that one of the additional feature modifications used (see Paper/Additional Features (p. 9) for a complete overview) is the sine-cosine transformation of the angles present in the feature set, this section will also point out which elements are angles.

- Kustaanheimo-Stiefel: Derivative elements are linear combinations of Cartesian elements. None of the KS elements are angles.

$$\begin{aligned}
\vartheta_1 &= s_{x_1}\sqrt{\frac{r+|x_1|}{2}}, & \vartheta_1' &= \frac{1}{2}\left(\vartheta_1 v_1 + \vartheta_2 v_2 + \vartheta_3 v_3\right),\\
\vartheta_2 &= \sqrt{\frac{2}{r+|x_1|}}\frac{x_2}{2}, & \vartheta_2' &= \frac{1}{2}\left(-\vartheta_2 v_1 + \vartheta_1 v_2\right) s_{x_1},\\
\vartheta_3 &= \sqrt{\frac{2}{r+|x_1|}}\frac{x_3}{2}, & \vartheta_3' &= \frac{1}{2}\left(-\vartheta_3 v_1 + \vartheta_1 v_3\right) s_{x_1},\\
& & \vartheta_4' &= \frac{1}{2}\left(-\vartheta_3 v_2 + \vartheta_2 v_3\right),
\end{aligned} \tag{4.1}$$

where

$$r = \sqrt{x_1^2 + x_2^2 + x_3^2}, \qquad s_{x_1} = \begin{cases} +1 & \text{for } x_1 \geq 0 \\ -1 & \text{for } x_1 < 0 \end{cases}.$$

- Modified Equinoctial Elements: L is clearly, and by definition, an angle. f_e, g_e, h_e, and k_e, are all functions of trigonometric transformations of angles (Walker et al., 1985),

$$\begin{aligned}
p &= a(1-e^2), & h_e &= \tan\frac{i}{2}\cos\Omega,\\
f_e &= e\cos(\omega+\Omega), & k_e &= \tan\frac{i}{2}\sin\Omega,\\
g_e &= e\sin(\omega+\Omega), & L &= \Omega+\omega+\theta,
\end{aligned} \tag{4.2}$$

and hence, not angles.

- Cartesian: The Cartesian elements all have physical meaning, representing distances and velocities in the $\mathbb{R}^3$ space. Sine/cosine transforms do not make sense here.
- Spherical: Following the convention for naming and order from Tudat 2022 (note that the flight-path angle from Tudat 2022 is the complementary angle to the flight-path angle for Chobotov, 2002; this matches the definition in Wakker, 2015):

$$\begin{aligned} r &= \sqrt{x^2+y^2+z^2}, & v &= \sqrt{\dot{x}^2+\dot{y}^2+\dot{z}^2}, \\ \sin\Phi &= \frac{z}{r}, & \sin\gamma &= \frac{\boldsymbol{r}\cdot\boldsymbol{v}}{\|\boldsymbol{r}\|\|\boldsymbol{v}\|}, \\ \tan\Lambda &= \frac{y}{x}, & \tan\psi &= \frac{((\boldsymbol{r}\times\boldsymbol{v})\times\boldsymbol{r})\times((\boldsymbol{r}\times\hat{\boldsymbol{e}}_z)\times\boldsymbol{r})\cdot\hat{\boldsymbol{r}}}{((\boldsymbol{r}\times\boldsymbol{v})\times\boldsymbol{r})\cdot((\boldsymbol{r}\times\hat{\boldsymbol{e}}_z)\times\boldsymbol{r})} \\ & & &= \frac{(\dot{y}x-\dot{x}y)\sqrt{x^2+y^2+z^2}}{-\dot{x}xz-\dot{y}yz+\dot{z}x^2+\dot{z}y^2}, \end{aligned} \tag{4.3}$$

where Φ is the latitude (or the declination, Chobotov, 2002); Λ is the longitude (or the right ascension, Chobotov, 2002); γ is the flight-path angle; ψ is the heading (or azimuth, Chobotov, 2002) angle; and $\hat{\boldsymbol{e}}_z$ is the unit vector in positive z direction. Clearly, all but r and v are angles and thus make the sin-cos transformation an obvious one.

- Keplerian: The Keplerian, or classical orbital, elements (a, e, ω, Ω, θ, i), are, except for the semi-major axis a and the eccentricity e all angles. Sine-cosine is thus appropriate here.
- USM-EM (exponential mapping, TU Delft, 2022) is the USM chosen.
  - USM7 uses quaternions, similar to what the Kustaanheimo-Stiefel elements do.
  - USM6 uses the Modified Rodrigues Parameters (MRP). While MEE only contains elements of the Classical Rodrigues Parameters (Peterson et al., 2022), their difference is only in using $\tan i/4$ for MRP vs $\tan i/2$ for CRP. This shifts the singularity, yet that is all.
  - USM-EM, following the definition from Facchinelli (2018), TU Delft (2022), and Vittaldev et al. (2012) and using the implementation provided by Tudat (for $\|\boldsymbol{e}_{\text{usm}}\| > 0$, otherwise the $\boldsymbol{e}_{\text{usm}}$ vector is 0):

$$\begin{aligned} C &= \sqrt{\frac{\mu}{a\cdot(1-e^2)}} = \sqrt{\frac{\mu}{p}}, & e_{\text{usm},1} &= \frac{\|\boldsymbol{e}_{\text{usm},m}\|}{\sin\|\boldsymbol{e}_{\text{usm},m}\|}\sin\frac{i}{2}\cos\frac{\Omega-\omega-\theta}{2}, \\ R_{f1} &= -eC\sin(\omega+\Omega), & e_{\text{usm},2} &= \frac{\|\boldsymbol{e}_{\text{usm},m}\|}{\sin\|\boldsymbol{e}_{\text{usm},m}\|}\sin\frac{i}{2}\sin\frac{\Omega-\omega-\theta}{2}, \\ R_{f2} &= eC\cos(\omega+\Omega), & e_{\text{usm},3} &= \frac{\|\boldsymbol{e}_{\text{usm},m}\|}{\sin\|\boldsymbol{e}_{\text{usm},m}\|}\cos\frac{i}{2}\sin\frac{\Omega+\omega+\theta}{2}, \\ S_{\text{usm}} &= \begin{cases} 1 & \text{if } \cos\frac{i}{2}\cos\frac{\Omega+\omega+\theta}{2} < 0 \\ 0 & \text{otherwise} \end{cases}, & \|\boldsymbol{e}_{\text{usm},m}\| &= 2\arccos\left(\cos\frac{i}{2}\cos\frac{\Omega}{2}\right). \end{aligned} \tag{4.4}$$

  In other words: $\|\boldsymbol{e}_{\text{usm},m}\|$ is an angle (arccos), and therefore so are all e elements.

- Cylindrical: Using the definition provided by Stubbig (2019)*,

$$r_{\text{cyl}} = \sqrt{x^2+y^2}, \qquad v_{r,\text{cyl}} = \frac{x\dot{x}+y\dot{y}}{\sqrt{x^2+y^2}}, \qquad \tan\theta_{\text{cyl}} = \frac{y}{x}, \qquad v_{\theta,\text{cyl}} = \frac{x\dot{y}-y\dot{x}}{\sqrt{x^2+y^2}}, \tag{4.5}$$

and z and its derivative are identical to the Cartesian definition. The angle is evident.

- Delaunay: The derivation for the Delaunay conversion is given in Section A.3.1 and restated here for completeness. From Modified Equinoctial Elements (undefined for $e >= 1$):

$$\begin{aligned} L_d &= \sqrt{\mu a} = \frac{h}{\sqrt{1-e^2}}, & \omega &= \begin{cases} \text{atan2}(g_e, f_e) - \Omega & \text{for } e > 0 \\ 0 & \text{for } e \approx 0 \end{cases}, \\ h &= \sqrt{\mu p}, & \Omega &= \text{atan2}(k_e, h_e), \end{aligned} \tag{4.6}$$

*Stubbig's (2019, App. A) equation for the cylindrical azimuth angle has the numerator and denominator swapped. This is likely a mistake, as it would be inconsistent with the definition used in the same work.

$$H_d = h \cdot \left(\frac{2}{h_e^2 + k_e^2 + 1} - 1 \right), \qquad M = \operatorname{atan2}\left(\sqrt{1-e^2} \sin\theta, e + \cos\theta \right) - e \frac{\sqrt{1-e^2}\sin\theta}{1 + e\cos\theta},$$

where L_d and H_d are Delaunay canonical variables; θ, the true anomaly, is found by using $L = \omega + \Omega + \theta$; and M is the mean anomaly. The first three elements (i.e., mean anomaly, argument of periapsis, RAAN) are angles. The remaining elements are not.

- Poincaré: The derivation of the Poincaré element conversion is given in Section A.3.2,

$$\begin{aligned} L_d &= \sqrt{\mu a} = \sqrt{\frac{\mu p}{1-e^2}}, & u_p &= k_e \sqrt{2L_d \sqrt{1-e^2}(1 + \cos(i))}, \\ \xi_p &= g_e \sqrt{\frac{2L_d}{1 + \sqrt{1-e^2}}}, & v_p &= h_e \sqrt{2L_d \sqrt{1-e^2}(1 + \cos(i))}, \\ \eta_p &= f_e \sqrt{\frac{2L_d}{1 + \sqrt{1-e^2}}}, & \ell &= M + \omega + \Omega, \end{aligned} \tag{4.7}$$

where ξ_p and η_p are the eccentric variables; u_p and v_p are the oblique variables (Smart, 1953); and ℓ is the mean longitude. Only the last element seems to relevantly be an angle.

4.2. Design of Experiments – Background and Implementation

Design of Experiments is the discipline of systematically describing a problem in terms of a statistical experiment, thereby allowing to draw conclusions on the impact the different parameters of an experiment have. It is called ‘Design of Experiments’ because the statistical experiment is carefully crafted to obtain the sought-after results with a minimum number of experiment runs required. While being one alternative of many to explore a design space, this framework allows one to do so in a precise and controlled way.

This section will start by briefly explaining how fractional factorial designs come about for two-level designs, define the required vocabulary, and explain how fractional factorial designs are generally chosen. The following part explains how multi- and mixed-level fractional factorial designs are generated within this work. Finally, the design resolution of the employed designs will be proven and discussed.

4.2.1. Fractional Factorial Designs

In Design of Experiments, fractional factorial designs are a set of experiments that do not contain an experimental run for every possible combination of factors. So, for n 2-level factors, any design that contains less than 2^n experiments is a fractional factorial design. In such a design, one can distinguish between two types of factors (Montgomery, 2017): (1) The factors of interest, i.e., the ones that are being investigated, and (2) nuisance factors. The latter are not directly part of the experiment but may still have an impact on it. Nuisance factors can be controllable, uncontrollable, and measurable (also called ‘covariates’) or uncontrollable and unmeasurable. If a nuisance factor can be controlled, there are two main ways to deal with it. The first option is to contain the effect of the nuisance through blocking (only evaluating results within a block that all have the same value as the given covariate). Alternatively, the other option is to promote the controllable nuisance to factor and include it in the design.

With the factors defined, a design can be built. The aforementioned full factorial is trivial: Taking all permutations of all factors is all it needs. When doing so, the variables are coded -1 and 1. The choice is arbitrary for categorical factors, but for numerical ones, coding the lower value to -1 and the higher value to 1 is usually sensible. This coding has the advantage that the confounding is immediately visible: Whenever two columns in the design matrix are identical, their effect can no longer be distinguished. In a full factorial design, all effects and all interaction effects are clearly distinguishable.

Fractional factorial designs are generated by deliberately confounding (or aliasing) certain columns. When doing that, the main effects are usually preserved. At first, only higher-order effects start being confounded (i.e., cannot be distinguished anymore). The design resolution is used to describe the level of effects that can still be uniquely distinguished. The design resolution describes the length of the shortest word in the defining relationship. Without going into too much detail, the defining relationship describes the aliasing or confounding structure of the fractional factorial design. In practice, the main resolutions of interest are (Montgomery, 2017):

Resolution III The main effects are not confounded with each other but with two-factor interactions.

Resolution IV The main effects are not confounded with each other or with two-factor interactions. The two-factor interactions are confounded with each other.

Resolution V The two-factor interactions are not confounded with each other, yet they are confounded with three-factor interactions.

Generally, for any number of factors, more than one configuration exists with the same number of experimental runs. Publicly available catalogs of minimum aberration designs for a given number of two-level factors and number of runs[†] do exist, however. Minimum aberration designs are designs of resolution R, that minimize the number of main effects confounded with $(R-1)$-factor interactions, two-factor interactions confounded with $(R-2)$-factor, and so on. Thus, while having the same design resolution, the number of affected (interaction) factors is minimized from the main factors up.

4.2.2. Mixed-Level Fractional Factorial Designs

Mixed-Level fractional factorial designs are more complex because aliasing two columns is more complex than multiplying their coded value anymore. It can no longer be done with the raw columns, as the coding itself is more involved. Additionally, in the case used within this work, some factor combinations do not exist (removing the position-on-orbit parameter is impossible if the feature set used does not contain one, after all). Therefore, several steps must be taken to get from a basic set of factors and their level to an experiment plan.

The most straightforward step to return to a two-level design is to dummy-code the multi-level factors. Before, the difference in magnitude between the two levels of a factor is what was encoded, and by using -1 and $+1$ for all factors, the effect on the metric is comparable. That same behavior can be recovered by just giving every possible value of a multi-level factor its column in the design matrix. That is, every column but one – the missing one represents the base level. Where the two-level comparison was the values resulting from a factor at the *high* level to its *low* level, now all levels are compared to a base level.

Dummy-coding does work, yet more is needed to exclude some combinations from the experimental design. For that to be possible, a different approach was chosen. It goes as follows:

- All factors with limited combinations are combined into one factor with levels representing all those combinations.
- The number of combinations present in a full factorial design under those circumstances is evaluated. This number is simply the product of every factor's level count.
- The number of binary columns required to represent those combinations is evaluated, which is the rounded-up $\log_2$ of the combination count.
- A regular, two-level fractional factorial design is created, the columns of which are binary values encoding the factor levels.
- Of that resulting design, the rows encoding values out of range (due to rounding up the column count) are dropped.
- The actual factor values at every row of this design matrix can be retrieved again by converting the encoded values back.

This procedure has one major drawback: It does not guarantee any design resolution. The lack of guarantee itself is not necessarily a problem as long as it can be ensured that the actual results are of sufficient design resolution. The final design matrix generated with the aforementioned procedure is a test plan. It will, eventually, have to be dummy-coded to evaluate the design. This final, dummy-coded matrix is the one that the linear model is fit on, and thus the one where confounding is to be avoided. The design resolution, or at least the minimum design resolution, can be tested quite trivially: By just generating the extended matrix, i.e., including all interactions one is interested in – and checking for duplicate columns. In other words, one evaluates that extended matrix' rank.

Using the procedure above, very little re-arranging of factors is required to obtain a Resolution III design. Testing for higher resolution designs is tricky, as the number of interaction columns gets large rapidly as higher-order interactions are considered.

[†]The catalogue used in this work can be found at (Grömping, 2014): https://cran.r-project.org/package=FrF2.catlg128 [visited 20-06-2023]

4.3. Tools Used

This work was only possible with a plethora of primarily open-source tools. This section highlights the most important ones, indicating how they have been instrumental in achieving the goal of this work.

- *Tudat (TU Delft, 2022).* Tudat provides a wide array of tools to solve all kinds of astrodynamical problems. While the propagator part was not performant enough for the use-case of this thesis, its extensive collection of integrators and conversion functions between state representations was extremely useful and is the backbone of the shooting method employed.
- *TensorFlow (Abadi et al., 2016).* All neural networks were trained using TensorFlow, which has the advantage of just running on about everything and making the best use of any available hardware. The number of networks that required to be trained could not have been achieved without it – or a similar open-source machine learning library.
- *SNOPT (Gill et al., 2005).* A state-of-the-art optimization library used to obtain convergence of the shooting problem towards the boundary conditions. This library is, unfortunately, not open source – though educational binaries and licenses are provided for research purposes.
- *pykep (Izzo et al., 2020).* While only used for verification purposes and not very successful at that, pykep was still helpful in providing another verification path.
- *pybind11 (Jakob et al., 2017).* Pybind11 allowed offloading all performance-critical code (i.e., optimization and propagation) to performant C++ code while keeping the convenience and interactivity of Python. Its simplicity was a pleasure to work with and also allowed for seamless interoperation between the Python part, C++ part, Tudat, Tudatpy, and pagmo.
- *PuLP (Dunning et al., 2011).* The PuLP library provides a Python interface to a number of linear and mixed-integer programming optimizers. In this work, it was used to optimize the column assignment of fractional factorial designs, ensuring that the most important interaction effects were individually distinguishable and not confounded with anything. The GLPSOL LP/MPI Solver from GLPSOL (Makhorin, 2020) was used as a backend.
- *pagmo (Biscani & Izzo, 2020).* Pagmo was used to (sadly, unsuccessfully) prototype different optimization methods – and eventually, it was used as interface for SNOPT. It allowed to optimization problem to be very defined in a straightforward manner, as the interface provided by SNOPT itself is everything but.
- *OpenMPI (Gabriel et al., 2004)* and *mpi4py (Dalcin & Fang, 2021).* This open implementation of the message-passing interface (MPI) enabled getting the most out of the trajectory generation and neural network training on the DelftBlue cluster. Moreover, it also proved invaluable when training multiple neural networks on the same machine in parallel.
- *DelftBlue (Delft High Performance Computing Centre, 2022).* The DelftBlue supercomputer was the only way to obtain appropriately sized datasets. Every interaction with the DelftBlue team was very productive, and using the cluster itself was a positive experience throughout.

5 Verification and Validation

In this chapter, the verification and validation process of the implementations used within this work is demonstrated. First, the verification framework is presented in Section 5.1. This is followed by the verification of the data generation section in Section 5.2. After that, the state and costate representation mapping functions are verified in Section 5.3 and Section 5.4, respectively. Finally, Section 5.5 discusses additional helper functions needed and how those were verified.

5.1. Verification of Methods Limited by Floating Point Arithmetic

Verification here is done in various ways throughout this chapter. Wherever possible, testing the implementations needed for this work is automated using Pytest (Krekel et al., 2004). This means that conversion functions and similar are automatically evaluated across their valid input range, and the resulting output is compared against known verified values. Due to the limitation of floating-point arithmetic, Pytest provides an `approx` function which allows some deviations in the results to pass the tests. Many of the functions, especially conversions between different state representations, have singularities, however. This means that when the full valid input range is evaluated, part of it will be close to the singularity and, therefore, less accurate. Hence, a different helper class is implemented, `approx_percentile`. This new function does essentially the same as Pytest's `approx`, yet instead of checking for relative and absolute errors on each and every value, it checks whether a given percentile of the data fits the definition.

The default value here is the 3σ value of a normal distribution: 99.73 %. The relative error is the one mostly considered, as it is the one limited by machine precision. Where the expected value is zero, the absolute error is used instead. In the following, all elements that could be tested in this way will state the percentile used and explain why it may be lower than the 3σ probably. An additional way to show, in most cases, that the error is indeed due to the limited precision of floating-point numbers is to show that it gets lower when one uses `long double`'s – in this case `Intel x86-64` 80 bit doubles.

5.2. Data Generation

The data generation itself uses a lot of external tools that are assumed to be validated. One of those is the adaptive-step Runge-Kutta-Fehlberg 8(9) integration implemented in Tudat. What is not part of any available library and does need validation is the implementation of the differential equations describing the fuel-optimal control problem, first mentioned in Paper/Optimal Control Problem (p. 4).

5.2.1. Comparison to Published Companion Data

Given that the description of the basic dynamics used is taken from Izzo and Öztürk (2021), the most straightforward comparison is one done against data from that publication. While it is rare for data to be published together with the research, it was, in this case (see Öztürk & Izzo, 2020), and can thus be used to verify the accuracy of the implementation done in the context of this work.

The data published consists of several trajectories from Earth to Venus, and along each trajectory, 100 points have been sampled. Having only the state and thrust vectors at some timestamps precludes a direct check of the derivative functions, which is what is implemented. However, it allows for a check of the thrust direction and throttle magnitude computed at a given (extended) state, which is likely the most essential aspect to be correct.

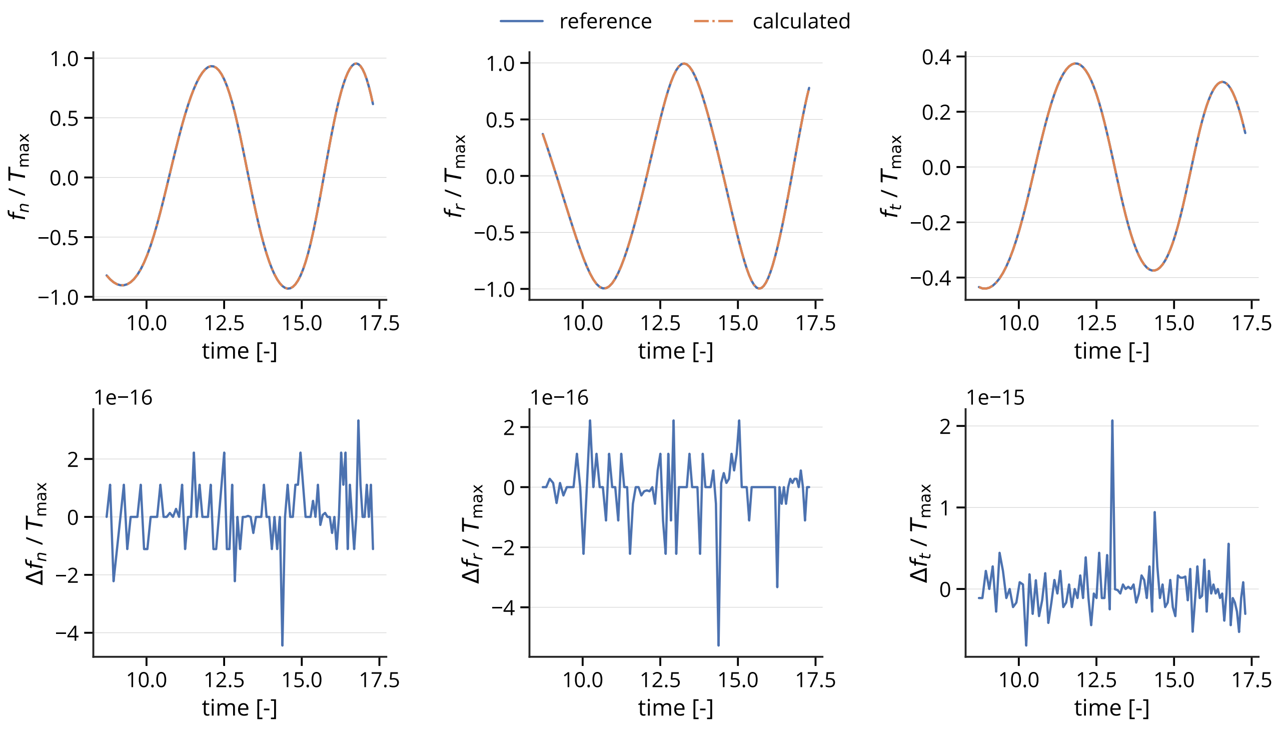


Figure 5.1: Computed thrust direction elements compared to trajectory with ID ’4’ from Öztürk and Izzo (2020). Top row: both throttle functions overlaid; bottom row: difference between the calculated and the reference throttle values.

This comparison is made in Figures 5.1 and 5.2 for the thrust direction and the throttle setting, respectively. Figure 5.1 shows that the thrust direction is as perfect as it gets and is in the order of machine precision for 64 bit floating-point numbers. This is, however, not the case for the throttle setting. Figure 5.2 shows that the error is fluctuating with the throttle function – which is to be expected. The throttle function for a fuel-optimal trajectory is a step function, which may explain slight discrepancies around the changes in value. This does not, however, explain why there is a delta several orders of magnitude below the machine epsilon in between those switching points. Clearly, there is a problem, yet it could not be solved – and thus, a secondary approach had to be taken to verification, which is done in the next section.

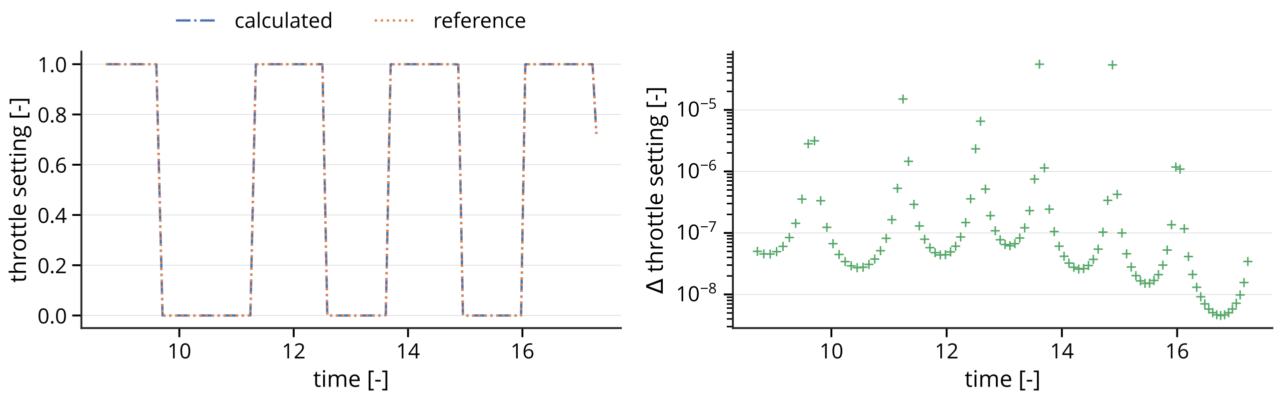


Figure 5.2: Computed throttle setting compared to trajectory with ID ’4’ from Öztürk and Izzo (2020).

5.2.2. Setup of Alternative Implementations for Verification

Section 5.2.1 demonstrated a problem in the thrust function, which does not match as closely as one would expect. From that alone, it is unclear whether that is a problem. A comparison of propagated trajectories needs to be done to check that, which is happening in this section. Some foreshadowing is inevitable at this point: Yes, it is a problem. The propagated trajectories are thus not only going to be compared against the reference data – the difference in thrust-magnitude is a given, after all. They will also be compared to publicly available implementations.

Note that the different implementations used provide only one thing: The state derivatives. All trajectories will be propagated using Tudat's Runge-Kutta-Fehlberg 5(6) integrator, using a relative and absolute error tolerance of 10^{-12}, to make the resulting trajectories as comparable as possible.

Tudat (TU Delft, 2022) does not provide an implementation for Pontryagin-optimal minimum-fuel trajectories. It does provide a general-purpose state propagation implementation, which can be used to verify that that part of the implementation is correct. Additionally, Tudat allows extending its state vector with mass and custom statement elements. Tudat can thus propagate the costates along with its verified state propagation implementation.

The Cowell propagator is used in this setup where Tudat propagates its own state. A propagator in Equinoctial Elements is implemented in Tudat, yet Cowell is preferred, as the acceleration calculations in Tudat happen in Cartesian. A non-Cartesian propagator introduces a lot of back-and-forth state transformations, which, in turn, introduce both inaccuracies and performance overhead.

Pykep (Izzo et al., 2020) is an astrodynamics toolbox by ESA, which the author of the reference paper used publishes. It provides an extended state space propagation of Pontryagin-optimal trajectories, yet it does so in Cartesian rather than MEE elements. Converting the state part between the elements is trivial, and mapping costates between representations is not much more involved either (see Paper/Transformation of Orbital Costates (p. 8)).
Now that everything is set up, it is time to propagate trajectories and compare the results, which is done in the next section.

5.2.3. Verification Against Other Publicly Available Implementations

The first aspect to verify, now that alternative implementations – of at least part of the state space propagation – are present, is to see whether the state derivatives themselves are correct. This zero-thrust use case is excellent, as this is the one that is fully implemented directly by both Tudat and pykep. The result of this is shown in Figure 5.3. Note that the choice of integrators in this section was made based on the fact that the performance of the integration of dynamics as provided by Tudat in combination with other (higher-order) choices increased runtimes by orders of magnitudes.

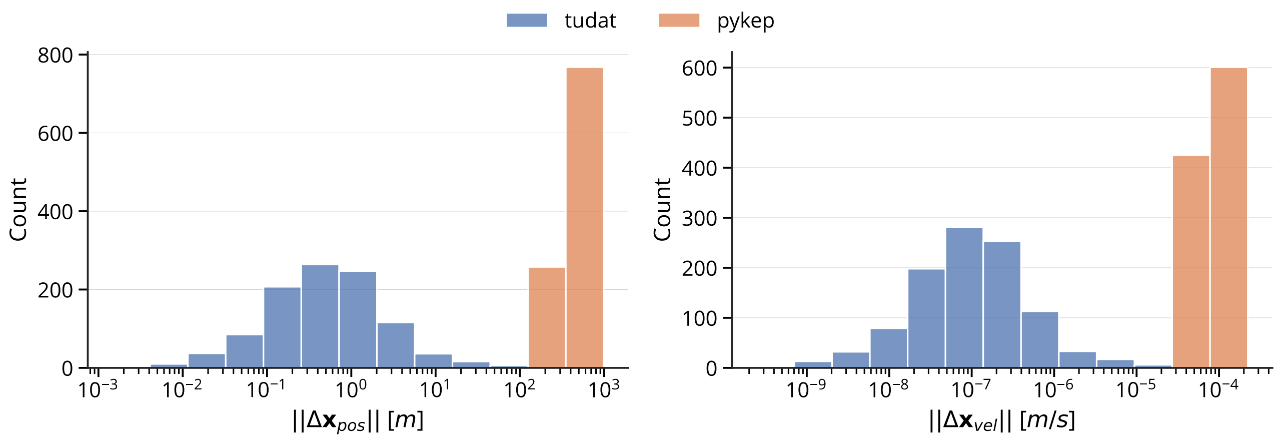


Figure 5.3: No-thrust state comparison of this work against a propagation done with Tudat and pykep, using Tudat's RKF5(6) implementation with absolute and relative tolerances of 10^{-12}. The first ~1000 trajectories of Öztürk and Izzo (2020) are used.

As can clearly be seen: Whether Tudat is propagating the costate or the costate is propagated using this work's implementation of Izzo and Öztürk (2021), the final state matches rather closely. The Error in the state is less than one meter, on average, for a transfer from Earth to Venus. The error w.r.t. pykep is slightly higher, which is potentially caused by the state translations and costate mappings. Those transformations do not have machine-level precision (due to their complexity in the involved transcendental functions). It must be noted, however, that even an error of order $\mathcal{O}(1\,\text{km})$ is still, in the scope of interplanetary trajectories, quite acceptable. It is something that could easily be corrected during transfer if detected.

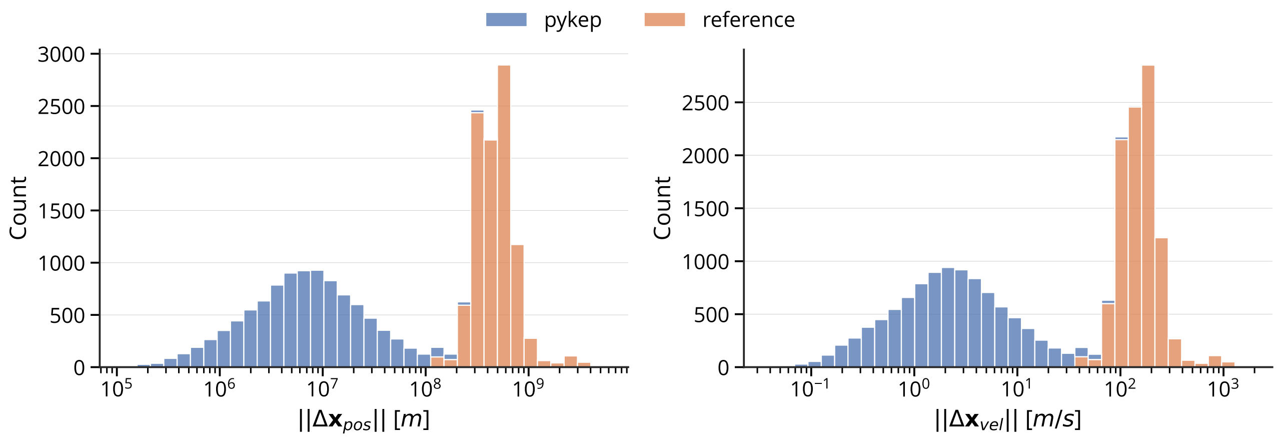


Figure 5.4: With-thrust state comparison of this work against a propagation done with pykep and the reference trajectories, using Tudat's RKDP8(7) implementation with absolute and relative tolerances of 10^{-12}. The initial states of the first ~10,000 trajectories of Öztürk and Izzo (2020) are used.

The next step is to enable thrust, the results of which can be seen in Figure 5.4, which compares the results from a propagation following a number of orbits from Öztürk and Izzo (2020). In this case, the comparison between Tudat and this implementation does not add anything, as, due to lack of native support, Tudat is provided with the same derivative function for the costate part of the extended state vector.

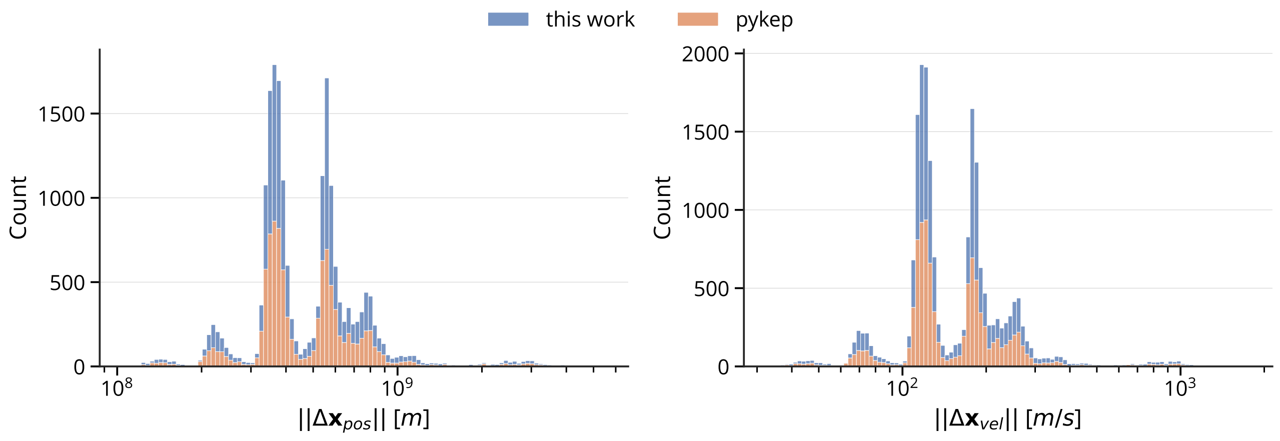


Figure 5.5: With-thrust state comparison of propagation of this work's implementation and pykep against reference trajectories, using Tudat's RKDP8(7) implementation with absolute and relative tolerances of 10^{-12}. The first ~10,000 trajectories of Öztürk and Izzo (2020) are used.

Furthermore, Figure 5.5 shows the error in position and velocity of both pykep and the implementation being verified here. Looking at both those figures, one can draw two conclusions: (1) The error, even when compared to pykep, is still high. Granted, on average, it is two orders of magnitude less – i.e., in the 10,000 km rather than in the 10^6 km – yet that is still a significant deviation. (2) When looking at the comparison of both pykep and this work to the reference trajectories, it can be seen that the error is distributed similarly.

Clearly, these three methods that allege to do the same, i.e., the data coming out of Izzo and Öztürk (2021), the implementation of the same equations in Cartesian in pykep, and the implementation of Izzo and Öztürk (2021) done here, do not. The reasons for this are unclear, and even correspondence with one co-author of Izzo and Öztürk (2021) did not manage to shed light on the source of the discrepancy.

Section 5.2.1 discussed one possible source: The throttle magnitude function. While this is, for a fully fuel-optimal trajectory, this is just a function stepping between zero and one, but the implementation used here only approximates that. It is approximated using a logarithmic barrier (see Paper/Equation 12), and when this work talks about 'fuel-optimal,' it follows Izzo and Öztürk (2021) in meaning a homotopy parameter of 10^{-6}. Mathematically, this does not make a difference in terms of computation, as the equation is defined for such a value of ε. When working in the realm of floating-point numbers, however, a construct like this, where very similar numbers are, effectively, subtracted from each other, can lead to something called 'catastrophic cancellation'. While not necessarily catastrophic in this case, it may be behind the differences in the throttle function values and fully depends on the specific order in which the mathematical operations are performed. An analysis of the potential of catastrophic cancellation specifically for the throttle function can be found in Appendix B.

Internal Consistency and Impact of Variable Step-Size Integration

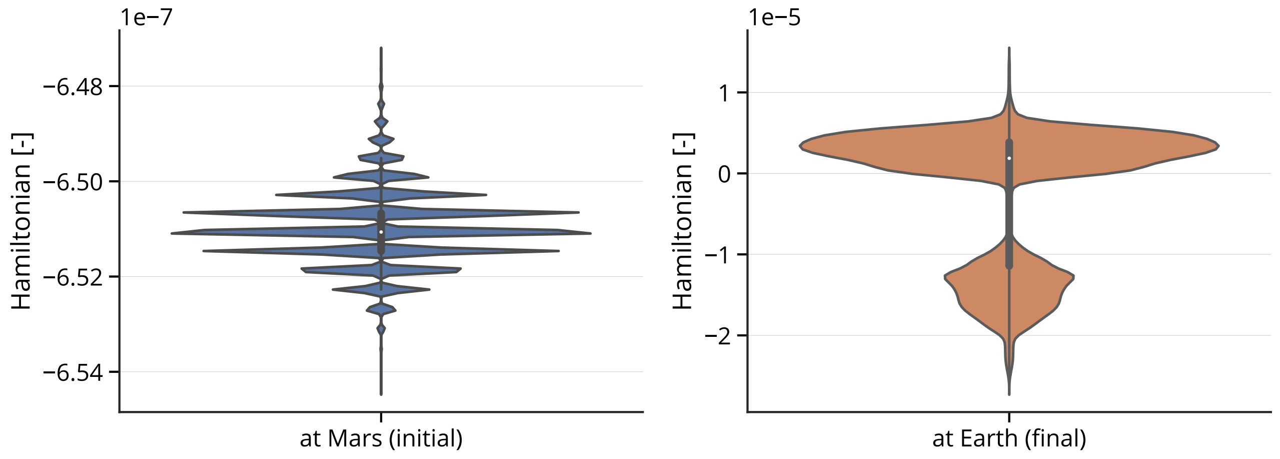


Figure 5.6: Distribution of Hamiltonian of the full training dataset used at the trajectories' starting and end-points.

The Pontryagin principle describes several conditions that an optimal control trajectory fulfills. Most of those are on the boundaries and can thus easily be enforced, yet one that applies to the entire trajectory is the constant Hamiltonian, thanks to the time-invariant dynamics. This constantness can, of course, be checked by making sure that the Hamiltonian at the start (which is ~0) and the end of propagation are identical (or, at least, very close). This check has been done in Figure 5.6. The order of magnitude is slightly increased (by 10^2) in the final propagation state over the initial one, yet the drift is rather small, and the final value is still within acceptable bounds. In fact, it is in the same order of magnitude as the nominal trajectory in Öztürk and Izzo (2020). This deviation is thus deemed satisfactory. Additionally, Figure 5.6 verifies Algorithm 1, as the costates are clearly scaled to approach $\mathcal{H} = 0$.

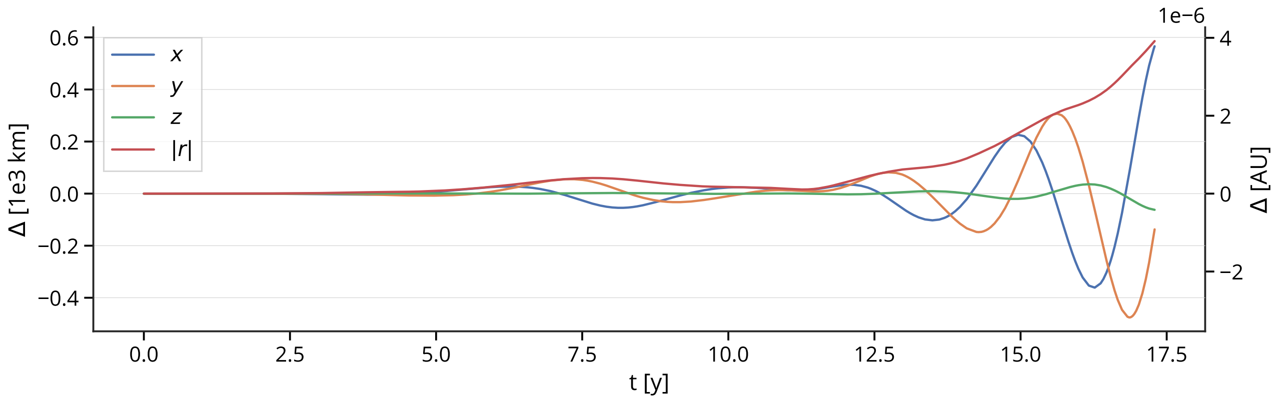


Figure 5.7: Euclidean distance between variable (absolute and relative error of 10^{-14}) and fixed (500,000 steps) step-size RKF8(9) propagation of the trajectory 'nominal' from Öztürk and Izzo (2020).

One additional verification step can be done: Based on what is described in literature, some integration routines can be quite sensitive to the rectangle nature of the throttle input. To see whether the integrator eventually used to generate the data (RKF8(9)) indeed has an issue with that. To gauge the impact of the variable step-size integration (without anything in the like of switching-detection), a comparison was made between a trajectory plotted with the variable RKF8(9) integrator and where that same integrator has its maximum and minimum step-size set such that it takes exactly 500,000 steps. This comparison can be seen in Figure 5.7. The final distance is 585.74 km, and the final velocity delta is 0.31873 m/s. This is about 5000 times more steps than the integrator would otherwise take, and the runtime scales close to linearly with the number of steps. Clearly, there is an impact, and higher precision (with similar runtime) can very likely be achieved by using a switching-detection technique. However, given the simplified nature of the dynamics considered, 500 km is deemed adequate. Chi et al. (2021) or Tang et al. (2018) are two examples making use of switching-detection.

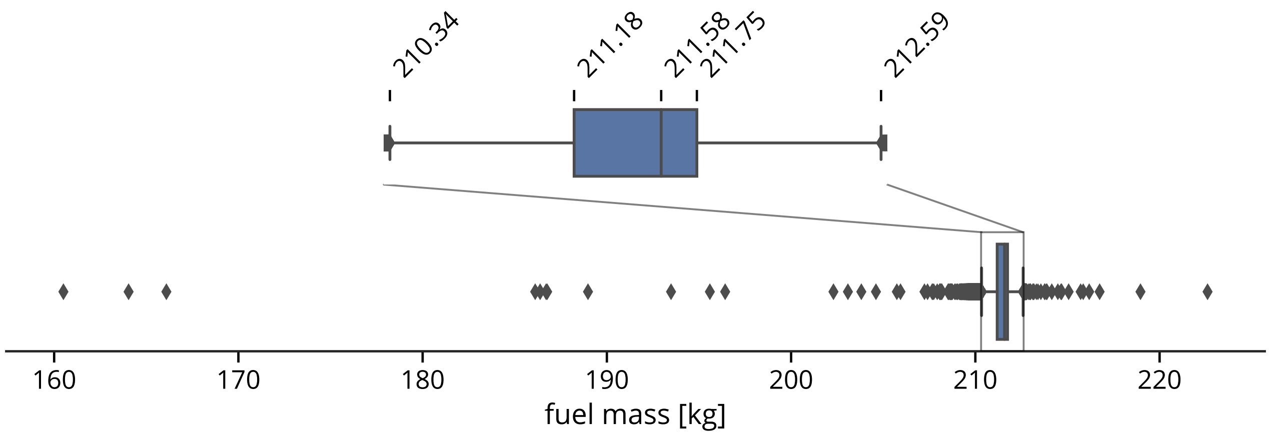


Figure 5.8: The distribution of the fuel mass found in the dataset used for training.

Finally, looking at the mass distribution of the found optimal trajectories and comparing the magnitude to literature is likely sensible. The fuel mass distribution in the training dataset has been visualized in Figure 5.8. In T. Li et al. (2021), a similar spacecraft (m_0 = 1533.35 kg, $T_{\max}$ = 0.35 N, and I_{sp} = 3000 s) is used and thus allows for comparison. The fuel mass found in the best-case here is 349.65 kg, which is of similar magnitude yet about 60 % more than the median seen in Figure 5.8. Taking into account that the thruster has a lower specific impulse than the spacecraft considered in this work (this work: 3800 s) and that the transfer time is fixed and not free, it is reasonable to assume that the fuel requirements are lower here compared to T. Li et al. (2021). The median required fuel mass of the trajectories found in this work of 211.58 kg thus seems reasonable.

5.3. Verification of Orbital State Representation Conversions

Tudat (TU Delft, 2022) does provide a significant number of orbital state representations, yet some were missing – or not available in extended precision. Others, like the Kustaanheimo-Stiefel elements, were redefined in this work and thus need to be verified. Please note that all conversions mentioned hereafter are only implemented and verified for prograde orbits, as the 'retrograde' element of the MEE set is assumed to be in prograde state ($+1$) throughout.

Unless mentioned, the state input data is converted from Keplerian states, which are drawn from a Sobol distribution and scaled as follows: $a \in [0.5, 3.5], e \in [0, 1], i \in [0, 0.8\pi]$, and $\{\omega, \Omega, \theta\} \in [0, 2\pi]^3$.

5.3.1. Cartesian to Spherical Conversion

Spherical elements are only used in the feature set, and hence only unidirectional conversion capabilities are required. The conversion is implemented to obtain extended precision, yet the same implementation is used for both – and given that Tudat provides the regular precision one, testing against that is deemed sufficient.

The 98.76 th percentile passes all tests, both in dimensional and nondimensional mode, pass with a relative accuracy of 5×10^{-14}. The exception is the flight-path angle, which required an accuracy threshold of 5.1×10^{-14}. The (reverse) trigonometric functions involved are the likely culprit here, yet the loss in accuracy is of negligible magnitude.

5.3.2. Modified Equinoctial Element to Poincaré Elements Conversion

The transformation from MEE to Poincaré is tested both in nondimensional and dimensional space for normal and extended precision. The conversion is tested back-and-forth and compared to the conversion from Keplerian to Poincaré elements, which has also been implemented as another point of verification. The default data is used as the base, as described in Section 5.3.

The 98.76 th percentile passes all tests with a relative accuracy of 5×10^{-14} and 5×10^{-16} for normal and extended precision, respectively. As for spherical elements, the threshold must be lowered for the longitude tests. Specifically, the mean longitude is only converted within a relative error of 5×10^{-15} in extended precision. The true longitude conversion in normal precision only passes the test for the 97.56 th percentile of the space considered.

5.3.3. Modified Equinoctial Elements to Delaunay Elements Conversion

The Delaunay element conversion uses the default base data, as explained in Section 5.3. It is tested against an implementation in Python closely following the mathematical description. Additionally, the conversion between true anomaly and mean anomaly used is the one provided by Tudat, and the Argument of Periapsis and the Right Ascension of Ascending Node is present in both the Keplerian set (which is what the test data is drawn in), and the set of Delaunay elements.

The 98.76 th percentile passes all tests that will not be explicitly described hereafter, with a relative accuracy of 5×10^{-14} and 5×10^{-16}. The normal precision conversions for the MEE f_e and f_g elements only pass for the 97.56 th percentile of test cases. This lower passing rate is probably due to the additional transcendental functions involved in the conversion. Furthermore, the true longitude only passes the tests for the 91.99 th percentile in *extended* precision. The 'wrapping' of the angle around 2π is presumably the culprit, as it is not happening in extended precision and thus negatively influences the results. Those inaccuracies mentioned seem to be artifacts of the testing methodology, however, and this conversion is thus nonetheless accepted as verified.

5.3.4. Cartesian to Kustaanheimo-Stiefel Conversion

The Kustaanheimo-Stiefel (KS) element testing is more involved for two reasons: (1) The representation used is different from what is described originally in Fukushima (2005), and (2) the original conversion from Cartesian includes a check for the sign of one of the Cartesian elements. The latter is, of course, a bad idea when dealing with floating-point numbers, as -10^{-16} and 10^{-16} could both be approximations of zero, yet the difference between the two means that a different element is 0 in the original KS representation. While that one is not used in this work, it is tested against to ensure that the reformulation is equivalent.

The following conversions are thus tested against each other:

- Cartesian $\xrightarrow{\text{'to-the-letter'}}$KS is compared to the regular Cartesian → KS mapping, where the column are conditionally rearranged (and multiplied by minus one) to match the original elements. 'to-the-letter' here means that the definition in Fukushima (2005) was followed to the letter.
- The 'to-the-letter' conversion is compared to a (performance-wise) improved reformulation.
- A back-and-forth, i.e., Cartesian → KS → Cartesian, conversion is done for the formulation used.
- A back-and-forth, i.e., KS → Cartesian → KS, conversion is done, again, for the actual formulation used.

The data for the KS elements is drawn from $\mathcal{N}(0, 5)$, which is the range of values encountered in interplanetary trajectories. Tests are, again, done both in regular spatial dimensions and nondimensionalized units. All tests pass with a relative accuracy of 5×10^{-14} and 5×10^{-16} for normal and extended precision, respectively, for the 99.73 th percentile.

5.4. Verification of the Costate Mappings

Mapping costates back and forth is relatively straightforward conceptually, yet the Jacobian matrix – and its inverse – can get quite complex. The implementation is thus carefully verified. This is done in two ways:

(1) The mapping between MEE and Cartesian is given in Taheri et al. (2021), and those figures can thus directly be used for verification. This is done in Section 5.4.1.

(2) Mapping back and forth, which does not ensure the mapping works either way, yet it does ensure that the mapping is internally consistent. Section 5.4.2 demonstrates that part – and also uses two different ways to handle the conversion for Keplerian elements, adding another level of confidence.

5.4.1. Example Figures from Reference Literature

The conversion between MEE and Cartesian costates can be verified using the examples in Taheri et al. (2021), as seen in table Table 5.1. The delta is a few orders of magnitude greater than the machine epsilon. The reason for that is presumably that a different implementation is used, and in floating-point arithmetic, things as simple as the order of addition can already change the result. In Taheri et al. (2021), the numbers were compared to the actual values, too, and the differences obtained here are more minor than the differences found in the source.

Table 5.1: Comparison of the mapping implementation and the reference values from Taheri et al. (2021, Tables 2 and 3).

Reference	Result	Difference
$1.78451232665985 \times 10^{2}$	$1.7845123266\underline{6110} \times 10^{2}$	1.242×10^{-10}
$-2.06416011743943 \times 10^{3}$	$-2.06416011743\underline{670} \times 10^{3}$	2.730×10^{-9}
$-3.49575926334692 \times 10^{2}$	$-3.49575926334\underline{550} \times 10^{2}$	1.420×10^{-10}
$1.35358251298554 \times 10^{3}$	$1.35358251298\underline{289} \times 10^{3}$	-2.650×10^{-9}
$-8.46061815385343 \times 10^{1}$	$-8.4606181538\underline{7407} \times 10^{1}$	-2.064×10^{-10}
$-6.38174235474277 \times 10^{2}$	$-6.38174235474\underline{305} \times 10^{2}$	-2.780×10^{-11}
$2.74460624199568 \times 10^{2}$	$2.74460624199568 \times 10^{2}$	0.000
$-2.11482459414456 \times 10^{3}$	$-2.11482459414456 \times 10^{3}$	0.000
$-4.80360304975842 \times 10^{2}$	$-4.8036030497584\underline{3} \times 10^{2}$	-1.000×10^{-13}
$1.37845088135386 \times 10^{3}$	$1.3784508813538\underline{7} \times 10^{3}$	2.000×10^{-12}
$-1.79013066954999 \times 10^{2}$	$-1.7901306695\underline{5000} \times 10^{2}$	-1.705×10^{-13}
$-1.05632346023203 \times 10^{3}$	$-1.0563234602320\underline{4} \times 10^{3}$	-5.000×10^{-12}

5.4.2. Back-And-Forth Mapping

Given the nature of the mappings, if they are correct, then they must work in both directions. This use case can be tested for the entire input range. In the following, the mapping to and from Cartesian and the mapping to and from Keplerian costates is demonstrated to be correctly implemented.

Mapping Between Modified Equinoctial Costates and Cartesian Costates

The input data is obtained in the following way:

- The MEE input costates are drawn from $\mathcal{N}(0, \sigma \in \{5, 0.1, 0.1, 0.1, 0.1, 0.1\})$.
- The Cartesian input costates are drawn from $\mathcal{N}(0, \sigma \in \{3, 3, 7.5, 6.5, 6.5, 1.5\})$
- The MEE-states are drawn from $\mathcal{N}(0, \sigma \in \{5, 0.1, 0.1, 0.1, 0.1, 0.1\})$, where the absolute value of the first value (i.e., the semi-latus rectum) is used.

Both normalized and non-normalized data is tested. The Sun's gravitational parameter is used. The resulting percentile and relative allowed errors obtained can be found in Table 5.2. While it is clear that all errors are due to floating-point arithmetic (all errors are smaller in extended precision), the dimensional mapping starting from MEE clearly has deplorable performance. The exact reason is unclear, yet it could be any floating-point effect, such as, e.g., catastrophic cancellation(Oliveira & Stewart, 2006). It could also simply be the fact that working with numbers that have considerably different orders of magnitude – MEE elements are often of order $\mathcal{O}(10^{-1})$ or smaller, while Cartesian elements in meters are of order $\mathcal{O}(10^{11})$ – is not the strength of finite precision floating-point numbers.

Mapping Between Modified Equinoctial Costates and Keplerian Costates

The testing here is done similarly to the one done for mapping to and from Cartesian costates.

The test data is drawn from a Sobol sequence and scaled in the following way:

Table 5.2: Verification and validation of obtained population accuracies.

Test-Case	Precision	Dimensionality	Percentile	Relative Accuracy
MEE → Cartesian → MEE	normal	nondimensional	99.73	5×10^{-14}
		dimensional	95.45	5×10^{-6}
	extended	nondimensional	99.73	5×10^{-16}
		dimensional	95.45	5×10^{-10}
Cartesian → MEE → Cartesian	normal	nondimensional	92.50	5×10^{-14}
		dimensional	92.50	5×10^{-14}
	extended	nondimensional	98.76	5×10^{-16}
		dimensional	98.76	5×10^{-16}

- The Keplerian elements are $a \in [0.5, 3.5], e \in [0, 1], i \in [0, 0.8\pi]$, and $\{\omega, \Omega, \theta\} \in [0, 2\pi]^3$.
- The MEE states are converted from the Keplerian ones.
- The MEE costates are drawn from the Sobol distribution using Muller (1959), i.e., are uniformly distributed on the surface of a unit 5-sphere.
- The (nondimensional) mass is $\in [0.6, 1.0]$.
- The mass costate is $\in [0, 10]$.

Two additional test parameters are varied, here:

1. The costates are not just taken as is, but are scaled for $\mathcal{H} = 0$, as they would be for an optimal control trajectory. Both this and the regular, 'unscaled' case are tested.
2. The mapping is implemented both using Keplerian and MEE states. Both are computed and compared.

The tests here are the same as for mapping to and from Cartesian, in addition to all combinations that result from varying the aforementioned two parameters. The results are not listed, as they can be summarized in one sentence: All tests pass with a percentile of 98.76 % and a relative accuracy of 6×10^{-14} and 5×10^{-16} for normal and extended precision, respectively.

5.5. Sobol Sequence

Some Sobol sequence was needed, yet was not available in an adequate version and thus had to be re-implemented. The implementation done in this work is based on an existing implementation[*] outlined in Joe and Kuo (2003, 2008). Given that the same implementation was used for the logic itself and was just modularized to generate the sequences incrementally, the comparison is exact – and matched exactly for all test cases. As the Sobol implementation used here does not only support drawing subsequent numbers but also supports drawing numbers at any arbitrary offset, this, too, was tested. Dimensions up to 20 were explicitly tested, but there is no reason to assume that the method would break down for higher dimensions. Note that this work never required Sobol numbers in more than seven dimensions, and the implementation thus remained well within what was tested.

[*]Available online: https://web.maths.unsw.edu.au/~fkuo/sobol/ [visited 24-01-2023]

6 Conclusions and Recommendations

The research question posed coming into the research was:

> How to select the feature set leading to the best possible performance of a feedforward neural network used for interplanetary Earth-Mars low-thrust fuel-optimal trajectories?

Before this question itself can be answered, the different sub-questions will be addressed in Section 6.1, first. From there, recommendations are formulated in Section 6.2.

6.1. Conclusions

- *What strategy can be used to select the elements describing an orbital state to use as features to maximize the neural network's performance?* This question cannot generally be answered from the work done here. It can be answered, specifically for low-thrust Earth-Mars transfers, however. In the case analyzed, using nondimensional Keplerian states turned out to be a solid starting point. When estimating costates, the final ones are better targets – at least in a free-time, free-longitude configuration. Additionally, when thinking about feature selection, being careful in one's target selection is just as, if not even more important. Here, Cartesian costates should be avoided, and the flavor of choice are Keplerian costates – though MEE costates are not very far behind. Doing so directly rather than estimating the spacecraft dry mass works best when estimating the fuel mass.
- *How to select the transformation to apply to features used for trajectory optimization?* It seems that the most important transformation is, as initially expected, to use nondimensional Keplerian states. As mentioned in the answer to the previous sub-question, while not a feature transformation, target transformations are important for good network performance, too. Finally, it has been shown that extending the feature space with additional components, such as the specific angular momentum or the orbital energy, has a greater chance of making the results worse rather than improving them.
- *Why does a transformation work for one feature, but not another?* Sine-cosine transformations were applied to variables representing angles and have been shown to make a marginal difference. Unfortunately, the 'why' could not be answered in this work as it would necessitate a different approach that allows for such insights.
- *Why does one state representation work better than another?* The problem being formulated in a free longitude manner would imply that representations having a single fast-moving parameter (such as the true longitude) would perform better. This has not necessarily been the case. The reason for that is, possibly, that while the position on the orbit is free in the problem formulation, the costates still have a strong dependence on that same position. This question, too, could thus not be conclusively answered.
- *How does one select features additional to the state representation?* For the case studied, additional parameters have not shown to be beneficial. In most cases, the networks that did not have any extra parameters performed better than the ones that did. One possible reason is that the network size was kept constant regardless of the number of elements in the feature vector. For that reason,

unless those extra parameters were able to add value, they would be a source of noise to the nodes that, in their absence, could have modelled relevant information.

- *How does one select the best combinations of features to synthesize new features in order to maximize the network's performance?* This question was not studied, in the end, as it is outside the scope of this work. Considering nine different state representations and additional (nuisance) factors already resulted in a considerable number of trained networks, such that combining different state representations – requiring exponentially more trained networks – was just not feasible.

Finally, the main question can be answered. In short: It depends. Well-scaled data is beneficial. This scaling can be achieved, on a greater scheme, using nondimensionalization – but also just normalizing the data can be beneficial to the performance. Feature engineering is very problem-specific, yet this work hopes to have shown the impact a number of factors have on the final performance of a network. When having to make a selection, as mentioned before, using Keplerian or MEE states and costates is likely a good starting point. Additionally, knowing that spending time adding additional parameters – unless one already knows that the function modeled contains those directly – is likely a waste of time. One should also pay attention to the target engineering part of the problem. Finally, if a certain feature selection shows promise for a small network and dataset, it is, at the very least, a good first guess to scale it up.

6.2. Recommendations

When working on a project like this, more questions than could possibly be answered tend to emerge. Here are a number of them that would nicely build on this work.

- *Expanding to more cases.* This work considered only Earth→Mars minimum fuel trajectories. Looking at Earth→Venus trajectories, trajectories to and between near-Earth asteroids, or more complex, gravity-assist trajectories is a logical next step. The data of those is likely different enough that conclusions drawn herein would not fully apply any longer, but only researching it can answer whether this is the case.
- *Investigate the effect of costate scaling.* The costate scaling is quite a relevant part of the data generation process. While the method used does converge – it likely is not the only solution resulting in a Hamiltonian of zero. Investigating different initial values and the performance resulting from that could potentially increase the convergence rate of the data generation method used.
- *Model more accurate orbit transfers.* The transfer orbits used go from Earth's orbit to Mars' orbit but do not include anything related to leaving a parking orbit or insertion into an orbit around Mars. This is a component that may be beneficial to model for more meaningful trajectories. Additionally, the free time and free true longitude are valid to find minimum mass trajectories, yet some iteration on the initial and final state might be valuable. That iteration would allow those trajectories to happen in a constellation that not only allows transfer between the two orbits but does so between the two celestial bodies.
- *Meta-algorithm predicting a feature set's performance.* One way to bring feature engineering to a next level would be to look at a meta-algorithm that predicts the performance of a given feature set. The input could be either the feature set itself or potentially just statistical descriptors of the different elements. The data needed to train such an algorithm is unattainably large, if based on machine learning. This problem may be resolvable once enough data from other published work is available.
- *In-depth statistical analysis of state representations.* The performance of a neural network depends on the features – that much is established. While this work tried numerous combinations, the 'why' is still unclear. Having a more profound look at whether there is a relationship between the behavior of individual state elements and the networks' performance is likely worth an investigation, too.
- *Estimating arrival position-on-orbit together with costates.* The optimal control problem considered kept the true longitude as free variable. The costates are still dependent on the arrival longitude (or the other way around), however. Therefore, including the targeted true longitude in the target may make sense and yield more meaningful results.
- *Estimating the initial mass together with arrival costates.* When propagating forward in time, one has the mass costate that starts strictly positive and ends at zero for the free-mass trajectory to be optimal. In backward time, this task is fulfilled by the mass itself, which must end at the initial wet mass of the

spacecraft. Estimating that parameter might thus be more valuable than estimating an always zero costate.

- *Investigate the effect of the dataset size more closely.* A smaller dataset turned out to be yield slightly better results in some cases. The analysis performed was not enough to unearth why this may be the case, yet more insight in that direction could be valuable. The results might have just been statistical artifacts, which could be straightforwardly confirmed or rejected through cross-validation.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv: 1603.04467 `[cs]`.

Ampatzis, C., & Izzo, D. (2009). Machine learning techniques for approximation of objective functions in trajectory optimisation. *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI09)*.

Arora, L., & Dutta, A. (2020). Reinforcement Learning for Sequential Low-Thrust Orbit Raising Problem. *AIAA Scitech 2020 Forum*. doi: 10.2514/6.2020-2186.

Bacon, R. H. (1959). Logarithmic Spiral: An Ideal Trajectory for the Interplanetary Vehicle with Engines of Low Sustained Thrust. *American Journal of Physics*, *27*(3), 164–165. doi: 10.1119/1.1934788.

Bertrand, R., & Epenoy, R. (2002). New smoothing techniques for solving bang–bang optimal control problems—numerical results and statistical interpretation. *Optimal Control Applications and Methods*, *23*(4), 171–197. doi: 10.1002/oca.709.

Biscani, F., & Izzo, D. (2020). A parallel global multiobjective framework for optimization: Pagmo. *Journal of Open Source Software*, *5*(53), 2338. doi: 10.21105/joss.02338.

Bonnans, F., Martinon, P., & Trélat, E. (2008). Singular Arcs in the Generalized Goddard's Problem. *Journal of Optimization Theory and Applications*, *139*(2), 439–461. doi: 10.1007/s10957-008-9387-1.

Brophy, J., Garner, C., Nakazono, B., Marcucci, M., Henry, M., & Noon, D. (2003). The Ion Propulsion System for Dawn. *39th AIAA/ ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. doi: 10.2514/6.2003-4542.

Carnelli, I., Dachwald, B., & Vasile, M. (2009). Evolutionary Neurocontrol: A Novel Method for Low-Thrust Gravity-Assist Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, *32*(2), 616–625. doi: 10.2514/1.32633.

Cassioli, A., Di Lorenzo, D., Locatelli, M., Schoen, F., & Sciandrone, M. (2012). Machine learning for global optimization. *Computational Optimization and Applications*, *51*(1), 279–303. doi: 10.1007/s10589-010-9330-x.

Chi, Z., Wang, Y., & Cheng, L. (2021). Indirect low-thrust trajectory optimization with gridded ion thruster model. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 09544100211043846. doi: 10.1177/09544100211043846.

Chobotov, V. A. (Ed.). (2002). *Orbital Mechanics* (3rd ed). American Institute of Aeronautics and Astronautics.

da Graça Marto, S., & Vasile, M. (2022). Many-objective robust trajectory optimisation under epistemic uncertainty and imprecision. *Acta Astronautica*, *191*, 99–124. doi: 10.1016/j.actaastro.2021.10.022.

Dachwald, B. (2004). Optimization of Interplanetary Solar Sailcraft Trajectories Using Evolutionary Neurocontrol. *Journal of Guidance, Control, and Dynamics*, *27*(1), 66–72. doi: 10.2514/1.9286.

Dachwald, B. (2005). Optimization of very-low-thrust trajectories using evolutionary neurocontrol [IAC-04-A.6.03]. *Acta Astronautica*, *57*(2), 175–185. doi: 10.1016/j.actaastro.2005.03.004.

Dachwald, B., & Seboldt, W. (2002). Optimization of Interplanetary Rendezvous Trajectories for Solar Sailcraft Using a Neurocontroller [AIAA 2002-4989]. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. doi: 10.2514/6.2002-4989.

Dahlquist, G., & Björck, Å. (2008). *Numerical methods in scientific computing*. Society for Industrial and Applied Mathematics.

Dalcin, L., & Fang, Y.-L. L. (2021). Mpi4py: Status Update After 12 Years of Development. *Computing in Science & Engineering*, *23*(4), 47–54. doi: 10.1109/MCSE.2021.3083216.

D'Ambrosio, A., Schiassi, E., Curti, F., & Furfaro, R. (2021). Pontryagin Neural Networks with Functional Interpolation for Optimal Intercept Problems. *Mathematics*, *9*(9), 996. doi: 10.3390/math9090996.

D'Ambrosio, A., Schiassi, E., Johnston, H., Curti, F., Mortari, D., & Furfaro, R. (2022). Time-energy optimal landing on planetary bodies via theory of functional connections. *Advances in Space Research*, *69*(12), 4198–4220. doi: 10.1016/j.asr.2022.04.009.

Delft High Performance Computing Centre. (2022). DelftBlue Supercomputer (Phase 1). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1

Dowell, M., & Jarratt, P. (1971). A modified regula falsi method for computing the root of an equation. *BIT Numerical Mathematics*, *11*(2), 168–174. doi: 10.1007/BF01934364.

Dunning, I., Mitchell, S., & O'Sullivan, M. (2011, September). PuLP: A Linear Programming Toolkit for Python – Optimization Online. Retrieved June 28, 2023, from https://optimization-online.org/2011/09/3178/

Epenoy, R. (2023). Fuel optimization of low-thrust trajectories under thrust and coast times constraints: A novel indirect approach. *Acta Astronautica*, *206*, 218–232. doi: 10.1016/j.actaastro.2023.02.030.

Facchinelli, M. (2018). *Aerobraking Navigation, Guidance and Control: A Comparison of State Variables Performance in Propagation* [Master's thesis, Delft University of Technology]. Retrieved May 7, 2023, from https://repository.tudelft.nl/islandora/object/uuid%3Aa4ed2919-e9a8-4ffb-8929-78a361a9ab07

Foroozandeh, Z., Shamsi, M., & Do Rosário De Pinho, M. (2018). A Hybrid Direct–Indirect Approach for Solving the Singular Optimal Control Problems of Finite and Infinite Order. *Iranian Journal of Science and Technology, Transactions A: Science*, *42*(3), 1545–1554. doi: 10.1007/s40995-017-0176-2.

Fukushima, T. (2005). Efficient Orbit Integration by Linear Transformation for Kustaanheimo-Stiefel Regularization. *The Astronomical Journal*, *129*(5), 2496. doi: 10.1086/429546.

Gabriel, E., Fagg, G. E., Bosilca, G., Angskun, T., Dongarra, J. J., Squyres, J. M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R. H., Daniel, D. J., Graham, R. L., & Timothy S. Woodall. (2004). Open MPI: Goals, concept, and design of a next generation MPI implementation. *Proceedings, 11th European PVM/MPI Users' Group Meeting*, 97–104.

Gaudet, B., Linares, R., & Furfaro, R. (2018). Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Powered Descent and Landing. arXiv: 1810.08719 [cs].

Gill, P. E., Murray, W., & Saunders, M. A. (2005). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, *47*(1), 99–131. doi: 10.1137/S0036144504446096.

Gómez Pérez, P., Liu, Y., & Cowan, K. (2020). Global optimization of low-thrust interplanetary trajectories using a machine learning surrogate [AAS 20-663]. *AAS/AIAA Astrodynamics Specialist Conference*, *175*, 5147–5166.

Grömping, U. (2014). R Package FrF2 for Creating and Analyzing Fractional Factorial 2-Level Designs. *Journal of Statistical Software*, *56*, 1–56. doi: 10.18637/jss.v056.i01.

Guo, T., Jiang, F., & Li, J. (2012). Homotopic approach and pseudospectral method applied jointly to low thrust trajectory optimization. *Acta Astronautica*, *71*, 38–50. doi: 10.1016/j.actaastro.2011.08.008.

Haenlein, M., & Kaplan, A. (2019). A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. *California Management Review*, *61*(4), 5–14. doi: 10.1177/0008125619864925.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proceedings of the IEEE International Conference on Computer Vision*, 1026–1034. doi: 10.1109/ICCV.2015.123.

Hintz, G. R. (2008). Survey of Orbit Element Sets. *Journal of Guidance, Control, and Dynamics*, *31*(3), 785–790. doi: 10.2514/1.32237.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. doi: 10.1016/0893-6080(89)90020-8.

Izzo, D., Binns, W., Mereta, A., Sprague, C. I., den Abbeele, B. V., Andre, C., Nowak, K., Guy, N., Murguía, A. I. B., Chapoton, F., v Looz, M., Heddes, M., Babenia, A., Fournier, B., Simon, J., Willitts, J., Polnik, M., Narayanaswamy, S., Badger, T. G., & Yarndley, J. (2020, October). Esa/pykep: Optimize. doi: 10.5281/zenodo.4091753.

Izzo, D., Märtens, M., & Pan, B. (2019). A survey on artificial intelligence trends in spacecraft guidance dynamics and control. *Astrodynamics*, *3*(4), 287–299. doi: 10.1007/s42064-018-0053-6.

Izzo, D., & Öztürk, E. (2021). Real-Time Guidance for Low-Thrust Transfers Using Deep Neural Networks. *Journal of Guidance, Control, and Dynamics*, *44*(2), 315–327. doi: 10.2514/1.G005254.

Izzo, D., Sprague, C., & Tailor, D. (2018). Machine learning and evolutionary techniques in interplanetary trajectory design. arXiv: 1802.00180 [cs].

Jakob, W., Rhinelander, J., & Moldovan, D. (2017). Pybind11 – Seamless operability between C++11 and Python. Retrieved June 8, 2023, from https://github.com/pybind/pybind11

Jawaharlal Ayyanathan, P., & Taheri, E. (2022). Mapped adjoint control transformation method for low-thrust trajectory design. *Acta Astronautica*, *193*, 418–431. doi: 10.1016/j.actaastro.2021.12.019.

Jiang, F., Baoyin, H., & Li, J. (2012). Practical Techniques for Low-Thrust Trajectory Optimization with Homotopic Approach. *Journal of Guidance, Control, and Dynamics*, *35*(1), 245–258. doi: 10.2514/1.52476.

Joe, S., & Kuo, F. Y. (2003). Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, *29*(1), 49–57. doi: 10.1145/641876.641879.

Joe, S., & Kuo, F. Y. (2008). Constructing Sobol Sequences with Better Two-Dimensional Projections. *SIAM Journal on Scientific Computing*, *30*(5), 2635–2654. doi: 10.1137/070709359.

Johnston, H., Schiassi, E., Furfaro, R., & Mortari, D. (2020). Fuel-Efficient Powered Descent Guidance on Large Planetary Bodies via Theory of Functional Connections. *67*(4), 1521–1552. doi: 10.1007/s40295-020-00228-x. arXiv: 2001.03572 [cs, eess, math].

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. arXiv: 1412.6980 [cs].

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc.

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., & Bruhin, F. (2004). Pytest 7.2.0. https://github.com/pytest-dev/pytest

Kuninaka, H., Nishiyama, K., Funaki, I., Yamada, T., Shimizu, Y., & Kawaguchi, J. (2006, July). Powered Flight of HAYABUSA in Deep Space. In *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2006-4318.

Kustaanheimo, P., & Stiefel, E. (1965). Perturbation Theory of Kepler Motion Based on Spinor Regularization. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, *1965*(218), 204–219. doi: 10.1515/crll.1965.218.204.

LaFarge, N. B., Miller, D., Howell, K. C., & Linares, R. (2021). Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment. *Acta Astronautica*, *186*, 1–23. doi: 10.1016/j.actaastro.2021.05.014.

Lawden, D. F. (1963). *Optimal Trajectories for Space Navigation*. Butterworths.

Li, H., Baoyin, H., & Topputo, F. (2019). Neural Networks in Time-Optimal Low-Thrust Interplanetary Transfers. *IEEE Access*, *7*, 156413–156419. doi: 10.1109/ACCESS.2019.2946657.

Li, H., Chen, S., Izzo, D., & Baoyin, H. (2020). Deep networks as approximators of optimal low-thrust and multi-impulse cost in multitarget missions. *Acta Astronautica*, *166*, 469–481. doi: 10.1016/j.actaastro.2019.09.023.

Li, H., Topputo, F., & Baoyin, H. (2019). Autonomous Time-Optimal Many-Revolution Orbit Raising for Electric Propulsion GEO Satellites via Neural Networks. arXiv: 1909.08768 [eess, math].

Li, T., Wang, Z., & Zhang, Y. (2021). A homotopy approach connecting time-optimal with fuel-optimal trajectories. *Astrophysics and Space Science*, *366*(1), 11. doi: 10.1007/s10509-020-03890-7.

Lyon, R. H. (2004, June). *Geosynchronous Orbit Determination Using Space Surveillance Network Observations And Improved Radiative Force Modeling* [Master's thesis, Massachusetts Institute of Technology]. Retrieved May 7, 2023, from https://core.ac.uk/display/80014985

Makhorin, A. (2020). GNU linear programming kit, version 5.0. https://www.gnu.org/software/glpk/

Markley, F. L. (1995). Kepler Equation solver. *Celestial Mechanics and Dynamical Astronomy*, *63*(1), 101–111. doi: 10.1007/BF00691917.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133. doi: 10.1007/BF02478259.

Mereta, A., Izzo, D., & Wittig, A. (2017). Machine Learning of Optimal Low-Thrust Transfers Between Near-Earth Objects. In F. J. Martínez de Pisón, R. Urraca, H. Quintián, & E. Corchado (Eds.), *Hybrid Artificial Intelligent Systems* (pp. 543–553). Springer International Publishing. doi: 10.1007/978-3-319-59650-1_46.

Miller, D., Englander, J. A., & Linares, R. (2019). Interplanetary Low-Thrust Design Using Proximal Policy Optimization [AAS 19-779]. *2019 AAS/AIAA Astrodynamics Specialist Conference*, (GSFC-E-DAA-TN71225). Retrieved December 17, 2021, from https://ntrs.nasa.gov/citations/20190029151

Miller, D., & Linares, R. (2019). Low-Thrust Optimal Control via Reinforcement Learning [AAS 19-560]. *29th AAS/AIAA Space Flight Mechanics Meeting*.

Montgomery, D. C. (2017). *Design and analysis of experiments* (Ninth edition). John Wiley & Sons, Inc.

Morelli, A. C., Hofmann, C., & Topputo, F. (2022). Robust Low-Thrust Trajectory Optimization Using Convex Programming and a Homotopic Approach. *IEEE Transactions on Aerospace and Electronic Systems*, *58*(3), 2103–2116. doi: 10.1109/TAES.2021.3128869.

Muller, M. E. (1959). A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, *2*(4), 19–20. doi: 10.1145/377939.377946.

Murty, K. G. (1983). *Linear programming*. Wiley.

Novara, M. (2002). The BepiColombo ESA cornerstone mission to Mercury. *Acta Astronautica*, *51*(1), 387–395. doi: 10.1016/S0094-5765(02)00065-6.

Oestreich, C. E., Linares, R., & Gondhalekar, R. (2021). Autonomous Six-Degree-of-Freedom Spacecraft Docking with Rotating Targets via Reinforcement Learning. *Journal of Aerospace Information Systems*, *18*(7), 417–428. doi: 10.2514/1.I010914.

Oh, D. Y., Snyder, J. S., Goebel, D. M., Hofer, R. R., & Randolph, T. M. (2014). Solar Electric Propulsion for Discovery-Class Missions. *Journal of Spacecraft and Rockets*, *51*(6), 1822–1835. doi: 10.2514/1.A32889.

Oliveira, S., & Stewart, D. (2006). *Writing scientific software: A guide for good style*. Cambridge University Press.

Ozaki, N., Yanagida, K., Chikazawa, T., Pushparaj, N., Takeishi, N., & Hyodo, R. (2022). Asteroid Flyby Cycler Trajectory Design Using Deep Neural Networks. arXiv: 2111.11858 [astro-ph].

Öztürk, E., & Izzo, D. (2020, January). Earth - venus low-thrust optimal transfers / database A. doi: 10.5281/zenodo.3613772.

Pan, B., Pan, X., & Ma, Y. (2019). A quadratic homotopy method for fuel-optimal low-thrust trajectory design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, *233*(5), 1741–1757. doi: 10.1177/0954410018761965.

Pérez-Cutiño, M., Rodríguez, F., Pascual, L., & Díaz-Báñez, J. (2021). Neural networks algorithms for ornithopter trajectory optimization. *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1665–1670. doi: 10.1109/ICUAS51884.2021.9476838.

Peterson, J. T. A., Junkins, J. L., & Majji, M. (2022, July). On equinoctial elements and Rodrigues parameters [AAS 22-832]. Retrieved September 4, 2022, from https://www.researchgate.net/publication/362243796

Poincaré, H. (1892). *Les méthodes nouvelles de la mécanique céleste* (Vol. 1). Gauthier-Villars et fils. Retrieved May 8, 2023, from http://archive.org/details/lesmthodesnouv001poin

Pontryagin, L. S. (1986). *The mathematical theory of optimal processes* (English ed). Gordon and Breach Science Publishers.

Pourtakdoust, S., Pazooki, F., & Fakhri Noushabadi, M. (2009). A neuro-optimal approach for thrust-insensitive trajectory planning. *Aircraft Engineering and Aerospace Technology*, *81*(3), 212–220. doi: 10.1108/00022660910954718.

Rosa Sentinella, M., & Casalino, L. (2006). Genetic Algorithm and Indirect Method Coupling for Low-Thrust Trajectory Optimization. *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. doi: 10.2514/6.2006-4468.

Sánchez-Sánchez, C., & Izzo, D. (2018). Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems. *Journal of Guidance, Control, and Dynamics*, *41*(5), 1122–1135. doi: 10.2514/1.G002357.

Schiassi, E., D'Ambrosio, A., Drozd, K., Curti, F., & Furfaro, R. (2022). Physics-Informed Neural Networks for Optimal Planar Orbit Transfers. *Journal of Spacecraft and Rockets*, *59*(3), 834–849. doi: 10.2514/1.A35138.

Schiassi, E., Furfaro, R., Leake, C., De Florio, M., Johnston, H., & Mortari, D. (2021). Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations. *Neurocomputing*, *457*, 334–356. doi: 10.1016/j.neucom.2021.06.015.

Shirobokov, M., Trofimov, S., & Ovchinnikov, M. (2021). Survey of machine learning techniques in spacecraft control design. *Acta Astronautica*, *186*, 87–97. doi: 10.1016/j.actaastro.2021.05.018.

Smart, W. M. (1953). *Celestial mechanics*. Longmans, Green.

Solano-López, P., Vilar, A. B., Gutiérrez-Ramon, R., & Cereijo, H. U. (2022). Assessment of Machine Learning Techniques for Time-Optimal Landing Trajectories. *Proceedings of the 9th European Conference for Aerospace Sciences*, 15 pages. doi: 10.13009/EUCASS2022-4761.

Stubbig, L. (2019). *Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories* [Master's thesis, Delft University of Technology].

Stubbig, L., & Cowan, K. (2020). Improving the evolutionary optimization of interplanetary low-thrust trajectories using a neural network surrogate model [AAS 20-658]. *AAS/AIAA Astrodynamics Specialist Conference*, *175*, 5127–5146.

Stuhlinger, E. (1955). Possibilities of Electrical Space Ship Propulsion. In F. Hecht (Ed.), *Bericht über den V. Internationalen Astronautischen Kongreß: Innsbruck, 5. bis 7. August 1954* (pp. 100–119). Springer. doi: 10.1007/978-3-662-38334-6_7.

Stuhlinger, E. (1957). The Flight Path of an Electrically Propelled Space Ship. *Journal of Jet Propulsion*, *27*(4), 410–414. doi: 10.2514/8.12798.

Taheri, E., Arya, V., & Junkins, J. L. (2021). Costate mapping for indirect trajectory optimization. *Astrodynamics*, *5*(4), 359–371. doi: 10.1007/s42064-021-0114-0.

Tang, G., Jiang, F., & Li, J. (2018). Fuel-Optimal Low-Thrust Trajectory Optimization Using Indirect Method and Successive Convex Programming. *IEEE Transactions on Aerospace and Electronic Systems*, *54*(4), 2053–2066. doi: 10.1109/TAES.2018.2803558.

Tsien, H. S. (1953). Take-Off from Satellite Orbit. *Journal of the American Rocket Society*, *23*(4), 233–236. doi: 10.2514/8.4599.

TU Delft. (2022). Tudat – TU Delft Astrodynamics Toolbox. Retrieved May 12, 2022, from https://github.com/tudat-team/tudat

Vallado, D. A., & McClain, W. D. (2007). *Fundamentals of Astrodynamics and Applications* (3. ed., 1. printing). Microcosm Press [u.a.]

van Smeden, M., de Groot, J. A. H., Moons, K. G. M., Collins, G. S., Altman, D. G., Eijkemans, M. J. C., & Reitsma, J. B. (2016). No rationale for 1 variable per 10 events criterion for binary logistic regression analysis. *BMC Medical Research Methodology*, *16*(1), 163. doi: 10.1186/s12874-016-0267-3.

Viavattene, G., & Ceriotti, M. (2022). Artificial Neural Networks for Multiple NEA Rendezvous Missions with Continuous Thrust. *Journal of Spacecraft and Rockets*, *59*(2), 574–586. doi: 10.2514/1.A34799.

Vittaldev, V., Mooij, E., & Naeije, M. C. (2012). Unified State Model theory and application in Astrodynamics. *Celestial Mechanics and Dynamical Astronomy*, *112*(3), 253–282. doi: 10.1007/s10569-011-9396-5.

Vittinghoff, E., & McCulloch, C. E. (2007). Relaxing the Rule of Ten Events per Variable in Logistic and Cox Regression. *American Journal of Epidemiology*, *165*(6), 710–718. doi: 10.1093/aje/kwk052.

Wakker, K. (2015). *Fundamentals of Astrodynamics*. Institutional Repository, Delft University of Technology.

Walker, M. J. H., Ireland, B., & Owens, J. (1985). A set of modified equinoctial orbit elements. *Celestial mechanics*, *36*(4), 409–419. doi: 10.1007/BF01227493.

Wang, Z., & Grant, M. J. (2018). Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach. *IEEE Transactions on Aerospace and Electronic Systems*, *54*(5), 2274–2290. doi: 10.1109/TAES.2018.2812558.

Xie, R., & Dempster, A. G. (2021). An on-line deep learning framework for low-thrust trajectory optimisation. *Aerospace Science and Technology*, *118*, 107002. doi: 10.1016/j.ast.2021.107002.

Yang, H., Li, S., & Bai, X. (2019). Fast Homotopy Method for Asteroid Landing Trajectory Optimization Using Approximate Initial Costates. *Journal of Guidance, Control, and Dynamics*, *42*(3), 585–597. doi: 10.2514/1.G003414.

Yin, S., Li, J., & Cheng, L. (2020). Low-thrust spacecraft trajectory optimization via a DNN-based method. *Advances in Space Research*, *66*(7), 1635–1646. doi: 10.1016/j.asr.2020.05.046.

Zhu, Z., Gan, Q., Yang, X., & Gao, Y. (2017). Solving fuel-optimal low-thrust orbital transfers with bang-bang control using a novel continuation technique. *Acta Astronautica*, *137*, 98–113. doi: 10.1016/j.actaastro.2017.03.032.

A

Mathematical Definitions and Derivations

This appendix provides mathematical reference material, ranging from derivations to simple definitions. First, Section A.1 provides the derivatives of the costates, which are part of the MEE dynamics used within this work. After that, the derivative in orbit descriptor difference is presented in Section A.2. This is followed by a section outlining a few derivations for conversions between state representations used, some of which are new derivations not available in literature (Section A.3). Finally, Section A.4 provides the matrices needed to map costates from MEE to Keplerian elements and back.

A.1. Full Costate Dynamic Equations

This appendix contains the full derivative matrices for the orbital costate dynamics outlined in Paper/Applying Pontryagin's Minimum Principle (p. 5), specifically Paper/Equation 13. All equations are taken from Izzo and Öztürk (2021).

The helper variables w and s^2 are defined as

$$w = 1 + f_e \cos L + g_e \sin L \quad \text{and} \quad s^2 = 1 + h_e^2 + k_e^2. \tag{A.1}$$

A.1.1. The derivative of λ_p

λ_p is given by

$$\dot{\lambda}_p = -\frac{Tu}{m}\boldsymbol{\lambda}^T \frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial p}\hat{\boldsymbol{i}}_\tau + \frac{3}{2}w^2\lambda_L\sqrt{\frac{\mu}{p^5}} \tag{A.2}$$

where

$$\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial p} = \frac{1}{2w\sqrt{\mu p}}\begin{bmatrix} 0 & 6p & 0 \\ w\sin L & f_e + (w+1)\cos L & -g_e(h_e \sin L - k_e \cos L) \\ -w\cos L & g_e + (w+1)\sin L & f_e(h_e\sin L - k_e\cos L) \\ 0 & 0 & \frac{s^2}{2}\cos L \\ 0 & 0 & \frac{s^2}{2}\sin L \\ 0 & 0 & h_e\sin L - k_e \cos L \end{bmatrix}. \tag{A.3}$$

A.1.2. The derivative of λ_f

λ_f is given by

$$\dot{\lambda}_f = -\frac{Tu}{m}\boldsymbol{\lambda}^T \frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial f_e}\hat{\boldsymbol{i}}_\tau - 2\lambda_L w\sqrt{\frac{\mu}{p^3}}\cos L \tag{A.4}$$

where

$$\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial f_e} = \frac{1}{w^2}\sqrt{\frac{p}{\mu}}\begin{bmatrix} 0 & -2p\cos L & 0 \\ 0 & w-(f_e+\cos L)\cos L & g_e(h_e\sin L - k_e\cos L)\cos L \\ 0 & (-g_e-\sin L)\cos L & (w-f_e\cos L)(h_e\sin L-k_e\cos L) \\ 0 & 0 & -\frac{s^2}{2}\cos^2 L \\ 0 & 0 & -\frac{s^2}{2}\sin L\cos L \\ 0 & 0 & (-h_e\sin L+k_e\cos L)\cos L \end{bmatrix}. \tag{A.5}$$

A.1.3. The derivative of λ_g

λ_g is given by

$$\dot{\lambda}_g = -\frac{Tu}{m}\boldsymbol{\lambda}^T\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial g_e}\hat{\boldsymbol{i}}_\tau - 2\lambda_L w\sqrt{\frac{\mu}{p^3}}\sin L \tag{A.6}$$

where

$$\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial g_e} = \frac{1}{w^2}\sqrt{\frac{p}{\mu}}\begin{bmatrix} 0 & -2p\sin L & 0 \\ 0 & (-f_e-\cos L)\sin L & (g_e\sin L-w)(h_e\sin L-k_e\cos L) \\ 0 & w-(g_e+\sin L)\sin L & -f_e(h_e\sin L-k_e\cos L)\sin L \\ 0 & 0 & -\frac{s^2}{2}\sin L\cos L \\ 0 & 0 & -\frac{s^2}{2}\sin^2 L \\ 0 & 0 & (-h_e\sin L+k_e\cos L)\sin L \end{bmatrix}. \tag{A.7}$$

A.1.4. The derivative of λ_h

λ_h is given by

$$\dot{\lambda}_h = -\frac{Tu}{m}\boldsymbol{\lambda}^{\mathsf{T}}\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial h_e}\hat{\boldsymbol{i}}_\tau \tag{A.8}$$

where

$$\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial h_e} = \frac{1}{w}\sqrt{\frac{p}{\mu}}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -g_e\sin L \\ 0 & 0 & f_e\sin L \\ 0 & 0 & h_e\cos L \\ 0 & 0 & h_e\sin L \\ 0 & 0 & \sin L \end{bmatrix}. \tag{A.9}$$

A.1.5. The derivative of λ_k

λ_k is given by

$$\dot{\lambda}_k = -\frac{Tu}{m}\boldsymbol{\lambda}^{\mathsf{T}}\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial k_e}\hat{\boldsymbol{i}}_\tau \tag{A.10}$$

where

$$\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial k_e} = \frac{1}{w}\sqrt{\frac{p}{\mu}}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & g_e\cos L \\ 0 & 0 & -f_e\cos L \\ 0 & 0 & k_e\cos L \\ 0 & 0 & k_e\sin L \\ 0 & 0 & -\cos L \end{bmatrix}. \tag{A.11}$$

A.1.6. The derivative of λ_L

λ_L is given by

$$\dot{\lambda}_L = -\frac{Tu}{m}\boldsymbol{\lambda}^T\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial L}\hat{\boldsymbol{i}}_\tau - 2w\sqrt{\frac{\mu}{p^3}}\lambda_L w_L \tag{A.12}$$

where

$$\frac{\partial \boldsymbol{B}(\boldsymbol{x})}{\partial L} = \frac{1}{w^2}\sqrt{\frac{p}{\mu}} \begin{bmatrix} 0 & -2p(g_e \cos L - f_e \sin L) & 0 \\ w^2 \cos L & -w(w+1)\sin L - w_L(f_e + \cos L) & [(wh_e + w_L k_e)\cos L + (wk_e - w_L h_e)\sin L]\, g_e \\ w^2 \sin L & w(w+1)\cos L - w_L(g_e + \sin L) & [(wh_e + w_L k_e)\cos L + (wk_e - w_L h_e)\sin L]\, f_e \\ 0 & 0 & -\frac{s^2}{2}(w \sin L + w_L \cos L) \\ 0 & 0 & \frac{s^2}{2}(w \cos L - w_L \sin L) \\ 0 & 0 & (wh_e + w_L k_e)\cos L + (wk_e - w_L h_e)\sin L \end{bmatrix}, \tag{A.13}$$

and

$$w_L = \frac{\partial w}{\partial L} = g_e \cos L - f_e \sin L. \tag{A.14}$$

A.1.7. The derivative of λ_m

λ_m is given by

$$\dot{\lambda}_m = -\frac{Tu}{m^2}\|\boldsymbol{\lambda}^\mathsf{T}\boldsymbol{B}(\boldsymbol{x})\|. \tag{A.15}$$

A.2. Derivative of the Difference in Orbits

Within this document mostly labelled as $\Delta\mathcal{M}$, the derivative of the difference in metric introduced in Paper/Equation 17 is,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}(\Delta\mathcal{M}) = {} & 2 \cdot \boldsymbol{c}_{\vec{e}}^2 \bullet \left(\left(p \cdot \boldsymbol{x}_{\vec{e}} - p_t \cdot \boldsymbol{x}_{\vec{e},t}\right) \odot \left(\dot{p} \cdot \boldsymbol{x}_{\vec{e}} + p \cdot \dot{\boldsymbol{x}}_{\vec{e}}\right)\right) \\ & + 2 \cdot \left(p^{-1} + p \cdot \boldsymbol{c}_{\vec{e}}^2 \bullet \boldsymbol{x}_{\vec{e}}^2 - p_t^{-1} - p_t \cdot \boldsymbol{c}_{\vec{e}}^2 \bullet \boldsymbol{x}_{\vec{e},t}^2\right) \cdot \left(\dot{p} \cdot \boldsymbol{c}_{\vec{e}}^2 \bullet \boldsymbol{x}_{\vec{e}}^2 - \dot{p}p^{-2} + 2p \cdot \boldsymbol{c}_{\vec{e}}^2 \bullet \left(\boldsymbol{x}_{\vec{e}} \odot \dot{\boldsymbol{x}}_{\vec{e}}\right)\right) \end{aligned} \tag{A.16}$$

where $\boldsymbol{x}_{\vec{e}} = \left[f_e,\ \ g_e,\ \ h_e,\ \ k_e\right]^\mathsf{T}$, $\boldsymbol{c}_{\vec{e}}^2 = \left[c_f^2,\ \ c_g^2,\ \ c_h^2,\ \ c_k^2\right]^\mathsf{T}$, and the suffix $\square_t$ stands for 'target' and identifies the elements of the target state. $\bullet$ is the dot-product and $\odot$ is the Hadamard (element-wise) product.

A.3. State Representation Conversion Derivations

In this part of the appendix, several state representations and their conversion to and from a known state representation are described and, if necessary, derived. This starts with the canonical elements, i.e., the Delaunay elements and Poincaré elements, in Section A.3.1 and A.3.2, respectively. This is followed by a conversion between MEE and Keplerian elements in Section A.3.3, which is needed for the next section in this appendix, Section A.4. Finally, the Kustaanheimo-Stiefel elements are described, first in one of the definitions available in literature and subsequently the definition used within this work. This happens in Section A.3.4.

A.3.1. Delaunay conversions

The Delaunay elements are the following:

- L_d Delaunay canonical variable,
- h specific angular momentum,
- H_d Delaunay canonical variable.
- ω argument of periapsis,
- Ω right ascension of ascending node,
- M mean anomaly.

The derivation coming here-after is structured in the following way. First, the relation between the MEE orientation parameters k_e and h_e will be exploited to obtain the cosine of the inclination (Equation A.17). From there, in Equation A.18 a relation for the mean anomaly will be derived, coming from MEE elements, again. Following that, everything is put together to get a full conversion from Modified Equinoctial Elements to Delaunay elements (see Equation A.19). Finally, the inverse relationship, the conversion from Delaunay elements to MEE, is derived – found in Equation A.20.

The Cosine of Inclination

According to Tudat (for non-retrograde orbits), $i = 2 \cdot \arctan\sqrt{h_e^2 + k_e^2}$. The definition of Delaunay elements does not contain i directly, yet it does contain $\cos i$.

Introducing the temporary $\varkappa$ for simplicity in the later computations,

$$\varkappa = \arctan\sqrt{h_e^2 + k_e^2} \quad \Rightarrow \quad \tan\varkappa = \sqrt{h_e^2 + k_e^2}.$$

Making use of trigonometric identities,

$$\tan^2\varkappa = h_e^2 + k_e^2 = \sec^2\varkappa - 1 = \frac{1}{\cos^2} - 1 \quad \Rightarrow \quad \cos^2\varkappa = \frac{1}{h_e^2 + k_e^2 + 1},$$
$$\cos 2\varkappa = 2\cos^2\varkappa - 1 = 2\left(\frac{1}{h_e^2 + k_e^2 + 1}\right) - 1 \quad \Rightarrow \quad \cos 2\varkappa = \frac{-h_e^2 - k_e^2 + 1}{h_e^2 + k_e^2 + 1}.$$

Finally, a direct relation that is only a function of the two MEE parameters h_e and k_e is found,

$$\Rightarrow \cos i = \cos\left(2 \cdot \arctan\sqrt{h_e^2 + k_e^2}\right) = \frac{-h_e^2 - k_e^2 + 1}{h_e^2 + k_e^2 + 1} = \frac{2}{h_e^2 + k_e^2 + 1} - 1. \tag{A.17}$$

The Mean Anomaly

Following Vallado and McClain (2007, eq. 2-9),

$$\sin E = \frac{\sin\theta\sqrt{1-e^2}}{1 + e\cos\theta}, \qquad \cos E = \frac{e + \cos\theta}{1 + e\cos\theta},$$

and combining the two using the atan2 function,

$$\begin{aligned} E &= \operatorname{atan2}\left(\frac{\sin\theta \cdot \sqrt{1-e^2}}{1 + e\cos\theta}, \frac{e + \cos\theta}{1 + e\cos\theta}\right) \\ &= \operatorname{atan2}\left(\sin\theta \cdot \sqrt{1-e^2}, e + \cos\theta\right). \end{aligned} \tag{A.18}$$

The Conversion From MEE to Delaunay

Combining the base definition, as given by Hintz (2008), TU Delft (2022), and Vallado and McClain (2007), with Equation A.17 and A.18 (undefined for $e >= 1$),

$$\begin{cases} L_d = \sqrt{\mu a} = \dfrac{h}{\sqrt{1-e^2}}, \\ h = \sqrt{\mu p}, \\ H_d = \sqrt{\mu p} \cdot \cos i = h \cdot \left(\dfrac{2}{h_e^2 + k_e^2 + 1} - 1\right), \\ \omega = \begin{cases} \operatorname{atan2}(g_e, f_e) - \Omega & \text{for } e > 0 \\ 0 & \text{for } e \approx 0 \end{cases}, \\ \Omega = \operatorname{atan2}(k_e, h_e), \\ M = E - e\sin E = \operatorname{atan2}\left(\sqrt{1-e^2}\sin\theta, e + \cos\theta\right) - e\dfrac{\sqrt{1-e^2}\sin\theta}{1 + e\cos\theta}, \end{cases} \tag{A.19}$$

where θ, the true anomaly, is found by using $L = \omega + \Omega + \theta$. Also note that the eccentricity is found through the relation $e^2 = f_e^2 + g_e^2$.

The Conversion From Delaunay to MEE

To go from Delaunay elements to MEE, one essentially only needs to reverse the previous equations

$$\begin{cases} p = \dfrac{h^2}{\mu}, & h_e = \sqrt{\dfrac{h - H_d}{h + H_d}}\cos\Omega, \\ f_e = \sqrt{1 - \left(\dfrac{h}{L_d}\right)^2}\cos(\omega + \Omega), & k_e = \sqrt{\dfrac{h - H_d}{h + H_d}}\sin\Omega, \\ g_e = \sqrt{1 - \left(\dfrac{h}{L_d}\right)^2}\sin(\omega + \Omega), & L = \omega + \Omega + \theta, \end{cases} \tag{A.20}$$

while the true anomaly θ is found via the eccentric anomaly E (Vallado & McClain, 2007),

$$\tan\frac{\theta}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2}.$$

The eccentric anomaly itself is found from the mean anomaly M and the eccentricity, using the method outlined by Markley (1995). It can also be found via other iterative methods solving, (Vallado & McClain, 2007)

$$M = E - e\sin E,$$

such as the ones implemented in Tudat (TU Delft, 2022).

A.3.2. Poincaré conversions

The Poincaré elements Poincaré, 1892, are:

- L_d, a Delaunay canonical variable, which is also used in the Delaunay set,
- ξ_p and η_p, the eccentric variables Smart, 1953.
- u_p and v_p, the oblique variables Smart, 1953.
- ℓ, the mean longitude.

Just as for the Delaunay elements in Section A.3.1, this section will start with the conversion from MEE to Poincaré – found in Equation A.21. This is not much of a derivation, as it is mostly taken from literature. The section thereafter, however, will derive and present the reverse, i.e., going from Poincaré to MEE (shown in Equation A.22).

The Conversion From MEE to Poincaré

The base definition is taken from Hintz (2008), yet there is a mistake where Hintz goes the step to define the Poincaré elements in terms of Equinoctial Elements rather than Keplerian elements. The derivation is trivial, and the error obvious when following it, yet a correct derivation can also be found in Lyon (2004). Following that, and translating from regular Equinoctial Elements to Modified Ones, the Delaunay elements are

$$\begin{cases} L_d = \sqrt{\mu a} = \sqrt{\dfrac{\mu p}{1-e^2}}, & u_p = k_e\sqrt{2L_d\sqrt{1-e^2}(1+\cos(i))}, \\ \xi_p = g_e\sqrt{\dfrac{2L_d}{1+\sqrt{1-e^2}}}, & v_p = h_e\sqrt{2L_d\sqrt{1-e^2}(1+\cos(i))}, \\ \eta_p = f_e\sqrt{\dfrac{2L_d}{1+\sqrt{1-e^2}}}, & \ell = M+\omega+\Omega, \end{cases} \tag{A.21}$$

where M can be found from the true anomaly θ using the method outlied for the Delaunay conversion. The eccentricity is given by the relation $e^2 = f_e^2 + g_e^2$, while the true anomaly itself is found by

$$\theta = L - \text{atan2}(g_e, f_e),$$

as $\text{atan2}(g_e, f_e) = \omega + \Omega$.

The Conversion From Poincaré to MEE

Using $f_e^2 + g_e^2 = e^2$,

$$\xi_p^2 + \eta_p^2 = \frac{2e^2L_d}{1+\sqrt{1-e^2}},$$

which can then be solved for e^2,

$$e^2 = \left\{0, \frac{(\eta_p^2+\xi_p^2)\cdot(4\cdot L_d - (\eta_p^2+\xi_p^2))}{4\cdot L_d^2}\right\}.$$

With e^2 known, one can solve for p, but also for f_e and g_e,

$$g_e = \xi_p \sqrt{\frac{1+\sqrt{1-e^2}}{2L_d}}, \qquad f_e = \eta_p \sqrt{\frac{1+\sqrt{1-e^2}}{2L_d}}.$$

Using the definition for $\cos i$ derived earlier,

$$\cos i = \frac{2}{h_e^2 + k_e^2 + 1} - 1,$$

one can also start solving towards h_e and k_e.

$$u_p^2 + v_p^2 = \frac{4L_d\sqrt{1-e^2}(h_e^2+k_e^2)}{h_e^2+k_e^2+1},$$

which can be rearranged for $h_e^2 + k_e^2$,

$$h_e^2 + k_e^2 = \frac{u_p^2 + v_p^2}{4L_d\sqrt{1-e^2} - (u_p^2+v_p^2)},$$

and, consequentially,

$$k_e = \frac{u_p}{\sqrt{4L_d\sqrt{1-e^2} - (u_p^2+v_p^2)}}, \qquad h_e = \frac{v_p}{\sqrt{4L_d\sqrt{1-e^2} - (u_p^2+v_p^2)}}.$$

Finally, L is found using the fact that

$$\text{atan2}(\xi_p, \eta_p) = \omega + \Omega,$$

and the same path as before (see Section A.3.1) is taken to go from mean anomaly M to true anomaly θ.

In summary, the conversion from Poincaré to MEE is given by

$$\begin{cases} p = \frac{1}{\mu}(1 - f_e^2 - g_e^2)L_d^2, & k_e = \frac{u_p}{\sqrt{4L_d\sqrt{1-e^2} - (u_p^2+v_p^2)}}, \\ g_e = \xi_p \sqrt{\frac{1+\sqrt{1-e^2}}{2L_d}}, & h_e = \frac{v_p}{\sqrt{4L_d\sqrt{1-e^2} - (u_p^2+v_p^2)}}, \\ f_e = \eta_p \sqrt{\frac{1+\sqrt{1-e^2}}{2L_d}} & L: \ M = \ell - \text{atan2}(\xi_p, \eta_p) \Rightarrow E \Rightarrow \theta. \end{cases} \tag{A.22}$$

A.3.3. Keplerian Conversions

This section will briefly demonstrate the conversion to (Equation A.23) and from (Equation A.24) Modified Equinoctial Elements. Do note, however, that the I parameter is assumed to be one throughout. This means that there is an anomaly at $i = 180$ deg, and thus these equations do not apply anywhere near there.

From Keplerian Elements to MEE

Following the original definition, i.e., Walker et al. (1985),

$$\begin{cases} p = a(1-e^2), & h_e = \tan\frac{i}{2}\cos\Omega, \\ f_e = e\cos(\omega+\Omega), & k_e = \tan\frac{i}{2}\sin\Omega, \\ g_e = e\sin(\omega+\Omega), & L = \Omega + \omega + \theta. \end{cases} \tag{A.23}$$

From MEE to Keplerian Elements
Reversing the equations, one can clearly see that

$$\omega + \Omega = \operatorname{atan2}(g_e, f_e) \quad \text{and} \quad \Omega = \operatorname{atan2}(k_e, h_e),$$

and thus, the following relations emerge,

$$\begin{cases} a = \dfrac{p}{1 - f_e^2 - g_e^2}, & \omega = \operatorname{atan2}(g_e, f_e) - \operatorname{atan2}(k_e, h_e), \\ e = \sqrt{f_e^2 + g_e^2}, & \Omega = \operatorname{atan2}(k_e, h_e), \\ i = 2\arctan\sqrt{h_e^2 + k_e^2}, & \theta = L - \operatorname{atan2}(g_e, f_e). \end{cases} \tag{A.24}$$

A.3.4. Kustaanheimo-Stiefel elements

The Kustaanheimo-Stiefel (Kustaanheimo & Stiefel, 1965) elements used within this work are based on the definition by Fukushima (2005). They have been slightly reworked, however, to reduce the number of elements from nine to seven without loss of information. While in Fukushima (2005) either the third or fourth element is always zero (which one depends on the sign of the first Cartesian element), that information is, in this variant, simply encoded in the sign of the first element. In the original definition, that first element, ϑ_1, is always positive. In essence, ϑ_4 is always zero in this revised definition – and thus omitted. $x_\square$ and $v_\square$ refer to the Cartesian position and velocity elements, respectively.

First, the definition as described in Fukushima (2005) is presented, in Equation A.25 and A.26. Following that, the revised definition is introduced, and a conversion to (Equation A.27 and A.28) and from (Equation A.29) Cartesian elements is provided.

The Original Definition
The original definition, as given by Fukushima (2005), is found in Equation A.25 with the derivatives given in Equation A.26. Note that the reason the state elements are conditionally defined is so that the derivatives can be uniquely defined. The radius is given by $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$, as usual.

$$\begin{cases} \vartheta_1 = \sqrt{\dfrac{r + x_1}{2}}, \quad \vartheta_2 = \sqrt{\dfrac{2}{r + x_1}}\dfrac{x_2}{2}, \quad \vartheta_3 = \sqrt{\dfrac{2}{r + x_1}}\dfrac{x_3}{2}, \quad \vartheta_4 = 0 \quad \text{for } x_1 \geq 0 \\ \vartheta_2 = \sqrt{\dfrac{r - x_1}{2}}, \quad \vartheta_1 = \sqrt{\dfrac{2}{r - x_1}}\dfrac{x_2}{2}, \quad \vartheta_4 = \sqrt{\dfrac{2}{r - x_1}}\dfrac{x_3}{2}, \quad \vartheta_3 = 0 \quad \text{for } x_1 < 0 \end{cases} \tag{A.25}$$

$$\begin{cases} \vartheta_1' = \dfrac{1}{2}\left(\vartheta_1 v_1 + \vartheta_2 v_2 + \vartheta_3 v_3\right), & \vartheta_2' = \dfrac{1}{2}\left(-\vartheta_2 v_1 + \vartheta_1 v_2 + \vartheta_4 v_4\right), \\ \vartheta_3' = \dfrac{1}{2}\left(-\vartheta_3 v_1 - \vartheta_4 v_2 + \vartheta_1 v_3\right), & \vartheta_4' = \dfrac{1}{2}\left(\vartheta_4 v_1 - \vartheta_3 v_2 + \vartheta_2 v_3\right), \end{cases} \tag{A.26}$$

The Revised Definition
This new definition was derived by first crystallizing out the two sets of equations (i.e., one for $x_1 \geq 0$ and one for $x_1 < 0$), rearranging and renaming the variables of the second set to achieve consistency with the first one, and dropping the always-zero variable ϑ_4. Finally, the one information missing without the index of the zero column, the sign of x_1, was reintroduced by just multiplying the strictly positive variable ϑ_1 by the sign of x_1. This requires a bit of bookkeeping to cancel the effects of that sign out in later equations again, yet it allows reducing the set of equations by one now-redundant column. The conversion from Cartesian is given by Equation A.27 and A.28, while the conversion back is given by Equation A.29.

$$\vartheta_1 = s_{x_1}\sqrt{\frac{r + |x_1|}{2}} \qquad \vartheta_2 = \sqrt{\frac{2}{r + |x_1|}}\frac{x_2}{2} \qquad \vartheta_3 = \sqrt{\frac{2}{r + |x_1|}}\frac{x_3}{2} \tag{A.27}$$

$$\begin{cases} \vartheta_1' = \dfrac{1}{2}\left(\vartheta_1 v_1 + \vartheta_2 v_2 + \vartheta_3 v_3\right), & \vartheta_2' = \dfrac{1}{2}\left(-\vartheta_2 v_1 + \vartheta_1 v_2\right) s_{x_1}, \\ \vartheta_3' = \dfrac{1}{2}\left(-\vartheta_3 v_1 + \vartheta_1 v_3\right) s_{x_1}, & \vartheta_4' = \dfrac{1}{2}\left(-\vartheta_3 v_2 + \vartheta_2 v_3\right), \end{cases} \tag{A.28}$$

where

$$s_{x_1} = \begin{cases} +1 & \text{if } x_1 \geq 0 \\ -1 & \text{if } x_1 < 0 \end{cases}, \qquad r = \sqrt{x_1^2 + x_2^2 + x_3^2}.$$

And back to Cartesian,

$$\begin{cases} x_1 = \left(\vartheta_1^2 - \vartheta_2^2 - \vartheta_3^2\right) s_{x_1}, & x_2 = 2\vartheta_2\vartheta_1 s_{x_1}, & x_3 = 2\vartheta_1\vartheta_3 s_{x_1}, \\ v_1 = s_{x_1}\frac{2}{r}\left(-\vartheta_2\vartheta_2' + s_{x_1}\vartheta_1\vartheta_1' - \vartheta_3\vartheta_3'\right), & v_2 = \frac{2}{r}\left(s_{x_1}\vartheta_1\vartheta_2' + \vartheta_2\vartheta_1' - \vartheta_3\vartheta_4'\right), & v_3 = \frac{2}{r}\left(\vartheta_3\vartheta_1' + \vartheta_2\vartheta_4' + s_{x_1}\vartheta_1\vartheta_3'\right), \end{cases} \tag{A.29}$$

where

$$s_{x_1} = \begin{cases} +1 & \text{if } \vartheta_1 \geq 0 \\ -1 & \text{if } \vartheta_1 < 0 \end{cases}, \qquad r = \vartheta_1^2 + \vartheta_2^2 + \vartheta_3^2.$$

Fukushima (2005) carries an additional value, what they call the 'negative Kepler energy' h_k as part of the state space. This h_k is also dropped as it does not carry any information not present in the other columns. To be precise, it can be calculated from the other columns in the following way,

$$\begin{aligned} \vartheta_1'^2 + \vartheta_2'^2 + \vartheta_3'^2 + \vartheta_4'^2 &= \frac{(v_1^2 + v_2^2 + v_3^2)(\vartheta_1^2 + \vartheta_2^2 + \vartheta_3^2)}{4}, \\ \Rightarrow v_1^2 + v_2^2 + v_3^2 &= 4 \cdot \frac{\vartheta_1'^2 + \vartheta_2'^2 + \vartheta_3'^2 + \vartheta_4'^2}{\vartheta_1^2 + \vartheta_2^2 + \vartheta_3^2}, \\ \Rightarrow h_K &= \frac{\mu}{r} - 2 \cdot \frac{\vartheta_1'^2 + \vartheta_2'^2 + \vartheta_3'^2 + \vartheta_4'^2}{r}. \end{aligned} \tag{A.30}$$

A.4. Costate Mapping from MEE to Keplerian

Applying costate mapping, as outlined in Taheri et al. (2021), to the MEE-to-Keplerian case,

$$\boldsymbol{\lambda}_{\boldsymbol{x}_{\text{MEE}}}(t)^{\mathsf{T}} \left[\frac{\partial \boldsymbol{\varrho}(\boldsymbol{x}_{\text{kep}})}{\partial \boldsymbol{x}_{\text{kep}}}\right]\Bigg|_t = \boldsymbol{\lambda}_{\boldsymbol{x}_{\text{kep}}}(t)^{\mathsf{T}}, \tag{A.31}$$

where $\boldsymbol{\varrho}$, here, is a function describing the transformation from Keplerian to MEE state. The resulting matrix, after replacing the state elements it contains in Keplerian by their equivalent value described in MEE (see Section A.3.3 for the relevant definitions), and considering the following identities for atan2,

$$\cos(\text{atan2}(y, x)) = \frac{x}{\sqrt{x^2 + y^2}} \quad \text{and} \quad \sin(\text{atan2}(y, x)) = \frac{y}{\sqrt{x^2 + y^2}},$$

is,

$$\frac{\partial \boldsymbol{\varrho}(\boldsymbol{x}_{\text{kep}})}{\partial \boldsymbol{x}_{\text{kep}}} = \begin{bmatrix} -f_e^2 - g_e^2 + 1 & \dfrac{2p\sqrt{f_e^2 + g_e^2}}{f_e^2 + g_e^2 - 1} & 0 & 0 & 0 & 0 \\ 0 & \dfrac{f_e}{\sqrt{f_e^2 + g_e^2}} & 0 & -g_e & -g_e & 0 \\ 0 & \dfrac{g_e}{\sqrt{f_e^2 + g_e^2}} & 0 & f_e & f_e & 0 \\ 0 & 0 & \dfrac{h_e\left(h_e^2 + k_e^2 + 1\right)}{2\sqrt{h_e^2 + k_e^2}} & 0 & -k_e & 0 \\ 0 & 0 & \dfrac{k_e\left(h_e^2 + k_e^2 + 1\right)}{2\sqrt{h_e^2 + k_e^2}} & 0 & h_e & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Its inverse is given by

$$\left[\frac{\partial \boldsymbol{\varsigma}(\boldsymbol{x}_{\text{kep}})}{\partial \boldsymbol{x}_{\text{kep}}}\right]^{-1} =$$

$$\begin{bmatrix} -\frac{1}{f_e^2+g_e^2-1} & \frac{2pf_e}{\left(f_e^2+g_e^2-1\right)^2} & \frac{2pg_e}{\left(f_e^2+g_e^2-1\right)^2} & 0 & 0 & 0 \\ 0 & \frac{f_e}{\sqrt{f_e^2+g_e^2}} & \frac{g_e}{\sqrt{f_e^2+g_e^2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2h_e}{\sqrt{h_e^2+k_e^2}\left(h_e^2+k_e^2+1\right)} & \frac{2k_e}{\sqrt{h_e^2+k_e^2}\left(h_e^2+k_e^2+1\right)} & 0 \\ 0 & -\frac{g_e}{f_e^2+g_e^2} & \frac{f_e}{f_e^2+g_e^2} & \frac{k_e}{h_e^2+k_e^2} & -\frac{h_e}{h_e^2+k_e^2} & 0 \\ 0 & 0 & 0 & -\frac{k_e}{h_e^2+k_e^2} & \frac{h_e}{h_e^2+k_e^2} & 0 \\ 0 & \frac{g_e}{f_e^2+g_e^2} & -\frac{f_e}{f_e^2+g_e^2} & 0 & 0 & 1 \end{bmatrix}.$$

B Catastrophic Cancellation Potential of the Throttle Function

When computing the optimal throttle, given by Izzo and Öztürk (2021)

$$u^*(t) = \frac{2\varepsilon}{2\varepsilon + \mathrm{SF}(t) + \sqrt{4\varepsilon^2 + \mathrm{SF}(t)^2}} \tag{B.1}$$

where $\varepsilon \in [0, 1]$ is the continuation parameter (see), and $\mathrm{SF}(t)$ the switching function.

When the switching function goes toward infinity, the denominator does too, and the throttle setting hence goes to zero. In the other direction, however, when the switching function approaches large negative values, the denominator approaches 2ε, and hence the throttle is expected to approach one. When dealing with digital computing, however, one must consider the limited accuracy of floating-point arithmetic. To be specific, the phenomenon occurs when two numbers close to each other are subtracted from each other: This is known as catastrophic cancellation (Oliveira & Stewart, 2006). To visualize what happens in that case, the denominator of Equation B.1 for various large negative values of SF has been plotted in Figure B.1. One can clearly see that from a certain magnitude of SF on, the behavior becomes erratic and eventually settles on a value of 0. The value of the denominator should, however, remain at $2 \cdot \varepsilon$, and this accuracy thus has to be accounted for manually.

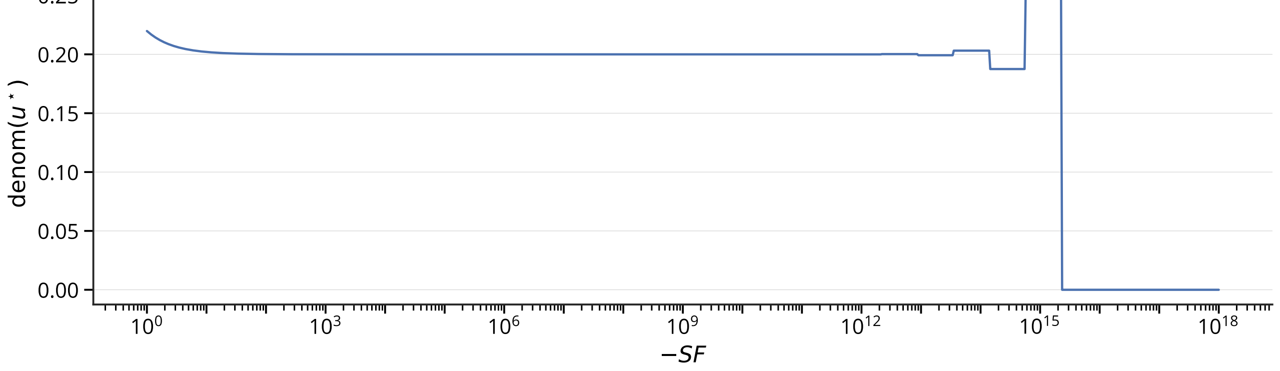


Figure B.1: Catastrophic cancellation demonstrated on the denominator of Equation B.1 with $\varepsilon = 0.1$.

C Literature Overview

This appendix contains various overview tables covering the relevant available literature. Section C.1 provides an overview of the neural networks used, including their features and targets. It also lists the neural network architecture and learning strategies used. This is followed by Section C.2, which provides an overview of the strategies used in literature to generate (training) trajectory data.

C.1. Neural Networks Trajectory Optimization Found in Literature

This section's tables use a case identifier to uniquely identify the different networks used. Some published literature only contains a single network, making the case identifier rather unique, yet others contain a number of them. For all case identifiers, the mapping to the corresponding reference can be found in Table C.5.

Table C.1 lists the different feature sets used in literature and Table C.2 does so for the targets. This is followed by an overview of the different network architectures, i.e., node count, layer count, and activation functions. This can be found in Table C.3. Finally, Table C.4 provides a brief synopsis of the learning strategies employed in the surveyed literature.

C.2. Minimum-Fuel Approaches in Literature

Table C.6 gives an overview of the approaches used in literature to solve the minimum-fuel transfer problem.

Table C.1: Overview of features found in literature.

x = used, - = unsuccessfully considered, Δ = difference used

Case Identifier¶	Features											
	mod. equ. elem.	cartesian elem.	kepler elem.	equinoct. elem.	delaunay elem.	init. mass	max init. mass	transfer time	spec. ang. mom.	spec. orb. ener.	ΔV Lambert	other
Yin2020		x‡	i+e									
Mereta2017			x§			x	x	x				
Xie2021	x+Δ	-	-			x		x	$\vec{x}$	x	x	
Viavattene2022	x	-	-	-	-	x						
Li2019_MEE_TOF_NEA	x											
Li2019_CAR_TOF_NEA		x										
Li2019_MEE_COS_NEA	x											
Li2019_CAR_COS_NEA		x										
Li2019_MEE_CON_NEA	x					x						
Li2019_CAR_CON_NEA		x				x						
Li2019_MEE_TOF_MARS	x											
Li2019_CAR_TOF_MARS		x										
Li2019_MEE_COS_MARS	x											
Li2019_CAR_COS_MARS		x										
Li2019_MEE_CON_MARS	x					x						
Li2019_CAR_CON_MARS		x				x						
Li2020_TOF	-	x†	-			x			x	x		
Li2020_MASS	x*	-	-			x		x	x	x	x	
Li2020_ΔV	-	-	x†					-	-	-	-	
Li2019a_LEO_HiT	-	x	-									
Li2019a_LEO_LoT	-	x	-									
Li2019a_GEO_HiT	-	x	-									
Li2019a_GEO_LoT	-	x	-									
Miller2019a_POL		x										
Miller2019a_CRIT		x										
LaFarge2021_POL		x+Δ				x						x‖
LaFarge2021_CRIT		x+Δ				x						x‖
Arora2020_PL			e						x			x**
Arora2020_NONPL			e+i						x			x**
Miller2019_POL12		x+Δ				x		x				x††
Miller2019_VAL12		x+Δ				x		x				x††
Miller2019_POL3		x+Δ				x		x				x††
Miller2019_VAL3		x+Δ				x		x				x††
Izzo2021_VAL	x					x						
Izzo2021_POL	x					x						

*The state of departure body at time of departure, state of target body at time of arrival.
†Departure and target body state at time of departure.
‡Only the position is used as feature.
§$\cos\theta$, $|\Delta a|$, $|\Delta e|$
¶See Table C.5 for mapping to references.

Table C.2: Overview of targets found in literature.

x = used, - = unsuccessfully considered, Δ = difference used

Case Identifier[¶]	Target						
	costate vector	final mass	propellant mass	time of flight	control vector	ΔV	score
Yin2020	x		x	x			
Mereta2017		x					
Xie2021							x[‡‡]
Viavattene2022				x		x	
Li2019_MEE_TOF_NEA				x			
Li2019_CAR_TOF_NEA				x			
Li2019_MEE_COS_NEA	x						
Li2019_CAR_COS_NEA	x						
Li2019_MEE_CON_NEA					x		
Li2019_CAR_CON_NEA					x		
Li2019_MEE_TOF_MARS				x			
Li2019_CAR_TOF_MARS				x			
Li2019_MEE_COS_MARS	x						
Li2019_CAR_COS_MARS	x						
Li2019_MEE_CON_MARS					x		
Li2019_CAR_CON_MARS					x		
Li2020_TOF				x			
Li2020_MASS		x					
Li2020_ΔV						x	
Li2019a_LEO_HiT					x		
Li2019a_LEO_LoT					x		
Li2019a_GEO_HiT					x		
Li2019a_GEO_LoT					x		
Miller2019a_POL					x		
Miller2019a_CRIT							x
LaFarge2021_POL					x		
LaFarge2021_CRIT							x
Arora2020_PL				x			
Arora2020_NONPL				x			
Miller2019_POL12					x		
Miller2019_VAL12							x
Miller2019_POL3					x		
Miller2019_VAL3							x
Izzo2021_VAL		x					
Izzo2021_POL					x		

[‖]The Jacobi constant and its target value.
[**]Relative weights of different vector components; implementation specific.
[††]The current time.
[‡‡]Convergence/divergence classification.

Table C.3: Overview of neural network architectures found in literature.

x = used, - = unsuccessfully considered

	Architecture		Activation function			Output activation		
Case Identifier[‡]	n_layers	n_neurons	sigmoid	ReLU	tanh	tanh	linear	sigmoid
Yin2020	6	64	-	x	-			
Mereta2017	2	80		x			x	
Xie2021	6	459		x[*]			—unknown—	
Viavattene2022	2	80	x		-		—unknown—	
Li2019_MEE_TOF_NEA	4	254	x	-	-		x	
Li2019_CAR_TOF_NEA	3	460	-	x	-		x	
Li2019_MEE_COS_NEA	4	452	-	-	x		x	
Li2019_CAR_COS_NEA	4	350	-	-	x		x	
Li2019_MEE_CON_NEA	3	382	-	x	-	x	-	
Li2019_CAR_CON_NEA	6	390	-	x	-	-	x	
Li2019_MEE_TOF_MARS	4	264	x	-	-		x	
Li2019_CAR_TOF_MARS	5	386	-	x	-		x	
Li2019_MEE_COS_MARS	5	444	x	-	-		x	
Li2019_CAR_COS_MARS	3	298	-	-	x		x	
Li2019_MEE_CON_MARS	4	392	-	x	-	-	x	
Li2019_CAR_CON_MARS	3	436	-	-	x	x	-	
Li2020_TOF	4	218	x	-	-		x	
Li2020_MASS	5	170	x	-	-		x	
Li2020_ΔV	4	210	x	-	-		x	
Li2019a_LEO_HiT	6	350	-	-	x		—unknown—	
Li2019a_LEO_LoT	6	452	-	x	-		—unknown—	
Li2019a_GEO_HiT	4	176	x	-	-		—unknown—	
Li2019a_GEO_LoT	5	138	x	-	-		—unknown—	
Miller2019a_POL	3	32		x		x		
Miller2019a_CRIT	3	32		x			x	
LaFarge2021_POL	3	120/60/30			x	x		
LaFarge2021_CRIT	3	120/24/5			x		x	
Arora2020_PL	1	150						
Arora2020_NONPL	1	150						
Miller2019_POL12	3	32		x		x		
Miller2019_VAL12	3	32		x			x	
Miller2019_POL3	3	150/87/50		x		x		
Miller2019_VAL3	3	150/27/5		x			x	
Izzo2021_VAL	9	200		x[†]			x	
Izzo2021_POL	3	200		x[†]			x	x

[*]ReLU, leaky ReLU, and ELU were tested; leaky ReLU was selected.
[†]The ReLU variation softplus was used.
[‡]See Table C.5 for mapping to references.

Table C.4: Overview of neural network learning architectures found in literature.

x = used, - = unsuccessfully considered

	Optimizer						Decay model		
Case Identifier[‡]	AGD	MGD	GD	SGD	LM	AMSGrad	ED	NED	Other
Yin2020	unknown								None
Mereta2017				x					None
Xie2021	x								Quadratic
Viavattene2022					x				None
Li2019_MEE_TOF_NEA	x	-	-				x	-	
Li2019_CAR_TOF_NEA	x	-	-				x	-	
Li2019_MEE_COS_NEA	x	-	-				x	-	
Li2019_CAR_COS_NEA	x	-	-				x	-	
Li2019_MEE_CON_NEA	x	-	-				-	x	
Li2019_CAR_CON_NEA	x	-	-				-	x	
Li2019_MEE_TOF_MARS	x	-	-				x	-	
Li2019_CAR_TOF_MARS	-	x	-				x	-	
Li2019_MEE_COS_MARS	x	-	-				-	x	
Li2019_CAR_COS_MARS	x	-	-				x	-	
Li2019_MEE_CON_MARS	x	-	-				x	-	
Li2019_CAR_CON_MARS	x	-	-				-	x	
Li2020_TOF	x	-	-				-	x	
Li2020_MASS	x	-	-				-	x	
Li2020_ΔV	x	-	-				-	x	
Li2019a_LEO_HiT	x	-	-				-[§]	-[§]	
Li2019a_LEO_LoT	x	-	-				-[§]	-[§]	
Li2019a_GEO_HiT	x	-	-				-[§]	-[§]	
Li2019a_GEO_LoT	x	-	-				-[§]	-[§]	
Miller2019a_POL	unknown								None
Miller2019a_CRIT	unknown								None
LaFarge2021_POL	x								Custom
LaFarge2021_CRIT	x								Custom
Arora2020_PL					x				Custom
Arora2020_NONPL					x				Custom
Miller2019_POL12	x								Schedule
Miller2019_VAL12	x								Schedule
Miller2019_POL3	x								Schedule
Miller2019_VAL3	x								Schedule
Izzo2021_VAL						x			None
Izzo2021_POL						x			None

[§]Unclear which one is selected.

Table C.5: Mapping between case identifiers and corresponding references.

Reference	Case Identifier
Yin et al. (2020)	Yin2020
Mereta et al. (2017)	Mereta2017
Xie and Dempster (2021)	Xie2021
Viavattene and Ceriotti (2022)	Viavattene2022
H. Li, Baoyin, and Topputo (2019)	Li2019_MEE_TOF_NEA
	Li2019_CAR_TOF_NEA
	Li2019_MEE_COS_NEA
	Li2019_CAR_COS_NEA
	Li2019_MEE_CON_NEA
	Li2019_CAR_CON_NEA
	Li2019_MEE_TOF_MARS
	Li2019_CAR_TOF_MARS
	Li2019_MEE_COS_MARS
	Li2019_CAR_COS_MARS
	Li2019_MEE_CON_MARS
	Li2019_CAR_CON_MARS
H. Li et al. (2020)	Li2020_TOF
	Li2020_MASS
	Li2020_ΔV
H. Li, Topputo, and Baoyin (2019)	Li2019a_LEO_HiT
	Li2019a_LEO_LoT
	Li2019a_GEO_HiT
	Li2019a_GEO_LoT
Miller and Linares (2019)	Miller2019a_POL
	Miller2019a_CRIT
LaFarge et al. (2021)	LaFarge2021_POL
	LaFarge2021_CRIT
Arora and Dutta (2020)	Arora2020_PL
	Arora2020_NONPL
Miller et al. (2019)	Miller2019_POL12
	Miller2019_VAL12
	Miller2019_POL3
	Miller2019_VAL3
Izzo and Öztürk (2021)	Izzo2021_VAL
	Izzo2021_POL

Table C.6: Overview of approaches to solve the minimum-fuel transfer problem in literature.

publication	homotopy	state repr.	use-case[*]	free		time/traj.	convergence
				time	POP		
Rosa Sentinella and Casalino (2006)	-	Cartesian	E-M w/ GA	yes	no	~75 s	~30 %
Epenoy (2023)	-	Cartesian	rendez-vous	no	no	~1 to 701 s[§]	unknown
Wang and Grant (2018)	-	spherical[†]	E-M	no	no	~2.49 s	unknown
Jawaharlal Ayyanathan and Taheri (2022)	throttle smoothing	Cartesian/MEE[‡]	E-D, GTO-GEO	no	no	unclear	~66 %[¶]
Zhu et al. (2017)	from min-thrust	Cartesian	E-M, orbit transfers	no	no	>30 min	unknown
T. Li et al. (2021)	from min-thrust	Cartesian	E-M, E-D	no	no	unknown	unknown
Jiang et al. (2012)	logarithmic barrier	Cartesian	E-V, E-J w/ GA	no	no	~28 s	unknown
Tang et al. (2018)	logarithmic barrier	cylindrical	E-M, E-V	no	no	unknown	unknown
Izzo and Öztürk (2021)	logarithmic barrier	MEE	E-V	yes	yes	N/A	N/A
Guo et al. (2012)	quadratic	Cartesian	E-V, E-A	no	no	unknown	unknown
Pan et al. (2019)	quadratic	Cartesian	E-M w/ GA	no	no	unknown	unknown
Morelli et al. (2022)	quadratic	Cartesian	E-D	no	no	~6 s	~58 %

[*]V = Venus, E = Earth, M = Mars, J = Jupiter, A = Asteroid, D = Dionysus, w/ GA = with gravity-assist manoeuvres.
[†]Radius, two angles, and the derivatives of the three. This is different from the spherical definition used in this.
[‡]Cartesian is used to estimate the costates, which are then mapped to MEE.
[§]Increases as the number of coast/thrust arcs increases. 1 s cooresponds to one coast and two thrust arcs.
[¶]Earth->Dionysus has a convergence of ~66 %, GTO-GEO's is ~45 to 55 % (the high value reached without J2, and using the h-e set)

D Towards Earth-Venus Convergence

This work ended up using Earth-Mars transfer trajectories only, which was not the intention from the start. The reason for this is, however, that Venus convergence could not be achieved in a timely manner. The attempts that were made are documented in this appendix to provide a potential starting point for future work.

The initial path that was intended to take towards generating any data was to use the homotopy outlined in Izzo and Öztürk (2021), which incidentally also deals with Earth-Venus transfers. When convergence for Earth-Mars was achieved, however, homotopy turned out to be unnecessary.

D.1. Scaling of the Objective Function Elements

The objective function used is $\Delta\mathcal{M}$. As discussed in Paper/Solving the Shooting Problem (p. 7), the choice of this can have an impact on convergence. This also applies to Earth-Venus trajectories, as can be seen in Figure D.1, where a clear difference is visible between different values of the individual scaling components. What is also visible, however, is that none of them reliably converge, and only outliers go anywhere near the goal of $\Delta\mathcal{M} < 10^{-5}$. The choice of this alone is thus not the missing piece for convergence.

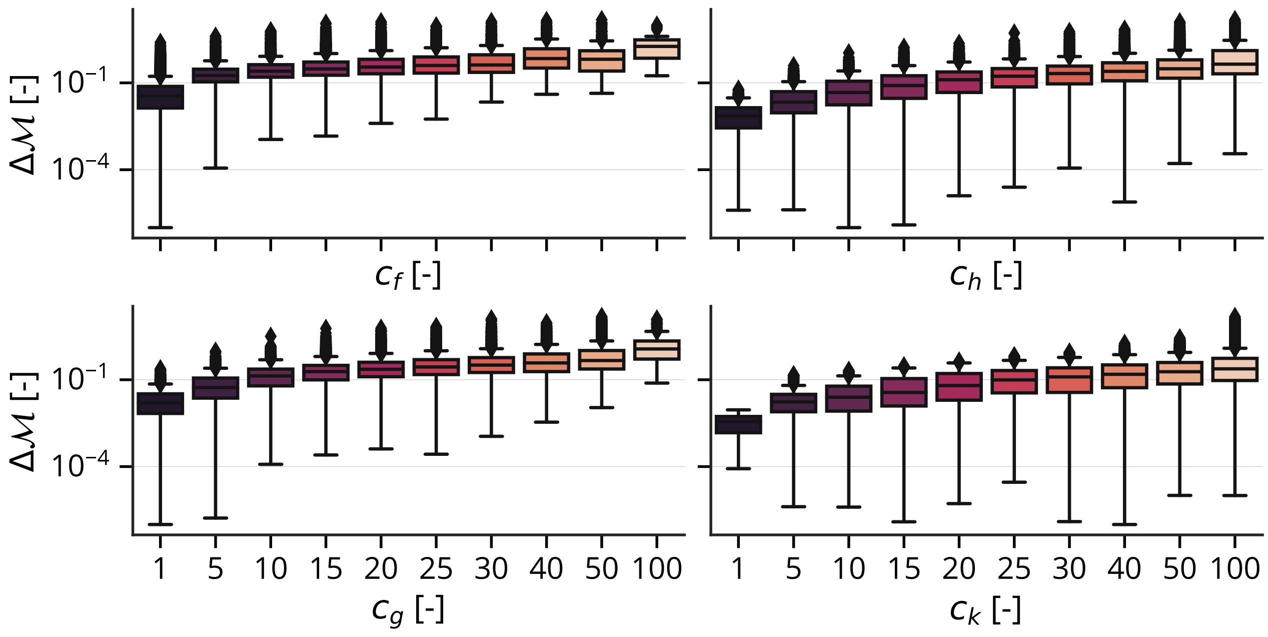


Figure D.1: $\Delta\mathcal{M}$ mismatch at Earth, as a function of the costate scaling parameters.

D.2. Homotopy Paths and their Convergence

The implementation initially followed in this work, Izzo and Öztürk (2021), made use of homotopy for convergence. As the solution found for Earth-Mars trajectories does not work for Earth-Vesus transfers, it thus seems reasonable to attempt homotopy once more. Yang et al. (2019) uses an iterative homotopy algorithm, reducing the step size on convergence failure. This did not turn out to be successful, as when convergence was not achieved at one step, the step size was reduced so much to be meaningless in the end. Alternatively, several homotopy paths were tried, and the results looked at. The homotopy paths tried can be found in Figure D.2 and are characterized by a parameter h_a, which describes the shape of said trajectory.

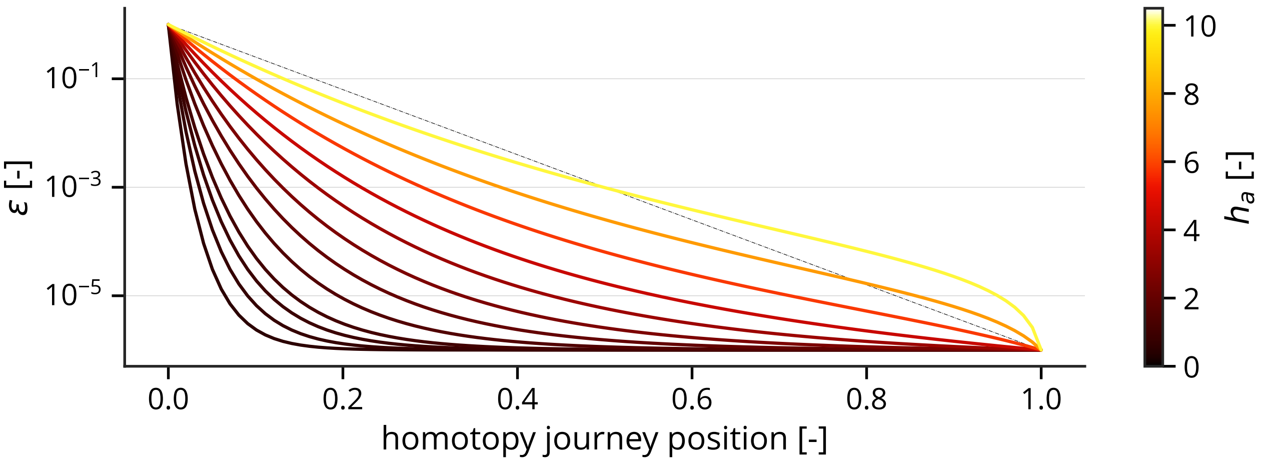


Figure D.2: Homotopy paths' attempted for Venus convergence, characterized by the parameter a, and the homotopy-step $\in [0, 1]$.

Figure D.3 and D.4 show the resulting $\Delta\mathcal{M}$ for the different homotopy paths. Once again, the same conclusion as for the variation of the scaling parameters can be drawn: Different paths yield different results, yet the differences are not major enough to produce a viable candidate. While solutions do exist – Izzo and Öztürk (2021) solves only one trajectory and generates the others suboptimally by a slight variation of the final costates, for example – more work is needed to make the specific implementation used in this work for Earth-Venus trajectories.

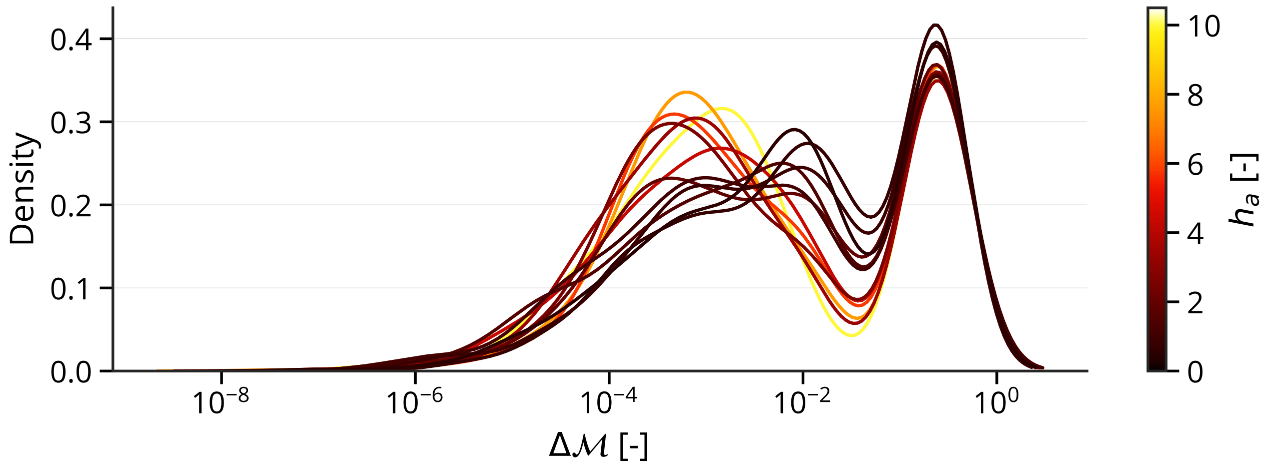


Figure D.3: Distribution of the trajectories following a given homotopy path, as kernel density estimate (KDE) plot.

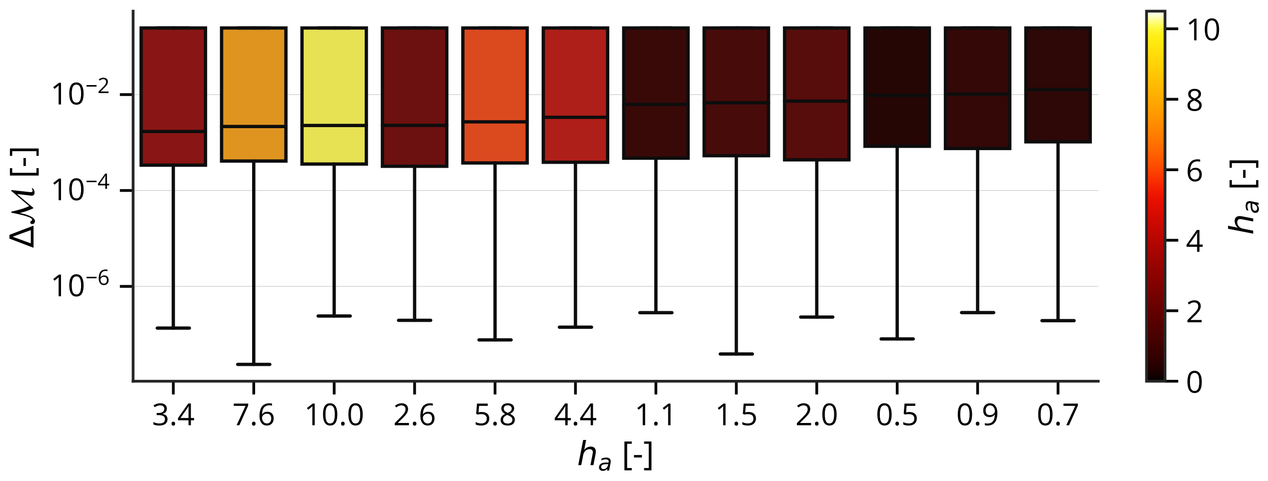


Figure D.4: Distribution of the trajectories following a given homotopy path, sorted by median.