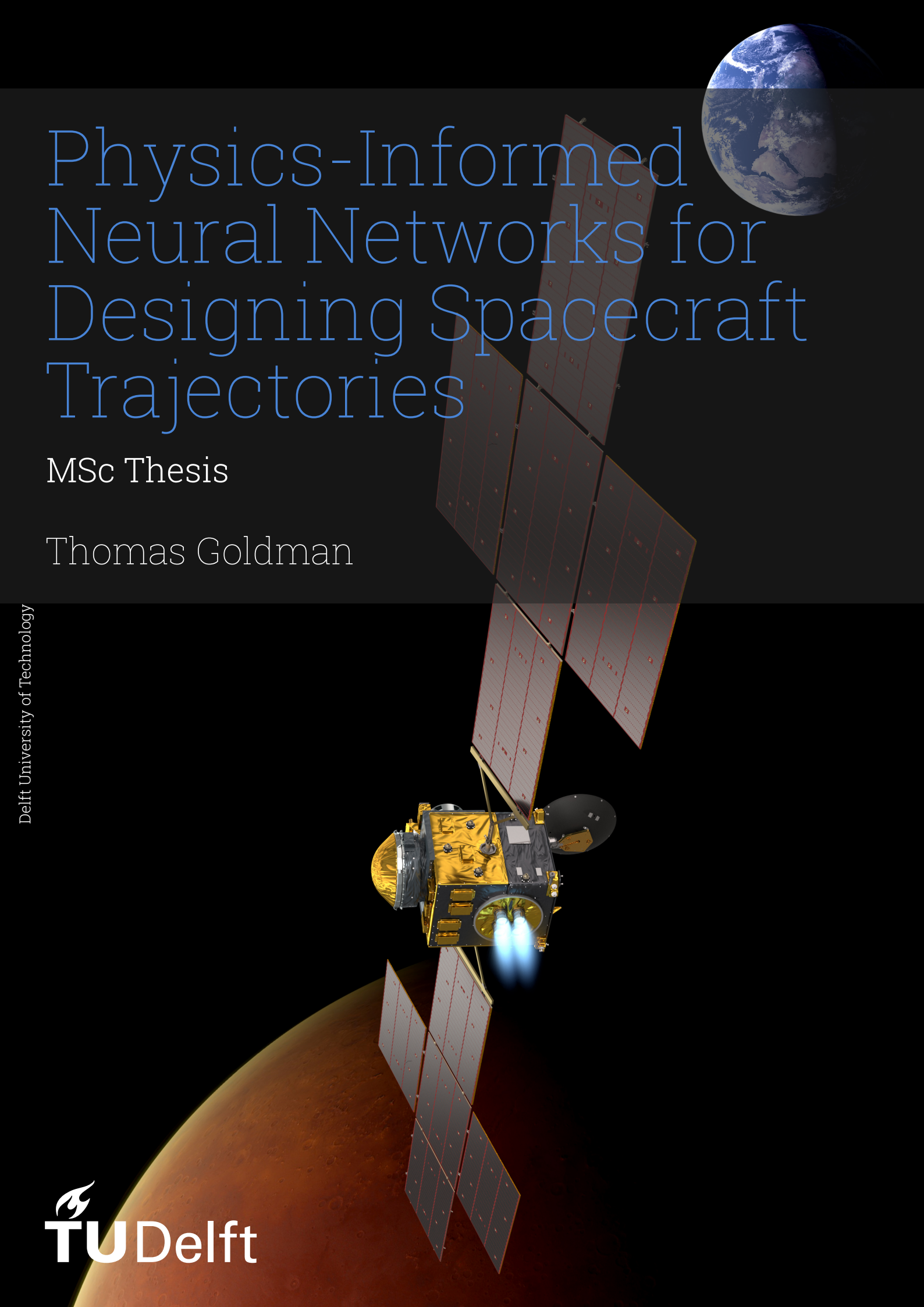
Physics-Informed Neural Networks for Designing Spacecraft Trajectories

MSc Thesis

Thomas Goldman

Delft University of Technology



TU Delft

Physics-Informed Neural Networks for Designing Spacecraft Trajectories

MSc Thesis

by

Thomas Goldman

to obtain the degree of

Master of Science

To be defended publicly on Friday 24 November, 2023 at 13:00

Student number:	5389100	
Thesis Committee:	Dr. Ir. E. Mooij	Committee chair
	Ir. K.J. Cowan	Supervisor
	Dr. I. Aklay	External examinator
Institute:	Astrodynamics & Space Missions section Faculty of Aerospace Engineering Delft University of Technology	

Cover: 'Earth Return Orbiter' - ESA/Science Office `https://www.esa.int/ESA_Multimedia/Images/2019/05/Earth_Return_Orbiter`

TU Delft

Preface

In October 2022, the brief mention of something called a Hamiltonian Neural Network (HNN) sparked a mutual interest between Kevin and myself. Although at the time neither of us had any idea what it entailed, we decided to explore this unknown domain of machine learning. The result was a comprehensive literature study with an outline on how to research the capabilities of the HNN within space applications. We went with the idea of developing an unsupervised HNN to compete with numerical integrators, a very daunting enterprise indeed. About at the same time when it was realized that exceeding an integrator might be too ambitious, it was realized how the method could be converted to simultaneously solve a two-boundary-value continuous optimal control problem. Being an astrodynamics nerd, this smelled like low-thrust transfers and the remaining story is told in this report.

I would like to express great appreciation to my supervisor Kevin Cowan, for granting me a great deal of freedom in doing research while helping me to keep my sanity. I would also like to extend my heartfelt thanks to my girlfriend Lara, my parents and brother for their unwavering support throughout this journey. Their patience in enduring my numerous conversations, explanations (to myself), and at times, incoherent remarks about this work has been truly admirable. Their encouragement and understanding have played an invaluable role in bringing this research to fruition. Thanks to all my friends, you know who you are. But in the end, this work is dedicated to Quincy, my Labrador, who passed away in the final weeks of this research and couldn't be here to see the end of it.

Abstract

This thesis proposes an unsupervised Physics-Informed Neural Network (PINN) for solving optimal control problems with the direct method to design and optimize transfer trajectories. The network adheres analytically to boundary conditions and includes the objective fitness as regularization in its loss function. A test scenario of a planar Earth-Mars low-thrust optimal-fuel transfer and rendezvous is chosen. Comprehensive examination of training strategies reveals that convergence is highly dependent on the initialization of the network and that correctly balancing loss terms is essential for navigating the intricate loss landscape. This balance is achieved by carefully selecting loss weights and implementing a refined learning rate schedule. Comparative analysis to hodographic shaping solutions demonstrates that the PINN effectively identifies near-optimal solutions across a wide range of initial and final constraints for the Earth-Mars transfer problem, with a maximum improvement of 4.5 km s^{-1} and median improvement of 0.55 km s^{-1}. The PCNN shows promise as a preliminary design tool for trajectory optimization in nonlinear dynamics.

Contents

Nomenclature

List of Acronyms

AD	Automatic Differentiation
ANN	Artificial Neural Network
AU	Astronomical Unit (Distance Earth-Sun)
BC	Boundary Condition
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DE	Differential Equation
DNN	Deep Neural Network
ECLIPJ2000	Reference frame based on the ecliptic
ESA	European Space Agency
FEM	Finite Element Method
FFNN	Feed-Forward Neural Network
G&CNETs	Guidance and Control Networks
GPU	Graphics Processing Unit
HNN	Hamiltonian Neural Network
HS	Hodographic Shaping
IC	Initial Condition
NaN	Not a number
NASA	National Aeronautics and Space Administration
NLP	Non Linear Programming
NN	Neural Network
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
PCNN	Physics-Constrained Neural Network
PDE	Partial Differntial Equation
PINN	Physics-Informed Neural Network
RK	Runge-Kutta, a constant time-step numerical integrator
RKDP	Runge-Kutta-Dorman-Prince, a variable time-step numerical integrator
RKF	Runge-Kutta-Fehlberg, a variable time-step numerical integrator
RNN	Recurrent Neural Network
RSW	Radial (R), Along-track (S) and Cross-track (W) reference frame
SEP	Solar Electric Propulsion
TUDAT	TU Delft Astrodynamics Toolkit
X-TFC	Extreme Theory of Functional Connections

List of Symbols

E_0	Constant Energy of Hamiltonian System
G	Gravitational Constant
I_{sp}	Specific impulse
J	Cost function
L	Running Cost
M	Number of training points
N	Entries in a state vector
P	Neurons in a NN input layer
Q	Entries in a control vector or Neurons in a NN output layer
S	Sensitivity matrix

ΔV	Delta-V
$\Delta \tilde{x}$	Linearized difference in state
Ω	Domain
Φ	Endpoint cost
β	Polar angle for thrust vector
η	Learning rate
γ	Environmental Parameters
$\mathbf{e}_r, \mathbf{e}_\theta$	Unit vectors spanning the basis for polar coordinates
$\mathbf{e}_x, \mathbf{e}_y$	Unit vectors spanning the basis for cartesian coordinates
$\mathbf{a}$	Empirical Acceleration in RSW frame
$\mathbf{h}$	NN hidden layer
$\mathbf{p}$	Weights and biases in a NN
$\mathbf{u}$	Control vector
$\mathbf{x}$	Spacecraft State Vector
$\mathbf{y}$	NN output layer
$\mathbf{z}$	State vector of arbitrary system
$\mathcal{B}$	Boundary conditions
$\mathcal{F}$	Partial Differential Equation
$\mathcal{H}$	Hamiltonian
$\mathcal{I}$	Initial Conditions
$\mathcal{L}_i$	Dynamics loss term of equation of motion i
$\mathcal{L}_o$	Objective loss term
$\mathcal{L}_{\mathbf{reg}}$	Regularization Loss
$\mathcal{L}$	Loss
$\mathcal{N}_i$	Neural network output
$\mathcal{N}$	Normal distribution
$\mathcal{R}_{c\mapsto p}$	Rotation from cartesian to polar coordinates velocities
$\mathcal{R}_{p\mapsto c}$	Rotation from polar to cartesian coordinates velocities
μ	Standard Gravitational Parameter or centre of normal distribution
ω_o	Objective loss weight
ω_w	Dynamics loss weight
$\oplus$	Earth
ϕ	Control angle
σ	Standard deviation or Activation Function
$\tilde{m}_p$	Propellant mass for correction
a	Constraint steepness parameter
b	Bias in NN
dx, dv, dm	Verificaiton metrics: position, velocity and mass
$f(\cdot)$	Ordinary Differential Equation or state constraint equation
$g(\cdot)$	Control constraint equation
g_0	Standard gravity
$h(\cdot)$	State constraint equation
m_p	Propellant mass
m	Spacecraft mass
n, i, j, k	Iterable
r, θ, v_r, v_θ	Polar coordinates
u_ϕ, u_T	Control in polar coordinates: control angle and thrust magnitude
$u_{\mathbf{max}}$	Maximum thrust
u_x, u_y	Control in cartesian coordinates
$v_\oplus$	Earth circular velocity
w	Weight in NN
x, p	Canonical coordinates: generalized coordinate and conjugate momentum
x, y, v_x, v_y	Cartesian coordinates
$x_v, y_v, v_{x,v}, v_{y,v}$	Verification Cartesian coordinates
t	Time

1 Introduction

The space industry is undergoing rapid transformation. While space exploration was initially led by national space agencies of powerful nations, today, commercial entities and smaller nations play a crucial role in shaping the industry and benefiting from activities in space. As a result, the space sector is evolving into a competitive and globally diverse industry. Prominent shifts in the space environment encompass the rise of mega constellations offering communication services [1], a notable surge in the accessibility of launch vehicles [2], and the miniaturization of satellite systems [3]. In the wake of increasingly more available computational power, space engineers are considering increasingly complicated space mission designs in order to maximize scientific output or commercial return. Simultaneously, a concurrent design approach, involving numerous experts engaging in brief sessions to explore diverse mission concepts and early trade-offs in planning and satellite subsystems, has gained widespread acceptance among major space agencies such as the European Space Agency (ESA) [4]. In order to keep up with the ambitions, technological advancements are being pursued on many aspects of space systems. Examples are increased autonomy in satellite systems [5] and specialized materials capable of sustaining some of the most demanding and hostile operational environments [6]. Solar electric propulsion (SEP) systems are an innovation that have significantly increased the amount of ΔV that can be accumulated by burning propellant highly efficiently through increased effective exhaust velocities. This thesis will focus on the challenges experienced in preliminary trajectory design and specifically those associated with low-thrust transfers performed by electric propulsion systems.

Low-thrust transfer trajectories involve spacecraft using propulsion systems that apply a sustained, small thrust force over an extended duration. Specific impulse of electric propulsion systems range between 1000 and 10000 s [7] making them vastly more efficient than chemical engines. The first implementation of a solar electric propulsion system was the NASA Deep Space 1 mission that accomplished a fly-by of both asteroid 9969 Braille and comet 19P/Borrelly [8]. Its purpose was to demonstrate the technology and the first NASA exploration mission that made use of SEP was the Dawn spacecraft. Dawn has achieved the highest ΔV accumulation of any spacecraft with more than 11 km s^{-1} [9]. In its lifetime it has orbited both Vesta and Ceres making it the only spacecraft that has orbited two extraterrestrial bodies. According to Russell et al. [10], if chemically propelled spacecraft were employed to meet the same scientific criteria, the expenses would have increased by over 1 billion dollars compared to the expenses of the Dawn mission, as it would have required two separate spacecraft. Unfortunately, low-thrust mission concepts are notoriously difficult to design. In order to steer the spacecraft to a required destination, a continuous optimal control problem has to be solved. Traditionally, techniques to solve these problems can be classified as direct or indirect methods. The indirect method restates the problem as a two-point boundary value problem utilizing the Pontryagin Maximum Principle (PMP) [11], which relies on the calculus of variations. It sets forth the first-order optimality conditions that the solution must meet in order to be considered optimal. On the other hand, the direct method transforms the low-thrust transfer into a nonlinear programming problem (NLP). This can be addressed through gradient-based or heuristic approaches, which aim to determine a set of control variables that directly optimize an objective function [12]. A survey of tools used in academia and industry for designing low-thrust transfers revealed that, in general, low-thrust optimization methods tend to be complex

and challenging to integrate into a concurrent design phase. They often lack the ability to incorporate mission constraints, perform multi-objective optimization, and efficiently explore large design spaces [13]

In parallel to the space industry, the field of machine learning has experienced immense advancements over the last decade. Most notably the artificial neural network (ANN) has manifested itself in everyday life by performing tasks as image recognition [14], language processing [15] and sequential decision making [16]. The overparameterization of the ANN's internal structure and statistical training procedure often result in thinking of the method as a black box. Because of this, scientists express reservations regarding the interpretability of ANN's when employing them in scientific contexts or situations demanding accountability or a high level of reliability [17]. However, it is worth noting that ANN have demonstrated their worth in scientific exploration, as seen in areas such as material science [18] and geoscience [19]. Surprisingly, the adaptation of machine learning methods to aid spacecraft trajectory design has been slow due to the unavailability of large data sets and often non-straightforward implementations [20]. Nevertheless, the opportunity of augmenting spacecraft flight dynamics related tools with machine learning is actively being researched [21, 20]. For example, Guidance and Control Networks (G&CNETs) in spacecraft employ deep learning to generate guidance profiles by mapping a spacecraft state to a control output [22]. This enables on-board neural networks, pre-trained on Earth, to provide optimal control feedback during missions enhancing spacecraft autonomy, especially in landing scenarios [23] and low-thrust transfers [24]. Another example of machine learning research on space mission design is found in the context of trajectory optimization. A proposed application is to have a machine learning model learn the costly fitness function such that optimization schemes can be more efficient [25, 26, 27].

The concept of a Physics-Informed Neural Network (PINN) has recently been introduced as a powerful extension on the standard ANN framework [28, 29]. These type of networks insert additional information about the physical process under consideration into the network in an attempt to better comply with physical models and partially lift the black box nature of the neural network. The loss function of the network is penalized by an additional term reflecting the implicit differential equations describing the governing physical model. Initial and boundary conditions can also be embedded by additional loss terms or analytically through a constraint layer. PINNs can be configured to perform a physics-aided regression on data, solve inverse problems and find solutions to differential equations in an unsupervised manner. From a computational science perspective, there is extensive research on PINNs, primarily centered around solving partial differential equations (PDE) [30, 31]. Very recently, applications of PINNs in spacecraft flight dynamics have started to appear in the literature. For example, Schiassi et al. [32] developed an unsupervised PINN-based method to solve optimal control problems in an indirect approach and use this to find optimal solutions of planar low-thrust trajectories. A different application is proposed by Martin et al., who modeled the gravity fields of the Earth and Moon [33] as well as those of smaller bodies [34]. Finally, Scorsoglio et al. [35] have developed a method to use PINNs to solve orbit determination problems.

In this work, an unsupervised PINN based method to solve optimal control problems will be designed and explored as a method to enhance low-thrust mission design. Specifically, a PINN with a direct approach to optimal control problems as proposed by Mowlavi et al. [36] and originally applied to PDE will be transferred to the field of spacecraft trajectory optimization. The goal is to ascertain if this approach can address current limitations in optimization tools and provide additional value alongside the existing tool set, particularly within the realm of concurrent design and low-thrust transfers. The research is written in a conference paper format (Chapter 2) that follows the guidelines of the Astrodynamics Specialist Conference. For that purpose, it is a concisely written manuscript that reports on the well-established findings of this research. Several appendices have been added in order to guide the reader with this presentation of the work. First of all, a more thorough description of some of the methods used is provided in appendix A. Secondly, explicit clarification on steps undertaken to verify and validate this work are elaborated on in appendix B. Thirdly, some additional findings that are not essential to report on in the context of a conference paper but might be valuable for future students are included in appendix C. Finally, extended conclusions and recommendations for future development of the method proposed in this work are added in appendix D.

2

Paper

The paper is written in the format of the AAS/AIAA Astrodynamics Specialist Conference [1] and is therefore limited to 20 pages.

[1] https://space-flight.org/conferences.html

(Preprint) AAS 23-XXX

AN UNSUPERVISED PHYSICS-CONSTRAINED NEURAL NETWORK FOR FINDING OPTIMAL LOW-THRUST TRANSFER TRAJECTORIES WITH THE DIRECT METHOD

Thomas Goldman[*], Kevin Cowan[†]

An unsupervised Physics-Constrained Neural Network (PCNN) for solving optimal control problems with the direct method to design and optimize transfer trajectories is proposed. The network adheres analytically to boundary conditions and includes the objective fitness as regularization in its loss function. A test scenario of a planar Earth-Mars low-thrust optimal-fuel transfer and rendezvous is chosen. Comprehensive examination of training strategies reveals that convergence is highly dependent on the initialization of the network and that correctly balancing loss terms is essential for navigating the intricate loss landscape. This balance is achieved by carefully selecting the loss weights and implementing a refined learning rate schedule. Comparative analysis to hodographic shaping solutions demonstrates that the PCNN effectively identifies near-optimal solutions across a wide range of initial and final constraints for the Earth-Mars transfer problem, with a maximum improvement of 4.5 km s^{-1} and median improvement of 0.55 km s^{-1}. The PCNN shows promise as a preliminary design tool for trajectory optimization in nonlinear dynamics.

INTRODUCTION

Low-thrust transfer trajectories are maneuvers performed by spacecraft with propulsion systems that deliver a continuous small thrust force over the course of an extended period. Examples of such propulsion systems are solar electric propulsion engines and solar sails. Electric propulsion systems are highly efficient due to the large effective exhaust velocity ($I_{sp} \sim 1000 - 10000$ s) they can produce,[1] while solar sails require no propellant at all, as the thrust force originates from the solar radiation pressure. For this reason, efficient low-thrust transfers are an attractive option for interplanetary missions, lunar missions and orbit raising maneuvers. NASA's Dawn spacecraft, which carried an ion propulsion system, has achieved the highest accumulation of ΔV of any spacecraft with nearly 11 km s^{-1}, allowing it to orbit both Vesta and Ceres in a single mission.[2] However, the continuous nature of the burn results in a complex design process to find optimal low-thrust transfers that satisfy all mission constraints. Traditionally, these problems are formulated as optimal control problems (OCP) and approached in a direct or indirect manner. The indirect method reformulates the problem as a two-point boundary value problem with the Pontryagin Maximum Principle (PMP),[3] based on the calculus of variations, and establishes the first-order optimality conditions that the solution must satisfy to be optimal. NASA developed several tools that use indirect approaches solved via single-shooting methods, which they collect as part of their Low-Thrust Trajectory Tool Suite,[4] and used these to support their Deep Space 1 mission.[5] Another example employs a particle swarm optimization to solve optimal interplanetary low-thrust transfer formulated as an indirect problem.[6] Alternatively, the direct method reformulates the low-thrust transfer as a nonlinear programming problem (NLP), which can be solved by gradient-based or heuristic methods that attempt to establish a set of control variables that directly optimize an objective function.[7] For instance, the MANTRA tool[8] developed by ESA uses a multiple-shooting technique to solve low-thrust transfers in a direct fashion and it has been used in designing the fly-by sequence with low-thrust arcs for the BepiColombo mission.[9] A survey on the tools in academia and industry for designing low-thrust

[*]Graduate Student, Faculty of Aerospace Engineering, Delft University of Technology, 2629 HS Delft, The Netherlands.
[†]Education fellow + Lecturer, Faculty of Aerospace Engineering, Delft University of Technology, 2629 HS Delft, The Netherlands.

transfers found that generally tools are complex and difficult to embed in a concurrent design phase as they lack the capability of including mission constraints, the capability of multi-objective optimization and general efficiency in searching wide design spaces.[5]

Recently, the Physics-Informed Neural Network (PINN) has been reintroduced.[10,11] These types of networks insert additional information about the physical process under consideration into the network in an attempt to better comply with physical models and partially lift the black box nature of the neural network. The original loss function of the network is combined with an additional term reflecting the implicit differential equations describing the governing physical model, thereby penalizing the loss if the physical model is being violated at certain collocation points. In supervised fashion, PINNs can be configured to perform a physics-aided regression on data or solve inverse problems. PINN-based representations exhibit greater confidence in generalizing beyond the observed domain and simultaneously filter errors in the data, thereby often surpassing the performance of standard neural networks.[12] Alternatively, in the absence of data, PINNs can be configured in an unsupervised manner working only with collocation inputs to solve partial differential equations (PDE) and ordinary differential equations (ODE). The advantages of the unsupervised PINN over traditional methods like finite element methods (FEM) and numerical integrators are that the solution is mesh-free, differentiable and does not suffer from discretization errors. Initial and boundary conditions can be embedded by additional loss terms or analytically through a constraint layer. Although the majority of research on PINNs is focusing on PDEs from a computational science perspective,[12,13] an increasing amount of research is being produced that seeks to apply these types of networks in applications regarding space flight dynamics. For example, a PINN in supervised form has been developed to learn the gravity fields of the Earth and Moon[14] as well as that of smaller bodies.[15] Very recently, supervised PINNs have also been constructed to solve orbit determination problems.[16] In unsupervised configuration, a PINN design has been proposed to solve optimal control problems by solving the two-boundary-value optimal control problem resulting from the indirect method and this has been applied on several low-thrust related optimal control problems.[17] Their method, which proves to be efficient, combines theory of functional connections with PINNs in a framework called X-TFC developed by the same authors.[18] Outside the context of astrodynamics, a PINN to solve optimal control problems for PDE has been developed, which includes the objective as an additional term to the loss function.[19] This can be considered as the PINN variant of applying the direct method to optimal control problems and this mechanism will be transferred in this work to the context of low-thrust interplanetary transfer trajectories. The goal of this research is to explore this approach as a method to improve preliminary optimization of low-thrust transfers in the context of a concurrent design phase.

This work starts by introducing a general form of the proposed PINN structure to solve continuous optimal control problems with the direct method as introduced by Mowlavi et al.[19] extended with analytical constraints inspired by Mattheakis et al.[20] Secondly, it will be demonstrated how the method can be explicitly implemented for interplanetary low-thrust transfers with an optimal fuel objective. Subsequently, a series of experiments are performed. First, the network's training procedure will be optimized. Secondly, a large number of Earth-Mars near-optimal transfers will be calculated and compared to those acquired by a shape-based method. Finally these results will be discussed in the greater context of tools in mission analysis.

PHYSICS-CONSTRAINED NEURAL NETWORK DESIGN

Consider a dynamical system that is to be transitioned from an initial state $\mathbf{z}_0$ to a final state $\mathbf{z}_f$ by means of a set of control parameters $\boldsymbol{u}$, while optimizing an objective J.

$$\text{Minimize} \quad J = \Phi(\mathbf{z}_0, \mathbf{z}_f, t_0, t_f) + \int_{t_0}^{t_f} L(\mathbf{z}(t), \mathbf{u}(t))\, dt \tag{1}$$

$$\text{Subject to} \quad \dot{\mathbf{z}} = f(\mathbf{z}, \mathbf{u}, t) \quad \mathbf{z} \in \mathbb{R}^N, \mathbf{u} \in \mathbb{R}^Q, t \in [t_0, t_f] \tag{2}$$

$$\mathbf{z}(t_0) = \mathbf{z}_0 \tag{3}$$

$$\mathbf{z}(t_f) = \mathbf{z}_f \tag{4}$$

A neural network $\mathcal{N}(t, \boldsymbol{p})$ is randomly initialized and designated as the solution, where $\mathbf{p}$ contains the weights and biases that make up the network. As such, it maps time t to the state vector $\mathbf{z}$ and control parameters $\mathbf{u}$. It does so by first mapping time t to a set of intermediate outcomes $\boldsymbol{\mathcal{N}}_z \in \mathbb{R}^N$ and $\boldsymbol{\mathcal{N}}_u \in \mathbb{R}^Q$. Subsequently, the intermediate outcomes are inserted in a set of constraint equations in order to analytically satisfy the boundary conditions on the state and any other constraints on the control parameters. This type of hard constraint classifies the network as a *physics-constrained neural network* (PCNN).[12]

$$\begin{bmatrix} \boldsymbol{\mathcal{N}}_z \\ \boldsymbol{\mathcal{N}}_u \end{bmatrix} = \mathcal{N}(t, \boldsymbol{p}) : \mathbb{R} \mapsto \mathbb{R}^{N+Q} \tag{5}$$

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} h(\boldsymbol{\mathcal{N}}_z,\, t) \\ g(\boldsymbol{\mathcal{N}}_u,\, t) \end{bmatrix} \tag{6}$$

Mattheakis et al. use a time-dependent parametric equation with exponential functions to satisfy an initial condition in their PCNN.[20] Their structure will be extended here to simultaneously enforce an initial and final condition on the state, see Equation 7.

$$\begin{aligned} \mathbf{z} = h(\boldsymbol{\mathcal{N}}_z,\, t) = \quad & e^{-a(t-t_0)}\, \mathbf{z}_0 \\ & + \left[1 - e^{-a(t-t_0)} - e^{a(t-t_f)}\right] \boldsymbol{\mathcal{N}}_z \\ & + \left[1 - e^{-a(t-t_f)}\right] \mathbf{z}_f \end{aligned} \tag{7}$$

where a is a parameter that controls the steepness with which the parametric equations vanish and rise. It is required that these functions are differentiable in the time domain of the problem. Figure 1 graphically depicts the various constraint terms in Equation 7, with $a = 10$. Note that not necessarily all state parameters have to be constrained at t_0 and t_f, specific entries can be left open at the boundaries, depending on the problem at hand.

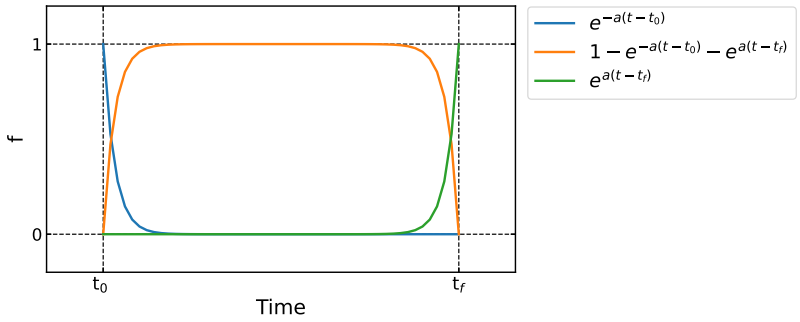


Figure 1. Graphical illustration of the terms in constraint Equation 7 with $a = 10$.

After initialization, the solution as described by the neural network is random and nonsensical. In order to solve the problem, the network has to be trained. The loss to be optimized during training consists of several terms. The first term, $\mathcal{L}_d$ (Equation 9), represents the dynamical constraints of the system and consists of N terms $\mathcal{L}_i$ each reflecting one of the N coupled ordinary differential equations $\dot{z}_i = f_i(\mathbf{x}, \mathbf{u}, t)$. The output $\mathbf{z}$ is differentiated with respect to the input t with automatic differentiation (AD) to get $\dot{\mathbf{z}}$ and this is compared to the right hand side of the equations of motion Eq. (2) for each entry i of the state vector separately. This is what makes the network *physics-informed*.

$$\mathcal{L}_i = \frac{1}{M}\sum_{j}^{M}\left[\dot{z}_i(t_j, \boldsymbol{p}) - f_i\big(\boldsymbol{z}(t_j, \boldsymbol{p}), \boldsymbol{u}(t_j, \boldsymbol{p}), t_j\big)\right]^2 \tag{8}$$

$$\mathcal{L}_d = \sum_{i}^{N}\mathcal{L}_i \tag{9}$$

where $j = 1, 2, ..., M$ are the collocation points in a batch. An additional loss term is added that represents the objective of the optimal control problem $\mathcal{L}_o$ (Equation 10), a procedure proposed by Mowlavi et al.[19] As a consequence, the fitness is directly optimized by altering a parameterized solution, the neural network, thereby classifying the approach as a direct method to solve the optimal control problem. The objective term is likely to be competing with the dynamics term and should be weighted with carefully selected weights: ω_d for the dynamics term and ω_o for the objective term. Note that it is possible to include more than one objective in the loss function.

$$\mathcal{L}_o = J = \Phi(\mathbf{z}_0, \mathbf{z}_f, t_0, t_f) + \int_{t_0}^{t_f} L(\mathbf{z}(t), \mathbf{u}(t))dt \tag{10}$$

$$\mathcal{L} = \omega_d\mathcal{L}_d + \omega_o\mathcal{L}_o \tag{11}$$

During training, the integral in J has to be numerically estimated from the set of training data. Training samples $t_0, t_1, ..., t_M$ are randomly chosen for each training epoch to mimic a continuous sampling space. In order to ensure that the integral in J can be fairly estimated, an equidistant grid of time points is used as the base sample, which is then perturbed randomly for each epoch.[20] The set of times sampled as a training batch is defined as T.

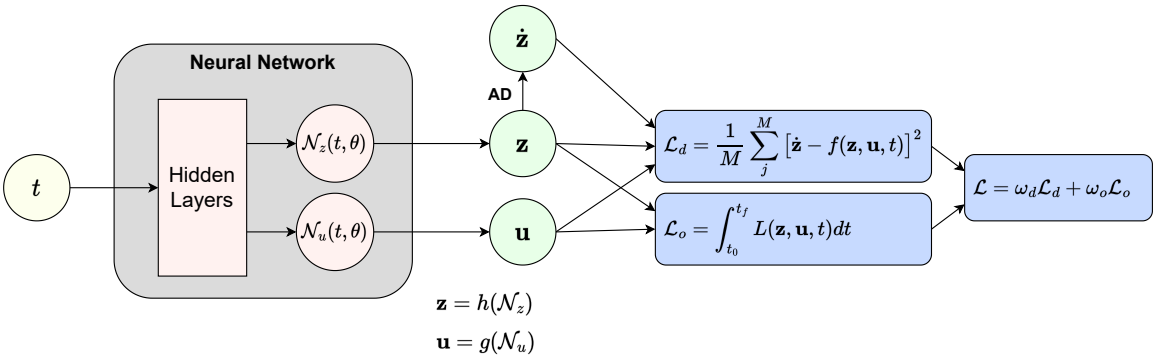


Figure 2. Overview of the Physics-Constrained Neural Network to solve two-boundary-value optimal control problems. AD refers to automatic differentiation.

$$T = \left\{\mathcal{G}(\mu = t_0 + n\Delta t, \sigma = 0.2\Delta t)\,|\,n \in \{0, 1, ..., M\}\right\} \tag{12}$$

$$\Delta t = \frac{t_f - t_0}{M} \tag{13}$$

where $\mathcal{G}(\mu, \sigma)$ is a Gaussian distribution centered around μ with standard deviation σ. As the loss $\mathcal{L}$ is being minimized by training the network using a gradient descent algorithm, the neural network takes on

the shape of a solution to the optimal control problem that consists of a continuous description of the state vector $\mathbf{z}(t)$ and control parameters $\mathbf{u}(t)$ that minimizes the objective J, satisfies the dynamical constraints and analytically adheres to the boundary conditions. An overview of the design with the corresponding loss functions is depicted in Figure 2.

PCNN FOR FUEL-OPTIMAL LOW-THRUST TRANSFER

The proposed PCNN design to solve continuous optimal control problems will be implemented for low-thrust transfer trajectories. As such, this section will first introduce the low-thrust transfer problem with accompanying frame of reference, coordinate system and equations of motion, followed by the explicit implementation into the PCNN method.

Optimal-Fuel Low-Thrust transfer problem

Consider the restricted two-body problem, where the mass of the orbiting body, a spacecraft, is negligible compared to the central body. The spacecraft is assigned a mass m and the central body a gravitational parameter $\mu = GM$. An electric propulsion system attached to the spacecraft is exerting a thrust force with a specific impulse I_{sp}. It can regulate the thrust force by adjusting the mass expulsion rate and it is free to point it in any direction in the plane. A polar reference frame with the central body at the origin is established where the position is described in terms of the distance from the origin to the spacecraft r and the polar angle θ. A basis is spanned with a radial unit vector $\hat{\mathbf{r}}$ pointing in the outward radial direction and a tangential unit vector $\hat{\boldsymbol{\theta}}$, perpendicular to the radial unit vector, pointing in the direction of increasing polar angle. The velocity is projected on the basis $\{\hat{\mathbf{r}}, \hat{\boldsymbol{\theta}}\}$, its entries are then $v_r = \mathbf{v} \cdot \hat{\mathbf{r}}$ and $v_\theta = \mathbf{v} \cdot \hat{\boldsymbol{\theta}}$. The thrust will be parameterized in terms of the thrust magnitude u_T and u_ϕ, which measures the angle between the tangential unit vector and the thrust vector. The state of the spacecraft is then written as $\boldsymbol{x} = \begin{bmatrix} r & \theta & v_r & v_\theta \end{bmatrix}^T$ and the thrust parameters as $\boldsymbol{u} = \begin{bmatrix} u_\phi & u_T \end{bmatrix}^T$. The equations of motion in these coordinates read[17]

$$\dot{r} = v_r \tag{14}$$

$$\dot{\theta} = \frac{v_\theta}{r} \tag{15}$$

$$\dot{v_r} = \frac{v_\theta^2}{r} - \frac{\mu}{r^2} + \frac{u_T \sin u_\phi}{m} \tag{16}$$

$$\dot{v_\theta} = -\frac{v_r v_\theta}{r} + \frac{u_T \cos u_\phi}{m} \tag{17}$$

$$\dot{m} = \frac{u_T}{I_{sp} g_0} \tag{18}$$

One objective that is minimized for transfer design is the propellant mass m_p. A way to formulate such a problem is by assuming a fixed initial mass $m(t_0) = m_0$ and subsequently minimizing the propellant use. The cost function, the propellant mass, can be calculated by integration over the thrust force.

$$J = m_p = \frac{1}{I_{sp} g_0} \int_{t_0}^{t_f} u_T(t)\, dt \tag{19}$$

An initial and final constraint is set on the state of the spacecraft: $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ and $\boldsymbol{x}(t_f) = \boldsymbol{x}_f$. Finally, the maximum thrust force is constrained to $u_T \leq u_{max}$.

Implementation of the PCNN method

A physics-constrained neural network as outlined in the previous section is constructed to solve the fuel-optimal low-thrust transfer trajectory. The neural network will map time $t \in \mathbb{R}$ to a set of intermediate

values of the state entries, mass entry, and control entries, $\mathcal{N}(t, \mathbf{p}) = \begin{bmatrix} \boldsymbol{\mathcal{N}_x} & \mathcal{N}_m & \boldsymbol{\mathcal{N}_u} \end{bmatrix}^T \in \mathbb{R}^7$, where $\boldsymbol{\mathcal{N}_x} = \begin{bmatrix} \mathcal{N}_r & \mathcal{N}_\theta & \mathcal{N}_{v_r} & \mathcal{N}_{v_\theta} \end{bmatrix}$ and $\boldsymbol{\mathcal{N}_u} = \begin{bmatrix} \mathcal{N}_{u_\phi} & \mathcal{N}_{u_T} \end{bmatrix}$. Subsequently, this output will be inserted in a set of constraint equations that map the networks output to the actual state $\mathbf{x}(t)$, mass $m(t)$ and control parameters $\mathbf{u}(t)$.

$$\begin{bmatrix} \boldsymbol{x} \\ m \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} h_x(\boldsymbol{\mathcal{N}_x}, t) \\ h_m(\mathcal{N}_m, t) \\ g(\boldsymbol{\mathcal{N}_u}) \end{bmatrix} \tag{20}$$

To adhere to the initial and final state constraint, the function h_x takes the shape of Equation 7, as proposed in the previous section. The amount of revolutions during the transfer can be controlled by increasing the final condition on the angle θ_f in steps of 2π for each additional revolution. The mass is constraint via

$$m = h_m(\mathcal{N}_m, t) = m_0 - (1 - e^{-a(t-t_0)})\ m_0\ \text{sigmoid}(\mathcal{N}_m) \tag{21}$$

This function makes sure that the spacecraft mass can only be in the range $[0, m_0]$ while also satisfying $m(t_0) = m_0$.

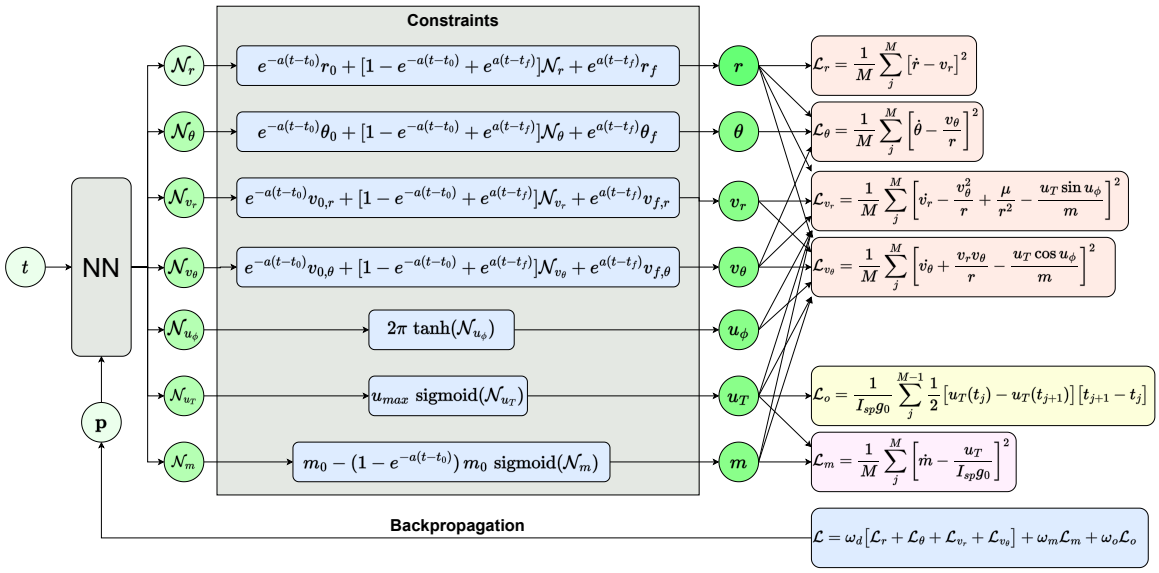


Figure 3. Overview of the Physics-Constrained Neural Network implemented for solving a planar optimal-fuel low-thrust transfer trajectory problem.

The control entries $\mathcal{N}_{u_\phi}$ and $\mathcal{N}_{u_T}$ are constrained via

$$\boldsymbol{u} = \begin{bmatrix} u_\phi \\ u_T \end{bmatrix} = g(\boldsymbol{\mathcal{N}}_\mathbf{u}) = \begin{bmatrix} 2\pi\ \tanh(\mathcal{N}_{u_\phi}) \\ u_{max}\ \text{sigmoid}(\mathcal{N}_{u_T}) \end{bmatrix} \tag{22}$$

The sigmoid activation of $\mathcal{N}_{u_T}$ followed by a multiplication with u_{max} makes sure that u_T is forced between $[0, u_{max}]$. Similarly, u_ϕ is forced in the range $[-2\pi, 2\pi]$. Although a range of $[-\pi, \pi]$ is sufficient to

be able to reach all possible control outputs, it is deliberately chosen to go for a larger range, because this provides the gradient descent optimization of the neural network more flexibility. If the value of u_ϕ would be near π somewhere during training and the gradient descent dictates that u_ϕ should increase, the larger range allows for it to rotate further instead of stalling out at π.

There are four dynamics loss terms, one for each of Equations 14-17 and a mass loss term, representing Equation 18 which is separately weighted. The objective, Equation 19, will be estimated by means of a trapezoid rule, using the set of inputs $\{t_1, ..., t_m\}$ of a batch, which are sampled as a perturbed equidistant grid (Equation 12), and the corresponding outputs of the thrust force magnitude $\{u_T(t_0), ..., u_T(t_M)\}$.

$$\mathcal{L}_o = \frac{1}{I_{sp} g_0} \sum_{j}^{M-1} \frac{1}{2} \big[u_T(t_j) - u_T(t_{j+1})\big] \big[t_{j+1} - t_j\big] \tag{23}$$

$$\mathcal{L} = \omega_d [\mathcal{L}_r + \mathcal{L}_\theta + \mathcal{L}_{v_r} + \mathcal{L}_{v_\theta}] + \omega_m \mathcal{L}_m + \omega_o \mathcal{L}_o \tag{24}$$

The entire network architecture with all constraints and loss terms is schematically presented in Figure 3.

Verification of solutions

Once a neural network has been trained, it provides $\mathbf{x}(\mathbf{t})$, $m(t)$ and $\mathbf{u}(t)$ that satisfy the equations of motion, if the dynamics loss terms are in a global minimum. In order to verify to what degree this is true and resulting trajectories are in fact physically valid ones, the initial position $\mathbf{x}_0$ together with the control profile $\mathbf{u}(t)$, as established by the network, will be numerically integrated by a Runge-Kutta 7(8) variable step-size integrator with relative and absolute tolerance set to 10^{-10}. The result is a discrete set of states $\{\mathbf{x}_v(t_k)\}_{k=0}^{k=M}$ that represent the real trajectory that the spacecraft would fly given the alleged optimal control profile. The verification state at the final epoch $\mathbf{x}_v(t_f)$ can be compared to the target state $\mathbf{x}_f$, where the network was analytically constrained to end. If these two coincide, the network has succeeded in creating a physically valid control profile and accompanying trajectory. Three metrics can be derived by comparing these final states: the final euclidean position distance dx (Equation 25), the final euclidean velocity distance dv (Equation 26) and difference in final mass dm (Equation 27).

$$dx = \sqrt{\big(x(t_f) - x_v(t_f)\big)^2 + \big(y(t_f) - y_v(t_f)\big)^2} \tag{25}$$

$$dv = \sqrt{\big(v_x(t_f) - v_{x,v}(t_f)\big)^2 + \big(v_y(t_f) - v_{y,v}(t_f)\big)^2} \tag{26}$$

$$dm = m(t_f) - m_v(t_f) \tag{27}$$

EXPERIMENTS

A baseline network architecture and training scheme will be postulated. Although many aspects of the configuration can be researched and optimized, this work will only focus on the training procedure and the selection of the loss weights. It was found early on that the training procedure, specifically the learning rate schedule, has a large influence on successful convergence and it is thus appropriate to elaborate on the choices on this matter. Secondly, the loss weights are a set of parameters that have to be established on a problem-to-problem basis, as these dictate the balance between converging to a physical solution (adhering to the dynamics) and minimizing the objective. Finally, the baseline configuration will be applied to a substantial number of Earth-Mars transfers to thoroughly evaluate the method across a wide spectrum of boundary values, some of which reflect quite nonlinear situations. This assessment will involve a comparative analysis with results obtained from a hodographic shaping method.

Setting	Baseline Configuration
Neurons	10
Hidden Layers	4
Activation	sin
Training points M	200
Learning rate Schedule	1
$[\omega_d \quad \omega_m \quad \omega_o]$	$\begin{bmatrix}1 & 10^{-5} & 10^{-7}\end{bmatrix}$
a	10

Table 1. Baseline configuration of the network. The different learning rate schedules are listed in Table 2.

The baseline configuration

The baseline configuration consists of the settings listed in Table 1. Instead of mapping time to the intermediate outputs in a fully-connected neural network, a bundle of networks is used to map time to each output separately. In the case of the low-thrust transfer this results in seven parallel networks that map $\mathbb{R} \rightarrow \mathbb{R}$. All networks are trained with the Adam optimizer.[21] Every 1000 epochs, a test evaluation of the network is carried out. This comprises the evaluation of 200 training points in an unperturbed equidistant time grid and the outcome is used to evaluate the metrics dx, dv and dm. The state of the network at the epoch with minimum total test loss is considered the solution. Note that this is not necessarily the final epoch. All experiments are implemented with the DeepXDE[22] python module with TensorFlow[23] as the backend. Numerical integrations of control profiles for verification were executed in the TU Delft Astrodynamics Toolkit (TUDAT).[24] The training of neural networks in large quantities has been performed in a paralleled configuration on CPU nodes* of the DelftBLue supercomputer.[25]

The test case: Earth-Mars transfer & rendezvous

The problem to be tackled by the baseline network to optimize the training procedure and loss weights settings will be that of a planar Earth-Mars transfer and rendezvous where both planets are assumed to be in circular orbits. The initial spacecraft mass is $m_0 = 100$ kg, the maximum thrust force is $u_{max} = 0.1$ N and the specific impulse of the electric propulsion system is $I_{sp} = 2500$ s. Units are dimensionless: the unit of distance is scaled to 1 AU and the unit of time is scaled with $\frac{\text{1AU}}{v_\oplus}$, where $v_\oplus$ is the circular velocity of the Earth. As a consequence the velocity is scaled by $v_\oplus$ and the orbital period of Earth corresponds to 2π of time units. Initially the spacecraft flies in Earths orbit $\mathbf{x}_0 = \begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}$. After $t_f = 17.21$, which corresponds to 1000 days, the spacecraft is constraint to fly in Mars's orbit where it is assumed Mars is located precisely two revolutions from the spacecraft's initial position. The final state is thus $\mathbf{x}_f = \begin{bmatrix}1.5 & 4\pi & 0 & 0.816\end{bmatrix}$. An example solution of this scenario as acquired by the baseline configuration is presented in Figure 4 and the corresponding loss evolution in Figure 5. For this specific solution, the verification metrics are $dx = 0.073$ AU, $dv = 0.040$ $\text{v}_\oplus$ and $dm = -0.12$ kg. The propellant mass, which is minimized, converged to $m_p = 19.84$ kg. If this transfer would have been performed as a Hohmann transfer, which burns the theoretical minimum propellant mass for this scenario, it would take 19.78 kg of propellant with the engines I_{sp}. The small difference between the low-thrust transfer and the Hohmann transfer demonstrates that the solution is indeed approaching a global minimum. However, the spacecraft misses its target by roughly 0.07 AU and closing this gap requires additional propellant. In the section on the selection of the weight parameter ω_o, it will be quantified how much this is.

The training procedure

The proposed optimal control problem and the numerous weights and biases that make up the neural network span an intricate loss landscape. If one omits the objective term, there are many global minima, because

*Equipped with Intel XEON Gold E5-6226R processors

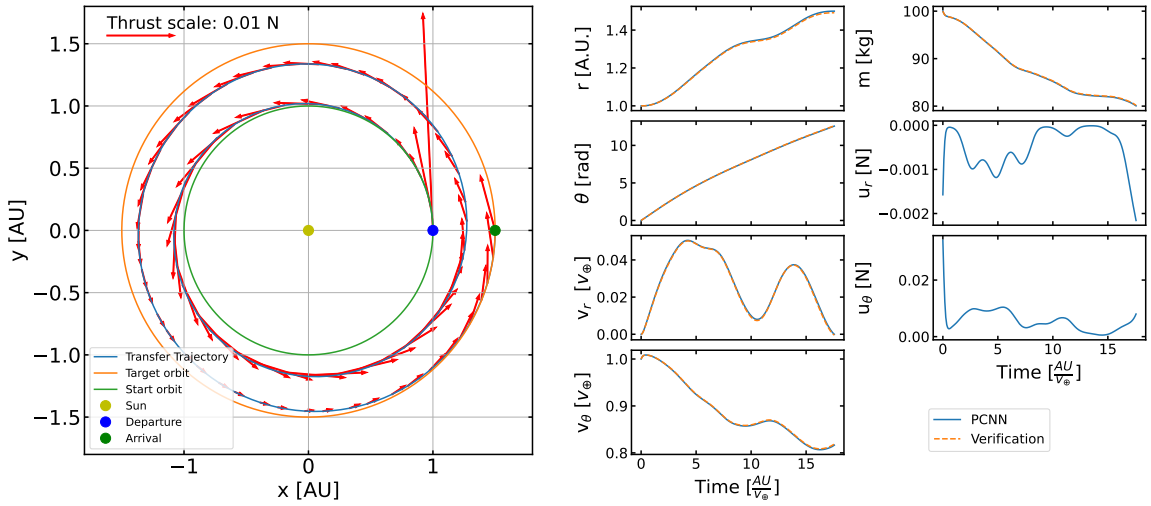


Figure 4. A near optimal-fuel solution for an Earth-Mars low-thrust transfer as acquired by the baseline configuration of the physics-constrained neural network. The verification metrics are $dx = 0.073$ AU, $dv = 0.040$ $\mathbf{v}_\oplus$ and $dm = -0.12$ kg. The propellant mass is $m_p = 19.84$ kg.

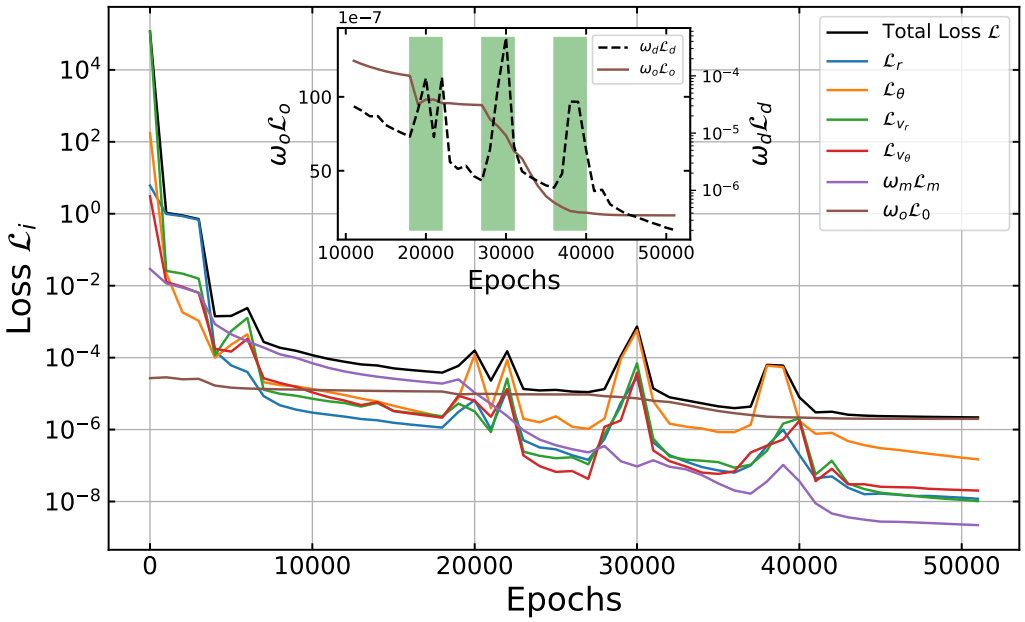


Figure 5. Training loss evolution for the solution as presented in Figure 4. The interior panel displays the objective loss (brown) on a linear scale and the dynamics loss (dashed black) on a log scale from epoch 10000 onwards where the green shaded regions indicate a shaking phase.

	Schedule [learning rate, epochs]	Description
Schedule 1	[10^{-2}, 3k]* $\rightarrow$ [10^{-3}, 5k] $\rightarrow$ [10^{-4}, 10k] $\rightarrow$ [5×10^{-3}, 4k] $\rightarrow$ [10^{-4}, 5k] $\rightarrow$ [5×10^{-3}, 4k] $\rightarrow$ [10^{-4}, 5k] $\rightarrow$ [5×10^{-3}, 4k] $\rightarrow$ [10^{-4}, 5k] $\rightarrow$ [10^{-5}, 6k]	Restarting + Scheduling + Shaking
Schedule 2	[10^{-2}, 3k] $\rightarrow$ [10^{-3}, 5k] $\rightarrow$ [10^{-4}, 10k] $\rightarrow$ [5×10^{-3}, 4k] $\rightarrow$ [10^{-4}, 5k] $\rightarrow$ [5×10^{-3}, 4k] $\rightarrow$ [10^{-4}, 5k] $\rightarrow$ [5×10^{-3}, 4k] $\rightarrow$ [10^{-4}, 5k] $\rightarrow$ [10^{-5}, 6k]	Scheduling + Shaking
Schedule 3	[10^{-2}, 3k]* $\rightarrow$ [10^{-3}, 5k] $\rightarrow$ [10^{-4}, 20k] $\rightarrow$ [10^{-5}, 23k]	Restarting + Scheduling
Schedule 4	[10^{-4}, 51k]	-

Table 2. Different schedules for the learning parameter. A star* indicates that the training will only proceed to the next phase if the loss is below a certain threshold, otherwise the network is reinitialized and the training starts over.

there are various physically valid trajectories with accompanying control profile that brings the spacecraft from $\mathbf{x}_0$ to $\mathbf{x}_f$. In order to demonstrate how to construct a training procedure that can navigate the capricious loss space, four candidate training schemes are proposed in Table 2. Although they all make use of the Adam optimizer, they differ in their learning rate schedule. Three building blocks in establishing a schedule are suggested. First of all the concept of *scheduling* refers to the gradual decrease of learning rate after a predefined amount of epochs. This is widely applied in neural network training, because in certain loss spaces, it might be necessary to take smaller steps in order to find a narrow minimum. Secondly, the concept of *restarting* is introduced: if the total loss of the network is not smaller than a certain threshold, the network will be re-initialized and the training will start over. This should ideally be invoked early in the training to avoid excessive run times. Finally, the concept of *shaking* is introduced, which is defined as the temporarily increasing of the learning rate in order to induce large steps that can potentially escape local minima. Applying all three building blocks, *scheduling, restarting and shaking*, is embodied in Schedule 1. The restarting phase has as criteria that the total test loss should be smaller than 5 after 3000 epochs. Alternative learning schedules for comparison, leaving one or more of the building blocks out, are Schedules 2 to 4.

The problem has been solved by each training scheme 40 times and the distribution of metrics are gathered in Figure 6. An analysis of these results will follow here. Training Schedule 1 performs best, which scores the best median values for all metrics and simultaneously has the smallest spread in outcomes. From the big spread of dx, dv and dm for Schedule 2, it has become evident that a portion of the network initialization will result in a non-physical solution. On top of that, 16 out of the 40 cases of training Schedule 2 have crashed resulting in NaN values. Restarting the network early once an insufficiently small loss is achieved (or NaN value encountered), like with Schedule 1, has resulted in a significantly more predictable outcome, as demonstrated by the small spread in metrics for Schedule 1 and no instances of NaN values. Apparently it can be fairly confidently recognized early on in training whether it will fail to converge to a physical solution or not. To visualize this effect more closely and justify the selection of the restarting criteria, see Figure 7. The loss evolution is plotted for all cases trained with Schedule 2 in the left panel. Clearly a distinction into two groups is visible, a set of cases converges to a value close to $\mathcal{L} = 10^{-5}$ while another set does not converges and does not get better than $\mathcal{L} = 10$. Plotting the loss at epoch 3000 vs the dx of the trained solution (right panel) demonstrates that it can be predicted in what group the solution will fall. Selecting only cases with a loss smaller than 5 at epoch 3000 therefore filters failed initialization. It is uncertain why only a subset of initialization is capable of converging to a solution. It would be highly appreciated to have a more theoretical understanding of what drives this distinction. Two outliers are present in the sample trained with Schedule 1, both of these solutions have $dx > 2$ AU and $dv > 1$ $\mathrm{v}_\oplus$ and by manual inspection it was found that the training loss was diverging with multiple orders of magnitude from the test loss after epoch 3000. A temporary solution might be to monitor the difference between train and test loss and implement a second restart if a significant discrepancy between the two is detected.

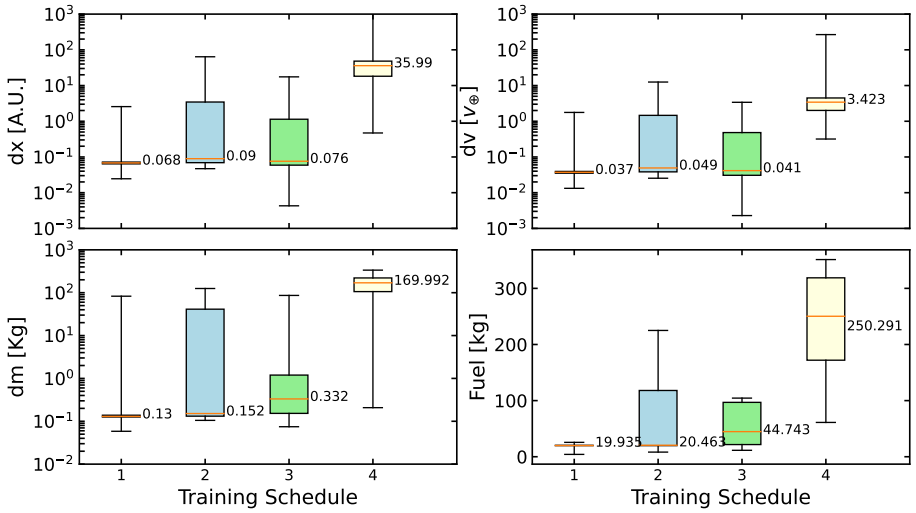


Figure 6. Comparison of learning training schedules as defined in Table 2. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

Schedule 3 restarts just like Schedule 1 but lacks shakes at the end of the training. This has resulted in a somewhat less predictable final accuracy for position, velocity and mass and significantly less optimal solution, as indicated by the larger median objective value. The effect of shakes is best understood by inspection of Figure 5. In the interior panel, the shaded regions indicate the shaking phase with increased learning rate. At the start of these phases, the objective loss suddenly starts to decrease and at the same time the dynamics loss terms sharply increase. If the learning rate is then lowered, the dynamics terms settle back to their original state, or better, but the objective loss term stays low, or keeps decreasing. For the example case, it requires two such cycles to permanently place the objective loss in a downward trend towards the global minimum. From manual inspection it was found that most cases require 0 or 1 such cycle.

The following hypothesis for the success in jumping towards a more global minimum by performing shakes is proposed. Before the shake, the network had found itself in a local minimum. That local minima approaches a physically valid solution that has not optimized the objective. It also hasn't quite found the minimum for the dynamics, because it has stalled into a battle with the objective term. Once the learning rate increases, the steps taken by the gradient descent algorithm are larger. The dominating loss term will dictate in what direction to move with the larger steps. In this case, that term is the objective and indeed this starts to decrease which means that underneath the hood the thrust magnitude values are decreasing. All competing terms will then be increasing. Those are the state vector related terms, which no longer make sense with the smaller thrust magnitudes. As the objective term is decreasing more and more with the large gradient descent steps, the domination of the loss terms is flipped and the opposite happens: the state estimations start to adapt to the new smaller thrust values. It does so not by increasing the thrust magnitudes, but by adapting the states, as indicated by the objective loss, which is not increasing again after a shake but remains the same or keeps on decreasing. The network slides into an alternative dynamically valid minimum, which is closer to the optimal objective. Repeating this cycle 'shakes' the network from one dynamically valid option to the next until it ultimately finds the option that also minimizes the objective. The superior median metric values in Figure 6 of Schedule 1 compared to Schedule 2 indicate that shaking is successful not merely in the example of Figure 5 but for the vast majority of the cases. Note that this effect can only be successfully applied if the loss weights allow for an alternating domination in the loss terms, once the dynamics loss terms have nearly converged to a minimum (a physically valid solution). If the objective weight is too large, the network is shaken into a physically invalid trajectory. Also, realize that this objective only depends on the thrust magnitude and is therefore quite simple to optimize. More complicated objective expressions might

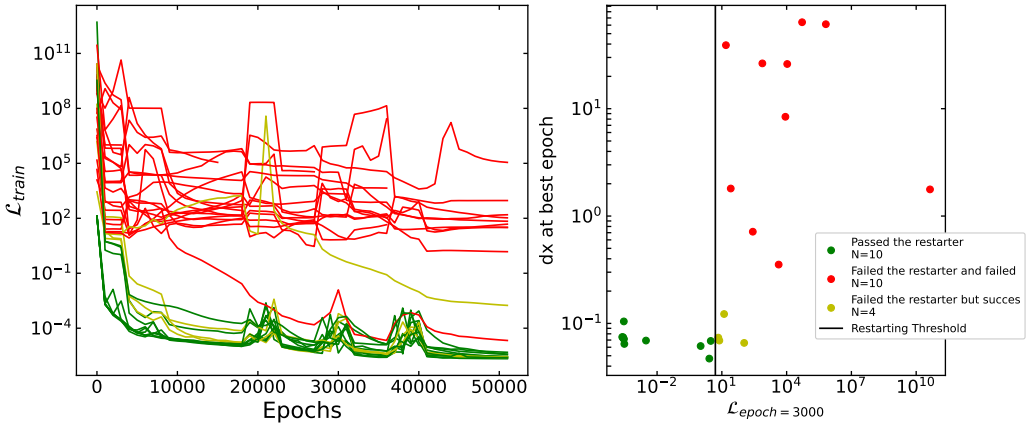


Figure 7. (*Left*) The training loss evolution of the cases trained with Schedule 2, which does not restart. (*Right*) The total test loss at epoch 3000 versus the dx of the solution after training ended. The color code in the legend applies to both panels

certainly have a more elaborate reaction to shakes potentially rendering them useless.

In conclusion, the loss landscape is quite unique and capricious, but with some unusual tools like restarting and shaking, it can be successfully navigated.

The loss weights

The total loss in Equation 11 consists of a dynamics term $\mathcal{L}_d$, a mass term $\mathcal{L}_m$ and an objective term $\mathcal{L}_o$ each multiplied with a weight. The dynamics weight is chosen $\omega_d = 1$ and the mass weight $\omega_m = 10^{-5}$. Because the dynamics and the objectives are competing, the relation between the dynamics weight and the objective weight is critical. The mass loss term is more decoupled from the others and therefore the training is not sensitive to its weight setting. Nevertheless, in order to ensure that it is not dominating in the training process, it is reduced by a weight of 10^{-5}. A grid search over the objective weight will determine the optimal value. Five options are considered 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6} and 10^{-5}.

The baseline network configuration is trained for each weight 40 times and the resulting metrics and objective is visualized in a box plot in Figure 8. As the objective weight increases, the median values of the dynamics metrics dx and dv increase as well. On the contrary, the median values of the objective are decreasing. This happens because the dominating loss term dictates where the biggest effect will be of the gradient descent steps. A dominating objective term means that the thrust magnitude values u_T will be reduced, which decreases the propellant mass. For large objective weights, the objective term will be dominating and thus the objective values will be smaller until they are zero, as is observed in the lower right panel of Figure 8. For smaller values of the objective weight, the dynamics loss terms are dominating resulting in physically correct descriptions of trajectories that are not per se optimal. The superior physical correctness is clearly observed for smaller objective weights in the upper row of plots in Figure 8.

An unusual situation arises here where optimality can be traded for physical accuracy of the solution. In order to choose the most appropriate weight, an estimation will be made of how much additional fuel it costs to close the gap in dx and dv for each weight. To achieve this a procedure to refine the solution will be applied. Consider the initial state and the control profile as provided by the neural network as a given. An additional empirical acceleration $\mathbf{a}$ in the RSW frame is defined that continuously acts on the spacecraft over the full course of the transfer. It will be the goal to choose the empirical acceleration vector such that it minimizes both dx and dv. A linearized sensitivity matrix $\mathbf{S}(t)$ relates the empirical acceleration to a change in linearized final position and velocity $\Delta\tilde{\mathbf{x}}$.

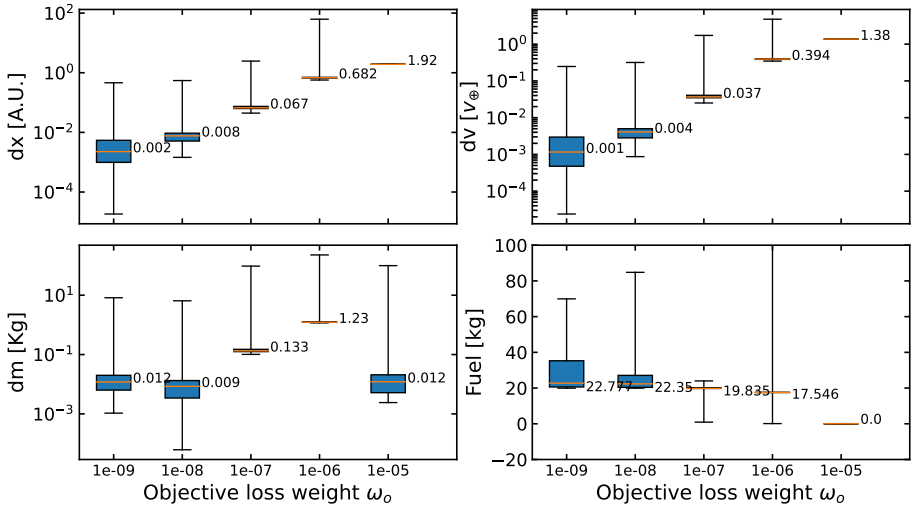


Figure 8. Grid search over the objective weight ω_o. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

$$S(t) = \frac{\delta \mathbf{x}(t)}{\delta \mathbf{a}} \tag{28}$$

$$\Delta \tilde{\mathbf{x}}(t) = \mathbf{S}(t) \Delta \mathbf{a} \tag{29}$$

The sensitivity matrix and state can be numerically integrated to find $\mathbf{S}(t_f)$ and $\mathbf{x}_v(t_f)$. The gap to close is then defined as $\Delta \tilde{\mathbf{x}} = \mathbf{x}_v(t_f) - \mathbf{x}_f$ where $\mathbf{x}_f$ is the original target state where the neural network was told to end. Inverting the sensitivity matrix allows to solve for the empirical accelerations that closes this gap.

$$\Delta \mathbf{a} = \mathbf{S}^{-1} \Delta \tilde{\mathbf{x}}(t_f) \tag{30}$$

Due to the linearization, multiple iterations are required to update the empirical acceleration until the final position and velocity is minimized. The resulting acceleration can be converted to a ΔV and then to a propellant mass via

$$\Delta V = ||\mathbf{a}||(t_f - t_0) \tag{31}$$

$$\tilde{m}_p = m_0 \left[1 - \exp \frac{\Delta V}{I_{sp} g_0} \right] \tag{32}$$

Note that this calculation assumes that only the correcting empirical acceleration is acting, while in reality the given control profile from the neural network solution is also acting on the vehicle. For that reason, the real empirical acceleration requires less propellant for the correction, because the burning happens at a smaller total vehicle mass. As a consequence $\tilde{m}_p$ is considered an upper limit of the propellant necessary to accommodate the empirical acceleration.

All individual cases in Figure 8 that have a $dx < 1$ AU will be corrected with the above-mentioned method by performing 8 iterations, in order to establish the most appropriate objective weight. Objective weight $\omega_0 = 10^{-5}$ is not considered as an option, because the propellant mass is consistently zero, implicating that

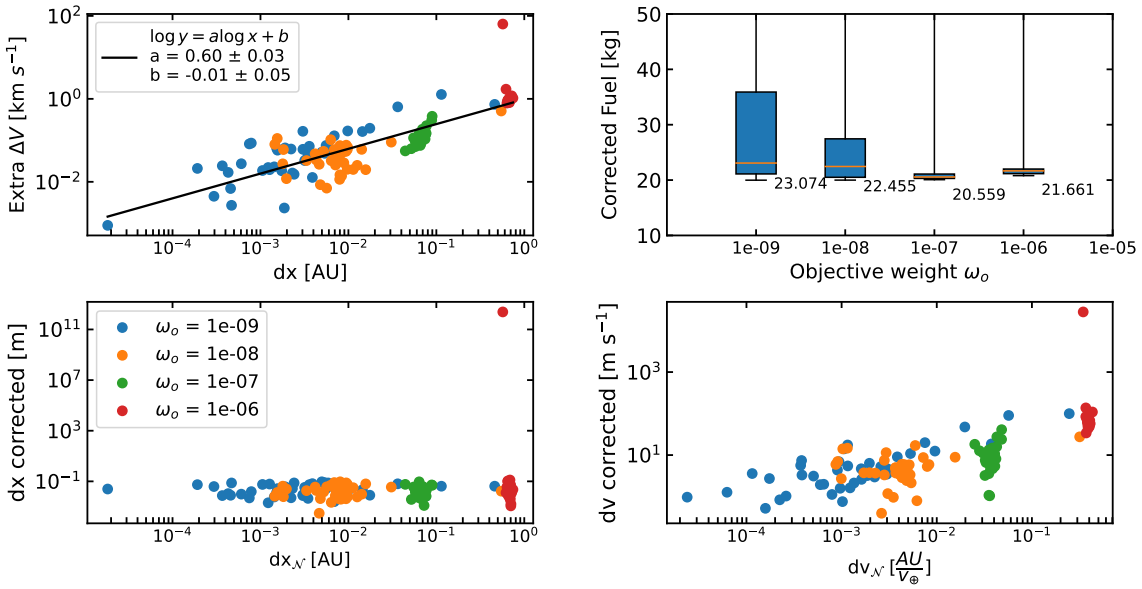


Figure 9. Refinement of dx and dv for the cases in the weight comparison (Figure 8) except for those with $\omega_o = 10^{-5}$. *Upper left*: the relation between the neural networks mismatch in position and the ΔV required to correct for it. *Upper right*: the distributions of total mass after including the propellant required for correction for each weight. *Lower left*: the mismatches in position after correction. *Lower right*: the mismatches in velocity after correction.

no change has occurred with respect to the initial orbit: the spacecraft is still flying in Earth's orbit and thus correcting it makes no sense. Integration of the sensitivity matrix and state is performed by a Runge-Kutta 4 integrator with constant step size of 3600 s, which has been benchmarked to a numerical integration error on the order of 1 m for the integration time considered here. The resulting ΔV required to accommodate the empirical acceleration is plotted as a function of the network's displacement dx in the upper left panel of Figure 9. A linear trend is observed in log-log space, indicating that a correlation exists between the final position error dx and the fuel required to correct for it. The lower panels in Figure 9 visualize the error after correction $\Delta\tilde{\mathbf{x}}$ as function of the initial error of the networks solution in position dx (*left*) and velocity dv (*right*). All but one transfer has been successfully refined to coincide with the final state $\mathbf{x}_f$ by less than a meter, which is considered a satisfying correction. The upper right panel in Figure 9 demonstrates the distribution of total propellant mass for the different weights where the total propellant mass is defined as the propellant mass from the neural network solution added to the additional propellant mass from the correction. The most balanced weight is $\omega_o = 10^{-7}$ which yields the smallest median total propellant and therefore this is selected as the optimal objective weight for this problem. As a consequence, the errors on the state entries are determined by the selection of this weight and not due to poor expressivity or training of the neural network.

Earth-Mars Transfer & rendezvous Opportunities

The test case as solved before has a time of flight in combination with relative phase angle that is convenient for an Earth-Mars transfer and rendezvous. With little effort by the electric propulsion, the spacecraft can be placed on an outward spiraling trajectory that meets with Mars and matches its velocity, as seen by the intuitive transfer trajectory in Figure 4. In order to test the PCNN method on a set of more challenging boundary conditions, the same problem will be solved for a large number of combinations of departure dates and times of flight. It's crucial to clarify that this pursuit does not aim to evaluate the PCNN method for

computationally efficient preliminary exploration of extensive departure and time of flight parameters. Its purpose is strictly to rigorously assess the capabilities of the PCNN method on a wide range of constraints. If Earth and Mars are positioned awkwardly for a given time of flight, it will take a more non-straightforward trajectory to reach it and calculating various such cases serves to gain some insight in the flexibility of the PCNN method.

In order to validate the optimality of the solutions a comparison will be made to near-optimal solutions acquired by a shape-based optimization method. Hodographic shaping (HS) methods can shape orbits by constructing functions that represent the time evolution of the velocity entries in the radial $\hat{\mathbf{r}}$ and tangential $\hat{\boldsymbol{\theta}}$ direction. These velocity functions consist of the sum of base functions, which can for example be an exponential, sine, cosine, power or any other functions that are analytically integrable. A linear combination of these base functions with a set of coefficients represent the shape. Initial and final constraints on position and velocity can be achieved through analytical computations of the coefficients. The thrust acceleration can then be acquired by differentiating the expressions for the velocities and subtracting the accelerations due to the environment. By allowing more base functions and corresponding coefficients than required to satisfy the boundary conditions, shapes can be optimized by selecting the free coefficients that minimize the objective. Hodographic shaping has been highly successful in efficiently providing preliminary near-optimal low-thrust transfer trajectories.[26] The PCNN method shares the underlying mechanism of modeling the solution with a function involving a free part. However, the highly over-parameterized nature of the neural network allows much more flexibility in the shape potentially providing superior optimality. This comes at the expense of requiring an elaborate physics-informed training procedure which yields orders of magnitude larger CPU times compared to hodographic shaping optimization.

A set of 52 equidistant dates between 1 January 2024 and 19 February 2026 will be used as departure dates. This corresponds to one synodic period of the Earth-Mars system with 2 departure dates per month. For each of those dates, a set of 25 times of flight options will be considered ranging from 100 days to 1060 days. The states of Earth and Mars at these epochs are extracted from the Spice kernel[27] in the ECLIPJ2000 reference frame. Positions and velocities are projected on the ecliptic by setting the z-components to 0. The objective value of an optimal solution will be converted to ΔV via the rocket equation. In order to mitigate some of the outliers detected during grid searches, a network is reinitialized if the test and train loss have diverged significantly at epoch 8000. Each case is solved with 0, 1 and 2 revolutions and the winning revolution is selected by choosing the case with the smallest ΔV that has a dx and dv smaller than 0.1. If there is no option that adheres to this criteria the launch opportunity will not be included. An additional ΔV to close the gap in dx has been added to the solution by making use of the relation between dx and ΔV as established by the linear fit in Figure 9. The same situations will be optimized by a hodographic shaping method using CPowPow2PSin05PCos05-CPowPow2PSin05PCos05 as base functions* for the radial and tangential velocities, which yield two free coefficients per velocity entry, as advised by the inventor of the method.[26] Optimization of the degrees of freedom in order to minimize the ΔV is carried out by the Nelder-Mead optimization scheme[28] implemented in the Pygmo module[29] in combination with a hodographic shaping tool implemented by Stubbig et al.[30] All launch opportunities are solved for 0, 1, 2 revolutions and the optimal amount is chosen. Options filtered from the PCNN solutions are also filtered from the HS solutions.

The optimal ΔV gathered from both methods are depicted in the launch opportunity plots in Figure 10 where they are plotted as a function of departure and arrival date. For the PCNN method, the position and velocity error metrics are plotted in Figure 11. From the successful adherence to the physical model for nearly all launch opportunities, as indicated by values of dx and dv smaller than 0.1 AU and $0.1\, v_{\oplus}$, it is concluded that the PCNN method in the baseline configuration is capable of providing solutions for a great range of constraints. In cases of brief transfer periods, the obtained solutions consistently fall short of reaching Mars within a 0.1 astronomical unit (AU) range. This limitation stems from the spacecraft's limited thrust, which proves insufficient for executing the transfer. Inspection of the launch opportunity plots reveal familiar patterns that are also found with the hodographic shaping method. For example, the launch window between approximately 200 to 300 days after 1 January 2024 is a good opportunity to have efficient short-duration transfers. Furthermore, as the time of flight increases, the ΔV becomes less sensitive to the departure date,

*notation adapted from Gondelach et al.[26]

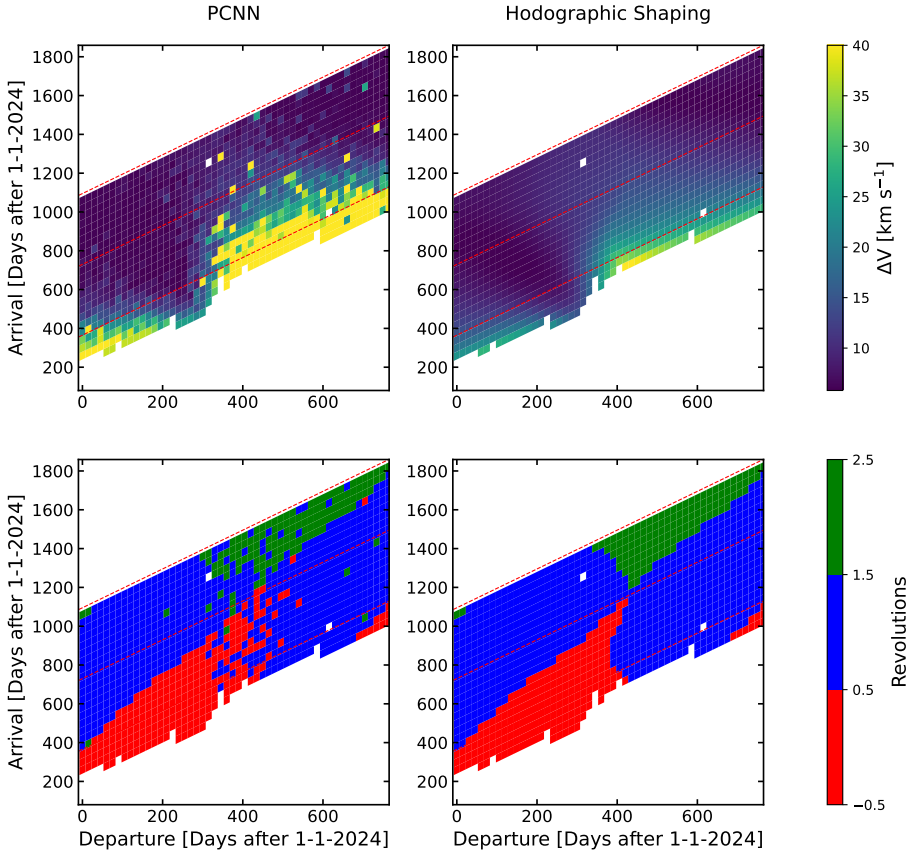


Figure 10. ***Upper row*: ΔV as a function of departure and arrival date for optimal-fuel Earth-Mars rendezvous transfers. *Lower row*: selected amount of revolutions. *Left column* is acquired with the PCNN proposed in this work and the *right column* is found by a hodographic shaping method.**

as it is most of the time capable of finding some amount of revolutions that can bring it there efficiently. This effect is well-known for low-thrust transfers because of the availability to loiter in a constant orbit awaiting a more efficient transfer opportunity. However, at the same time, the accuracy of the solution in terms of dx and dv decreases for larger times of flight and it requires more restarts during training to pass the threshold, as explicitly depicted in Figure 12. It is likely that to have an increased accuracy for larger times of flight, additional training epochs or larger network size is required. Finally, it should also be noted that there are misplaced high values among the solutions in the launch opportunity plot. These outliers tend to align with a deviating amount of revolutions compared to neighboring opportunities, indicating that the preferred amount of revolutions has failed resulting in the selection of a less optimal amount of revolutions. No structural reasons behind these failures have been recognized.

In Figure 13, the residuals in ΔV objectives between the methods are quantified for each launch opportunity. A striking observation is that the HS method outperforms the PCNN method in areas where both methods agree that the ΔV required is high. This can be attributed to the fact that the shape method doesn't have a limit on the thrust acceleration while the PCNN is constraint to $u_{max} = 0.1$ N. While the shape based method purely constructs a velocity profile and than retrieves the thrust acceleration to accommodate it, the PCNN method contains practical constraints on mass, thrust ans I_{sp}. Having the ability to produce more thrust allows it to generate more optimal solutions due to the shorter burns that are less sensitive to gravity losses. This is specifically important for short duration transfers that require high amounts of thrust early on, which explains the superior solutions of the HS. The right panel of Figure 13 shows only cases where the HS

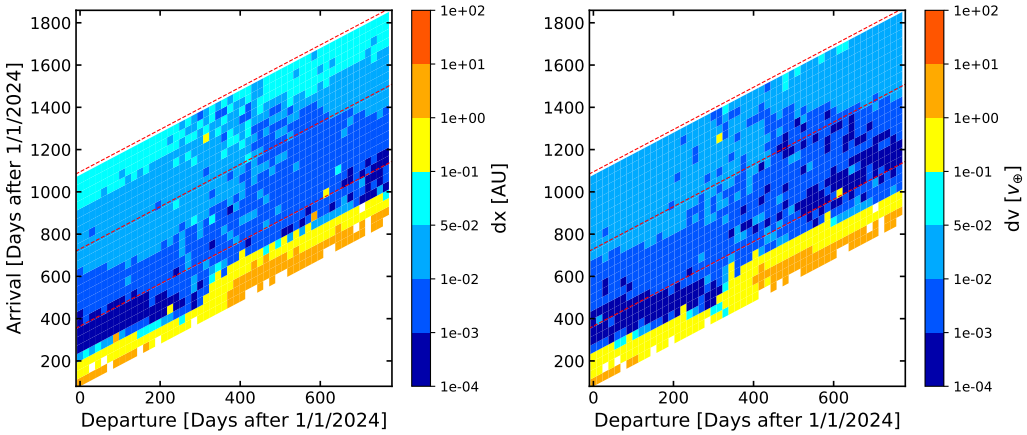


Figure 11. The verification metrics dx and dv of the PCNN solution as a function of departure and arrival date for optimal-fuel Earth-Mars rendezvous transfers.

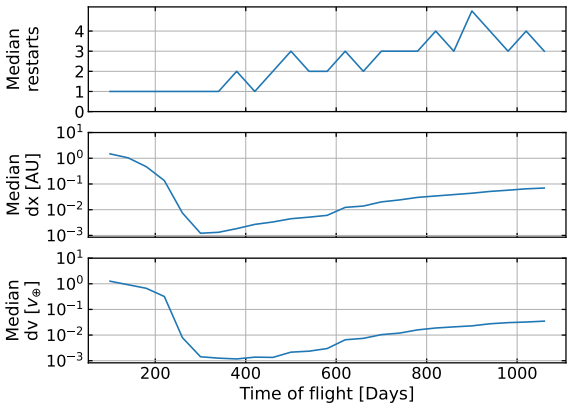


Figure 12. The influence of the time of flight on the accuracy of the solution and the required restarts for the cases in the Earth-Mars launch opportunity plot Figure 10.

solution complies with $u_{max} < 0.1$ N to allow an equal comparison. In this plot, it can be observed, with the exception of some outliers, that the HS method moderately outperforms the PCNN only in the regions where ΔV is generally very optimal. These are the light red shaded regions in the upper right and lower left. Those opportunities are representing phase angles in combination with times of flight that are quite convenient as the spacecraft can achieve the transfer by monotonically progressing radially outward to meet with Mars. These convenient boundaries make the optimal control problem more linear. In other areas the PCNN method outperforms the HS method with ΔV improvements of up to 4.5 km s^{-1}. The boundary conditions for those scenarios pose a more nonlinear optimal control problem due to the awkward time of flight with respect to the phase angle. The superior performance of the PCNN in specifically those areas suggests that the method excels in searching solutions to nonlinear problems. All cases together in the right panel of Figure 13 have a median residual of -0.55 km s^{-1}, rendering the PCNN method the preferred option for optimality. On the other hand, the average CPU training time per PCNN case in the launch opportunity plot is 770 seconds compared to something on the order of seconds for the HS method. Nevertheless it should be remembered that the PCNN has absolutely no prior knowledge and it is capable of taking on any shape. Shape-based methods employ simple functions that are analytically and highly efficiently adapted to adhere to constraints but also provide less versatility in generating solutions when compared to a PCNN based method, as indicated by the improved optimality of solutions generated by the PCNN method for challenging boundary conditions.

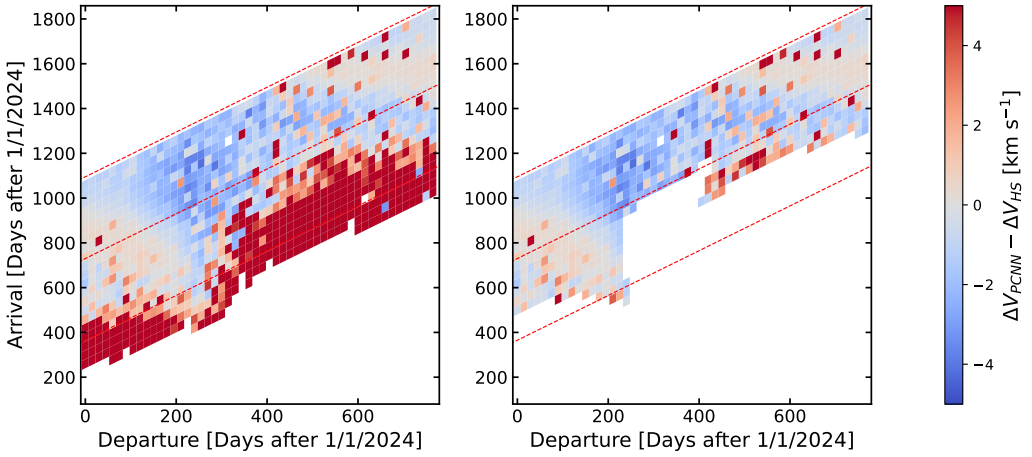


Figure 13. ΔV residuals between the PCNN method and a HS method for optimal Earth-Mars transfer and rendezvous opportunities. Right plot excludes cases where the HS method provided solutions with $u_{max} > 0.1$ N, which was a constraint for the PCNN method.

DISCUSSION ON POTENTIAL FOR MISSION ANALYSIS

In this section the prospects for the use of this work as a tool for mission analysis engineers will be discussed. Physics-Constrained Neural networks solving optimal control problems with the direct method provide some unique features. Although the underlying physics is encoded in the loss functions, the networks themselves contain no prior information about the problem to solve and thus explore a massive design space in shaping the trajectories that comply with the physical model. In finding the optimum, physical validness is traded for optimality in the training process, a peculiar situation. Finding a solution therefore requires navigating an obscure loss space, which is found to be challenging, but not impossible with intricate learning rate schedules. It has been demonstrated that a network configuration and training process can be established that can find near-optimal solutions for many constraints of the planar Earth-Mars transfer and rendezvous problem. Additionally, in many nonlinear circumstances these solutions provide similar or improved optimality compared to a shape-based optimization method. This is not a surprise as it is widely acknowledged that deep neural networks are proficient in capturing nonlinear relationships. Therefore, this method might be very suitable in the context of highly nonlinear dynamics, like the circular-restricted three-body problem, where the PCNNs might be able to excel in providing preliminary concepts of mission designs. A challenge in extending the model to different scenarios arises from the need to reevaluate loss weight parameters when encountering new objectives, environmental parameters or system characteristics, like the initial spacecraft mass. This presents a hurdle for seamless generalization across various scenarios. Furthermore, the method can naturally handle multi-objective optimization, under the assumption that it can navigate the complicated loss-landscape. Many low-thrust optimization tools lack this capability. Additionally, mission constraints, like spacecraft mass, specific impulse and maximum thrust can also be intuitively included as both constraints and objectives. This feature makes the method flexible and of particular interest for analysis in a concurrent design phase, due to the lack of tools capable of doing this. Finally, it has been found that the success in converging to a solution is highly dependent on the initialization of the network, which appears to fall into two groups: failing and non-failing. Fortunately, it could be fairly early recognized during training in which group a solution is going to fall allowing to implement a restarting mechanism. However, in some cases, even after passing the criteria, a physically valid solution can not be constructed. A more theoretical understanding of what drives failed initializations would be desirable in order to make the method more reliable.

CONCLUSION

An unsupervised Physics-Constrained Neural network has been configured to solve optimal control problems with the direct method in order to design fuel-optimal low-thrust transfer trajectory problems. The objective is included as a loss term in the training procedure and the network is made to analytically satisfy the boundary values via a set of constraint equations. With a planar Earth-Mars low-thrust transfer and rendezvous as a test scenario, it was found that the constraints on dynamics in combination with the objective span a capricious loss landscape that is difficult to navigate. However, some unusual strategies, like re-initializing the network under certain conditions and temporarily increasing the learning rate to escape local minima, allow training to succeed in finding near-optimal solutions. The method finds solutions without any prior assumptions of the trajectories shape and does so solely guided by the physics-informed nature of the neural network training process. Upon evaluating solutions across numerous transfer opportunities and comparing them with those obtained via hodographic shaping, it has been determined that the PCNN demonstrates proficiency in identifying near-optimal solutions across a diverse set of constraints. Furthermore, the PCNN exhibits superior performance in terms of optimality, particularly in scenarios characterized by higher degrees of non-linearity, when compared to the shape-based method. This feature allows this method to be a potential preliminary design tool for optimal trajectories in nonlinear dynamics.

REFERENCES

[1] S. Mazouffre, "Electric propulsion for satellites and spacecraft: established technologies and novel approaches," *Plasma Sources Science and Technology*, Vol. 25, No. 3, 2016, p. 033002.

[2] C. E. Garner, M. M. Rayman, G. J. Whiffen, J. R. Brophy, and S. C. Mikes, "Ion propulsion: an enabling technology for the dawn mission," *AAS/AIAA Spaceflight Mechanics Meeting*, 2013, pp. AAS 13–342.

[3] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[4] L. Kos, T. Polsgrove, R. Hopkins, D. Thomas, and J. Sims, "Overview of the development for a suite of low-thrust trajectory analysis tools," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006, p. 6743.

[5] D. Morante, M. Sanjurjo Rivo, and M. Soler, "A survey on low-thrust trajectory optimization approaches," *Aerospace*, Vol. 8, No. 3, 2021, p. 88.

[6] M. Pontani and B. A. Conway, "Particle swarm optimization applied to space trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1429–1441.

[7] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the astronautical Sciences*, Vol. 135, No. 1, 2009, pp. 497–528.

[8] A. Rocchi and R. Jehn, "Low-thrust navigation tools at esoc mission analysis section," *6th International Conference on Astrodynamics Tools and Techniques, 2016, Darmstadt*, 2016.

[9] R. Jehn, D. Garcia, J. Schoenmaekers, and V. Companys, "Trajectory design for BepiColombo based on navigation requirements," *Journal of Aerospace Engineering*, Vol. 4, No. 1, 2012, p. 1.

[10] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, Vol. 378, 2019, pp. 686–707.

[11] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, Vol. 9, No. 5, 1998, pp. 987–1000.

[12] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, Vol. 3, No. 6, 2021, pp. 422–440.

[13] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," *Journal of Scientific Computing*, Vol. 92, No. 3, 2022, p. 88.

[14] J. Martin and H. Schaub, "Physics-informed neural networks for gravity field modeling of the Earth and Moon," *Celestial Mechanics and Dynamical Astronomy*, Vol. 134, No. 2, 2022, p. 13.

[15] J. Martin and H. Schaub, "Physics-informed neural networks for gravity field modeling of small bodies," *Celestial Mechanics and Dynamical Astronomy*, Vol. 134, No. 5, 2022, p. 46.

[16] A. Scorsoglio, L. Ghilardi, and R. Furfaro, "A Physic-Informed Neural Network Approach to Orbit Determination," *The Journal of the Astronautical Sciences*, Vol. 70, No. 4, 2023, pp. 1–30.

[17] E. Schiassi, A. D'Ambrosio, K. Drozd, F. Curti, and R. Furfaro, "Physics-informed neural networks for optimal planar orbit transfers," *Journal of Spacecraft and Rockets*, Vol. 59, No. 3, 2022, pp. 834–849.

[18] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, and D. Mortari, "Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations," *Neurocomputing*, Vol. 457, 2021, pp. 334–356.

[19] S. Mowlavi and S. Nabi, "Optimal control of PDEs using physics-informed neural networks," *Journal of Computational Physics*, Vol. 473, 2023, p. 111731.

[20] M. Mattheakis, D. Sondak, A. S. Dogra, and P. Protopapas, "Hamiltonian neural networks for solving equations of motion," *Physical Review E*, Vol. 105, No. 6, 2022, p. 065305.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Review*, Vol. 63, No. 1, 2021, pp. 208–228, 10.1137/19M1274067.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. Software available from tensorflow.org.

[24] D. Dirkx, M. Fayolle, G. Garrett, M. Avillez, K. Cowan, S. Cowan, J. Encarnacao, C. F. Lombrana, J. Gaffarel, J. Hener, *et al.*, "The open-source astrodynamics Tudatpy software–overview for planetary mission design and science analysis," *EPSC2022*, No. EPSC2022-253, 2022.

[25] D. H. P. C. C. (DHPC), "DelftBlue Supercomputer (Phase 1)," `https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1`, 2022.

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

[27] C. H. Acton Jr, "Ancillary data services of NASA's navigation and ancillary information facility," *Planetary and Space Science*, Vol. 44, No. 1, 1996, pp. 65–70.

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

[29] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: pagmo," *Journal of Open Source Software*, Vol. 5, No. 53, 2020, p. 2338.

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

A

Background on the used Methods

This chapter provides additional background to the methods that are used in the research as described in the paper. The paper is written in a form that is meant to be eligible for admission to a conference in the field of astrodynamics. In order to be concise, it assumes certain knowledge that not every space engineer is readily equipped with. For example, space engineering students are not naturally exposed to the field of machine learning and specifically the concept of Physics-Informed Neural Networks. Besides, the frame of references, coordinates and conversion between them are quite briefly introduced in the paper. This being a master thesis, some additional clarification on these matters is appropriate and this appendix is meant to guide students and faculty members in the Astrodynamics and Space missions section in comprehending the work.

A.1. Artificial Neural Networks

Within the field of machine learning, Artificial Neural Networks (ANN) are universal function approximaters [37] that are loosely based on the connections between neurons in animal brains. Due to their descriptions in terms of linear transformation, they can be efficiently differentiated and be iteratively updated by gradient descent algorithms to perform tasks such as regression, classification and clustering. At the same time they are capable of describing highly non-linear mappings with the use of activation functions and the feasibility of making them really large, which is referred to as a Deep Neural Network (DNN). The large amount of parameters that make up the network and dynamic training of them often result in thinking of the neural network as a black box, because the internal structure that the network takes on has no intuitive interpretation and is not containing any fundamental information about the system that is researched. Nevertheless, neural networks have been highly successful in finding patterns in large datasets even though they don't provide insight into the mechanisms behind the patterns. The Physics-Informed neural network (PINN) partially lifts this black box nature by infusing physical laws into the network architecture or training procedure. This section will elaborate on the mathematical description of an ANN, the training procedure and the subclass of PINNs.

A.1.1. Mathematical formulation

Mathematically the ANN can be described as a series of linear images mapping the input to various hidden layers and then finally to an output layer. A *layer* is nomenclature for a vector and the various entries are referred to as *neurons*. Consider an ANN $\mathcal{N}$ that maps an input layer $\mathbf{x} \in \mathbb{R}^P$ to an output layer $\mathbf{y} \in \mathbb{R}^Q$. The various entries in the input layer are often referred to as *features* while the entries in the output layer are called *labels* or *targets*.

$$\mathbf{y} = \mathcal{N}_{\mathbf{p}}(\mathbf{x}) \tag{A.1}$$

All the weights and biases that define the linear mappings are collected together in $\mathbf{p}$. In neural networks, often hidden layers are defined in between the input and output layers to be able to more easily capture complicated dependencies. That means that the input layer is not necessarily mapped directly to the

output layers, but to a sequence of hidden layers up to the output layer. A generic mapping of layer $\mathbf{h}^i \in \mathbb{R}^P$ to $\mathbf{h}^{i+1} \in \mathbb{R}^Q$, where the superscript refers to the layer, is an operation that can be vectorized and written as matrix equation A.2.

$$\mathbf{h}^{i+1} = \sigma^{(i+1)}\left(\boldsymbol{W}^{(i+1)}\mathbf{h}^i + \boldsymbol{b}^{(\boldsymbol{i+1})}\right) \tag{A.2}$$

$$= \sigma^{(i+1)}\left(\begin{bmatrix} w_{00}^{(i+1)} & \dots & w_{0P}^{(i+1)} \\ \vdots & \ddots & \vdots \\ w_{Q0}^{(i+1)} & \dots & w_{QP}^{(i+1)} \end{bmatrix}\begin{pmatrix} h_0^i \\ \vdots \\ h_P^i \end{pmatrix} + \begin{pmatrix} b_0^{(i+1)} \\ \vdots \\ b_Q^{(i+1)} \end{pmatrix}\right) \tag{A.3}$$

The outcome after multiplication with the weight matrix $\mathbf{W}^{i+1}$ and addition with bias vector $\mathbf{b}^{i+1}$ is often extended by the insertion into a (non-linear) *activation function* σ^{i+1}. Many activation functions have been proposed [38]. Some commonly selected options are the linear activation function, the sigmoid, the relu and the hyperbolic tangent.

$$\text{relu}(x) = \max(0, x) \tag{A.4}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{A.5}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{A.6}$$

$$\text{linear}(x) = x \tag{A.7}$$

$$\tag{A.8}$$

An alternative option is the sine function, which has been shown to be superior in at least some physics-informed neural networks [39]. The amount of hidden layers, their neurons, their activation functions and the connections between them are together referred to as the *architecture* of the network. If all neurons in layer i are connected to those in layer $i + 1$ and $i - 1$, like in the mathematical example given above, it is called a *fully-connected feed-forward network*. Non-fully connected neural networks are also possible. For instance, in a convolutional neural network (CNN), not all neurons are connected with each other due to the presence of convolutional and pooling layers. These types of networks are very suitable for image recognition tasks [40]. The feed forward nature of a network is also not required, as the opposing Recurrent Neural Network (RNN) includes a form of feedback loop in the evaluation of the network allowing them to more successfully capture sequences of data like time-series or recognizing speech [41]. Finding an architecture that best fits the problem is a laborious process which is commonly done heuristically by performing large grid searches.

A.1.2. Training

When defining a neural network, all the weights and biases are randomly initiated. Training a neural network is the process of choosing the weights and biases such that the network performs the required task. This is typically done by a gradient descent based optimization scheme after the derivatives of a loss function with respect to the weights are acquired by a process called *back propagation*. First a loss function is defined that has to be optimized. The most common example is to have the ANN mimic a set of many input-output pairs, collectively referred to as a training set. A sample with N entries is taken from the set and the inputs are all evaluated by the neural network. Such an evaluation of the network is called a *forward pass*. Subsequently, the outcome is compared to the target values in a loss function. The average loss is calculated over all N input/output pairs in the sample. The loss can be squared to be more sensitive to outliers (equation A.9), but this is not necessarily the best option for all applications.

$$\mathcal{L} = \frac{1}{N}\sum_i^N \left[\mathcal{N}_{\mathbf{p}}(\mathbf{x}_i) - \mathbf{y}_i\right]^2 \tag{A.9}$$

The back-propagation algorithm is able to acquire the derivatives of the loss function with respect to the weights and the biases by successively applying the chain rule, a step called the *backward pass*. Modern

neural network software, like TensorFlow and PyTorch, use a technique called *automatic differentiation* (AD) to perform this differentiation. AD tracks the basic arithmetic operations with known derivatives performed by a computer and sequentially applies the chain rule [42]. The accuracy of the derivatives is then to machine level precision and it can be performed fast. Gradient descent algorithms, like the Adam optimizer [43], are then used to update the weights and biases accordingly. A generic update step is written in equation A.10.

$$w_{jk}^{(i)} = w_{jk}^{(i)} - \eta \frac{\partial \mathcal{L}}{\partial w_{jk}^{(i)}} \tag{A.10}$$

The learning rate parameter η is defining how large the update steps are. A too small learning rate results in too small steps which slows down training. Too large learning rate might not find a narrow minimum and skips over it. The clever Adam optimizer tries to overcome this by having a separate adaptable learning rate per weight. Nevertheless, it still requires a choice for the initial learning rate which greatly influences the step sizes taken. Depending on the training algorithm, several additional parameters might be required to control the training and these types of parameters are called *hyperparameters*. For instance, the Adam optimizer also requires the declaration of the betas β_1 and β_2 which control the learning rate's ability to adapt. To efficiently train a neural network, hyper parameter values have to be optimized in a process called tuning.

A.1.3. Physics-Informed

The Physics-informed Neural Network (PINN) uses additional information about the physical process under consideration and inserts this into the neural network. Not only does this practice serve to more easily describe physical processes, it is also an attempt to partially lift the black box nature of the network and make its behavior more predictable. The term physics-informed is not strictly defined and has been used in many different circumstances. A first option is to have the architecture of the neural network implicitly reflect the physical process. For example, network architectures have been designed that enforce the underlying symplectic structure of Hamiltonian systems [44]. More generally, CNNs are capable of resembling certain symmetry groups like rotations and reflections [45]. For the interested reader, many examples of these type of CNN or other architectures inserting physics priors can be found in the overview on physics informed machine learning by Karniadakis et al. [30]. A different approach to including physics into neural networks is by regularizing the loss term with physical models in the form of differential equations. This type of physics infusion is the one under consideration in this work and will be treated in a bit more detail in this section. The first mentioning of such systems go back to the nineties by Lagaris et al. [29]. However, in 2019 it has been revived into its current form by Raissi et al. [28]. The building blocks to create such a network will be reiterated here to demonstrate how PINNs can be used to solve forward problems in an unsupervised manner and solve inverse and regression problems in a supervised manner.

Consider a physical model providing an approximate description of a physical system in the real world. Mathematically, the model is written as a partial differential equation (PDE). Note that an ordinary differential equation (ODE) is a subclass of PDE. Equation A.11 is an example of an arbitrary physical model written as an implicit PDE $\mathcal{F}$. This model is extended by the presence of boundary conditions $\mathcal{B}$ in the space domain and an initial condition $\mathcal{I}$ in the temporal domain.

$$\mathcal{F}(u(\mathbf{x},t),\mathbf{x},t,\boldsymbol{\gamma}) = 0 \quad \mathbf{x} \in \Omega,\ t \in [t_0, t_f] \tag{A.11}$$

$$\mathcal{B}(u(\mathbf{x},t),\mathbf{x},t,\boldsymbol{\gamma}) = 0 \quad \mathbf{x} \in \partial\Omega,\ t \in [t_0, t_f] \tag{A.12}$$

$$\mathcal{I}(u(\mathbf{x},t),\mathbf{x},t,\boldsymbol{\gamma}) = 0 \quad \mathbf{x} \in \Omega, t = t_0 \tag{A.13}$$

where $\mathbf{x}$ represents the space coordinates in the domain Ω, t the time parameter, $u(\mathbf{x},t)$ the solution to the PDE and $\boldsymbol{\gamma}$ the physical parameters of the environment. $\mathcal{F}$ can contain one or multiple derivatives of the solution $\nabla_{\mathbf{x}} u$ and $\nabla_t u$. A neural network, a universal approximater, is assigned as the solution and maps $(\mathbf{x},t)$ to u. The physical system is being observed in the real world (or simulated) and this results in N_{data} input/output pairs $\big((\mathbf{x},t),u\big)$. Note that this data has errors and might also be prone to

perturbations with respect to the model. To regress this data and have the neural network describe the solution, it can be trained in a supervised manner by feeding it this data and establishing a loss function $\mathcal{L}_{\text{data}}$.

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i}^{N_{\text{data}}} \left[\mathcal{N}_{\mathbf{p}}(\mathbf{x}_i, t_i) - u_i \right]^2 \tag{A.14}$$

The supervised network will now be made physics-informed by including a loss term $\mathcal{L}_{\text{PDE}}$ that reflects the PDE in equation A.11. This term penalizes the data loss if the output of the network does not comply with the underlying physical model. Automatic differentiation is used to acquire the derivatives that might be present in $\mathcal{F}$. These derivatives, together with $\mathbf{x}$, t and u are plugged into the PDE to quantify the residual. The input domain from the observations is extended by a set of N_{PDE} additional inputs called *collocation points* and these are used to build the loss due to the residual of the PDE.

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}} \sum_{i}^{N_{\text{PDE}}} \left[\mathcal{F}(\mathbf{x}_i, t_i, u_i) \right]^2 \tag{A.15}$$

This way the model is not only learning to satisfy the data but also the physical model which allows to generalize outside the observed domain and to simultaneously filter errors in the data. An extension can be to include a soft enforcement of the boundary conditions and initial conditions via more loss terms. A third set of N_{BC} collocation points is sampled on the physical boundary and a fourth set of N_{IC} collocation points on the initial time to establish the loss terms $\mathcal{L}_{\text{BC}}$ and $\mathcal{L}_{\text{IC}}$.

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{i}^{N_{\text{BC}}} \left[\mathcal{B}(\mathbf{x}_i, t_i, u_I) \right]^2 \tag{A.16}$$

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{i}^{N_{\text{IC}}} \left[\mathcal{I}(\mathbf{x}_i, t_i, u_I) \right]^2 \tag{A.17}$$

Alternatively, boundary and/or initial conditions can be satisfied analytically via an additional layer to the network. When this type of hard constraint is included, the network is often called a *Physics-Constrained Neural Network* (PCNN). When implementing this, one should make sure that the automatic differentiation tracks the operations performed in the constraint layer to acquire the derivative of the outcome. This hard constraint is what was implemented in the paper to satisfy the initial and final constraint in the dynamics problem encountered during the low-thrust transfer problem.

The four different loss functions encountered constitute the framework of the Physics-Informed Neural Network. A linear combination between the loss terms and a set of loss weights make up the total loss $\mathcal{L}$.

$$\mathcal{L} = \omega_{\text{data}}\mathcal{L}_{\text{data}} + \omega_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \omega_{\text{BC}}\mathcal{L}_{\text{BC}} + \omega_{\text{IC}}\mathcal{L}_{\text{IC}} \tag{A.18}$$

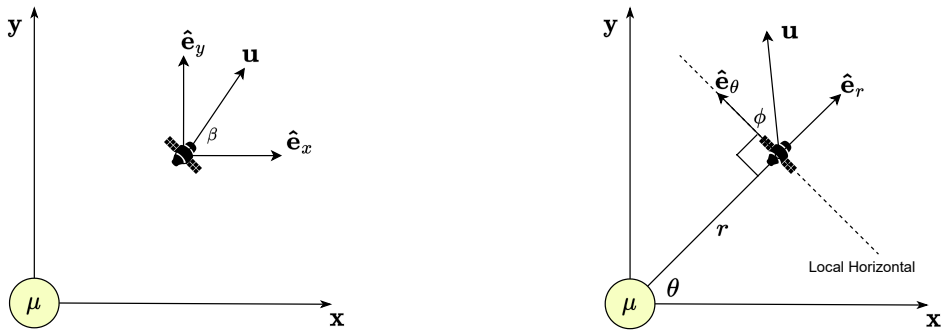
The several loss terms together span an intricate loss landscape that can be quite capricious. A widely encountered issue in training PINNs is competing loss terms [46]. This issue is actively being researched with one of the proposed solutions being adaptive loss weights training algorithms [47].

Depending on the problem, not all loss terms have to be present. Many configurations are possible. When using all loss terms, a physics-informed regression is performed on observed data, which is quite similar to a Kalman Filter often used in space flight applications to handle state estimation. Alternatively, it is possible to extend this system by defining environmental parameters γ in the differential equations as trainable parameters which are updated simultaneously with the weights and biases during training, thus solving an inverse problem in a physics-informed way. Finally, in the absence of data $\mathcal{L}_{\text{data}}$,

when the input samples only contain simulated collocation points that do not have any target data associated to them, the PINN is converted into an unsupervised method to solve the PDE (or ODE). This configuration thus serves as an alternative to the traditional finite element methods or numerical integrators that are built on the discretization of the input domain. The major advantage of the PINN is that it produces a continuous and differentiable solution that does not suffer from discretization errors. The numerical accuracy of the solution to the PDE as described by the PINN is determined by the expressivity of the network.

A.2. Reference Frames, Coordinates and Transformations

In this work the ECLIPJ2000 reference as defined by SPICE [48] was used as a reference frame where the z-axis was omitted in order to describe planar dynamics. Within this framework, two coordinate systems have been used, polar coordinates (Figure A.1b) and cartesian coordinates (Figure A.1a).



(a) Cartesian **(b)** Polar

Figure A.1: Coordinate systems

In the cartesian coordinate system, the position vector $\begin{bmatrix} x & y \end{bmatrix}^T$, velocity vector $\begin{bmatrix} v_x & v_y \end{bmatrix}^T$ and thrust vector $\begin{bmatrix} u_x & u_y \end{bmatrix}^T$ of the spacecraft is described in the $\{\mathbf{e}_x, \mathbf{e}_y\}$ basis. The position of the spacecraft in polar coordinates is described with $\begin{bmatrix} r & \theta \end{bmatrix}^T$ which is complemented with the velocity vector $\begin{bmatrix} v_r & v_\theta \end{bmatrix}^T$ and thrust vector $\begin{bmatrix} u_r & u_\theta \end{bmatrix}^T$ in the $\{\mathbf{e}_r, \mathbf{e}_\theta\}$ basis. Transformation between cartesian and polar positions are performed via

$$x = r\cos\theta \tag{A.19}$$
$$y = r\sin\theta \tag{A.20}$$
$$r = \sqrt{x^2 + y^2} \tag{A.21}$$
$$\theta = \arctan\left(\frac{y}{x}\right) \tag{A.22}$$

The transformation between polar (P) and cartesian (C) coordinates for the velocity and thrust vectors can be performed by means of the basis transformation matrices $\mathcal{R}_{\mathrm{C}\to\mathrm{P}}$ and $\mathcal{R}_{\mathrm{P}\to\mathrm{C}}$.

$$\mathcal{R}_{\mathrm{C}\to\mathrm{P}} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{A.23}$$
$$\mathcal{R}_{\mathrm{P}\to\mathrm{C}} = \mathcal{R}^{-1}_{\mathrm{C}\to\mathrm{P}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{A.24}$$

The velocity and control are then converted via

$$\begin{bmatrix} v_r \\ v_\theta \end{bmatrix} = \mathcal{R}_{\mathrm{C}\to\mathrm{P}} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{A.25}$$
$$\begin{bmatrix} u_r \\ u_\theta \end{bmatrix} = \mathcal{R}_{\mathrm{C}\to\mathrm{P}} \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{A.26}$$
$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \mathcal{R}_{\mathrm{P}\to\mathrm{C}} \begin{bmatrix} v_r \\ v_\theta \end{bmatrix} \tag{A.27}$$
$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \mathcal{R}_{\mathrm{P}\to\mathrm{C}} \begin{bmatrix} u_r \\ u_\theta \end{bmatrix} \tag{A.28}$$

Finally, it is possible to describe the control vector in terms of the magnitude of the control vector $||u||$ and an angle. In the case of cartesian coordinates that angle is measured from the x-axis to the control vector. In the case of polar coordinates that angle is measured from the local horizontal to the control vector. The control can than be written as

$$u_x = ||u|| \cos\beta \tag{A.29}$$
$$u_y = ||u|| \sin\beta \tag{A.30}$$
$$u_r = ||u|| \sin\phi \tag{A.31}$$
$$u_\theta = ||u|| \cos\phi \tag{A.32}$$

The neural network from the paper described the spacecrafts motion in terms of polar coordinates where the control is parameterized in terms of ϕ and $||u||$. Similarly, the neural network build on cartesian coordinates given in the appendix on additional findings is parameterized in terms of $||u||$ and β for the control. The verification numerical integrations performed to establish the metrics dx, dv and dm made use of cartesian coordinates where the control was inserted in its inertial description u_x, u_y.

B

Verification and Validation

B.1. Verification

Verification is the process of making sure that the methods and models have been implemented correctly and comply with the requirements of the system. In the context of this work, that means that the physics-constrained neural network (PCNN) should be correctly implemented, the dynamical model should be correctly implemented and that the results it produces are in fact physically valid solutions. Verification on these three aspects will be explicitly elaborated on in this chapter. Finally, the implementation of the hodographic shaping method used in the paper for validation will be verified.

B.1.1. Verification of PCNN implementation

The PCNN is implemented with the DeepXDE [49] python package which is specifically designed to do research on Physics-Informed Neural Networks. It has the capability of working with different neural network software tools as backend. The TensorFlow [50] backend was chosen, which is widely used for neural network applications in both industry and for scientific research and therefore does not require explicit verification. Although DeepXDE is a peer-reviewed piece of software and quite commonly used in the PINN community, it is fit to verify its implementation because of the continuous development that it undergoes. Mattheakis et al. [39] publish a method to use a physics-informed neural network in supervised form to solve Hamilton's equations of motion, thereby classifying as a Hamiltonian Neural Network (HNN). Their work served as inspiration for the method to enforce constraints analytically as well as for the sampling of collocation points, in a perturbed equidistant grid of time points. If the final state constraint, the control output and objective loss are stripped from the method proposed in the paper, the neural network configuration proposed by Mattheakis et al. remains. It is chosen to recreate part of this work in order to (1) verify the correct implementation of constraint layers (2) verify the correct implementation of collocation sampling method (3) verify the correct implementation of the DeepXDE package and (4) verify the correct handling of loss functions with automatic differentiation.

In their publication paper, Mattheakis et al. solve an initial value problem of a nonlinear harmonic oscillator described in phase space $\mathbf{z} = \begin{bmatrix} x & p \end{bmatrix}^T$. The Hamiltonian[1] of the system is written in equation B.1

$$\mathcal{H}(x,p) = \frac{p^2}{2} + \frac{x^2}{2} + \frac{x^4}{4} \tag{B.1}$$

Applying Hamilton's equation results in the equations of motion.

[1] Note that this is the Hamiltonian from Hamiltonian mechanics which is different from the Hamiltonian that appears in optimal control theory

$$\dot{x} = \frac{\delta \mathcal{H}}{\delta p} = p \tag{B.2}$$

$$\dot{p} = -\frac{\delta \mathcal{H}}{\delta x} = -(x + x^3) \tag{B.3}$$

A neural network $\mathbf{z} = \mathcal{N}_\mathbf{p}(t)$ is designated as the free part of the solution. The phase space coordinates are calculated by inserting the network's output in a constraint equation that enforces the initial condition.

$$\mathbf{z} = \mathbf{z}_0 + f(t)\mathcal{N}_\mathbf{p}(t) \tag{B.4}$$

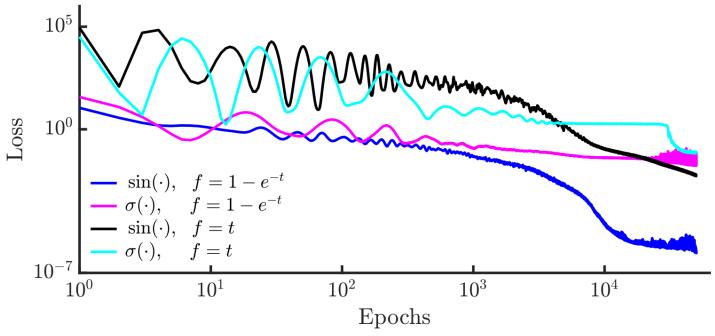
where $f(t)$ is a time-depending function that vanishes at $t = 0$. The loss to be optimized thus contains two terms due to the dynamics, one for x and one for p. Additionally, the loss is regularized by the total energy of the system, because in a Hamiltonian system, energy is conserved. The total loss is written as

$$\mathcal{L} = \mathcal{L}_x + \mathcal{L}_p + \mathcal{L}_{\text{reg}} \tag{B.5}$$

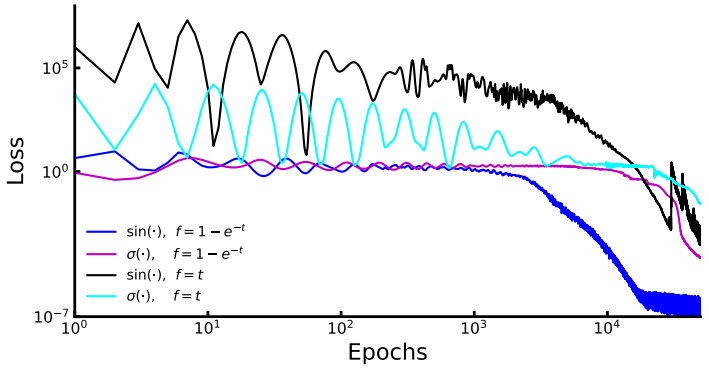
$$= \frac{1}{N}\sum_i^N \left(\dot{x}_i - p_i\right) + \frac{1}{N}\sum_i^N \left(\dot{p}_i + x_i + x_i^3\right) + \frac{1}{N}\sum_i^N \left(\mathcal{H}(x_i, p_i) - E_0\right) \tag{B.6}$$

The authors perform an experiment with as initial condition $\mathbf{z}_0 = \begin{bmatrix}1.3 & 1\end{bmatrix}^T$ which corresponds to $\mathcal{H}(\mathbf{z}_0) = E_0 = 2.06$ and time interval $[0, 4\pi]$. A training set with collocation points is sampled at each training epoch and consists of $N = 200$ points that are distributed in an equidistant grid and subsequently perturbed by a value taken from a normal distribution with mean zero and standard deviation 0.06π. The network is configured with 2 hidden layers each holding 50 neurons. Training is performed by the Adam optimizer for 5×10^4 epochs with a learning rate of 8×10^{-3} and $\beta_1 = 0.999$ and $\beta_2 = 0.999$. The choice of constraint function $f(t)$ and activation function is varied. For the constraint function, the authors consider $f(t) = t$ and $f(t) = 1 - e^{-t}$ and the sigmoid and sin function are considered as activation functions. Training of this situation with corresponding network design variations is repeated in our implementation of this PCNN that is built on top of DeepXDE (and TensorFlow).

Loss evolutions are depicted in figure B.1 where they are compared to the same plot from the original paper. Errors of the solution are quantified by comparison with the ODEINT numerical integrator from the SCIPY module, which has also been used by Mattheakis et al. Those residuals in x and p are plotted in figure B.2. Just like in the original paper, the best configuration is with a sine activation and $f(t) = 1 - e^{-1}$ constraint equation as this converges the fastest and yields the optimal final loss value. The loss trend for the other cases are very similar. Small differences originate from the statistical nature of the neural network training process: a random weight and bias initialization and the perturbed collocation points. The errors for x and p are also on the same scale and the initial condition is enforced. With these nearly identical results and expected behavior, the DeepXDE method with its automatic differentiation and handling of TensorFlow is considered verified. At the same time, the implementation of the constraint layer and the perturbed equidistant collocation sampler are verified.

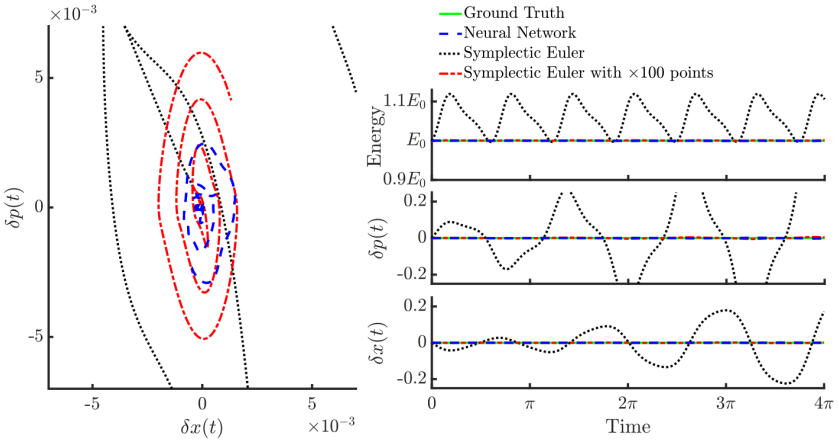


(a) Adapted from Mattheakis et al. [39]

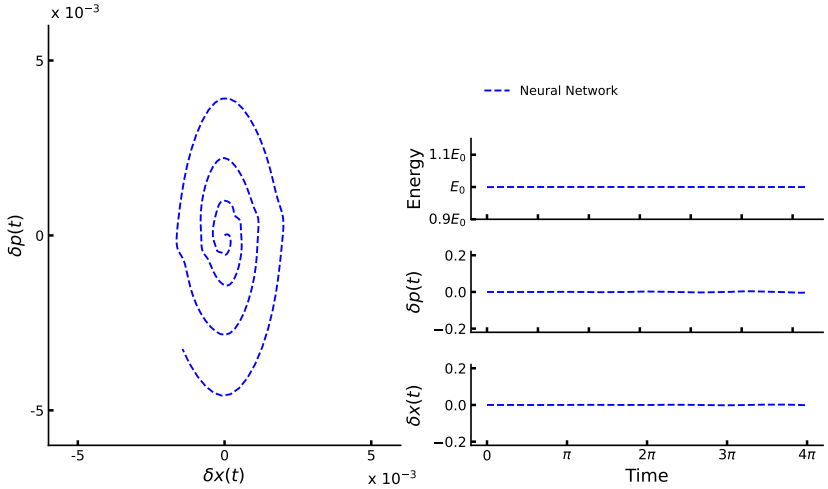


(b) This work. Implementation build on top of DeepXDE+TensorFlow

Figure B.1: Loss evolution's for a nonlinear oscillator problem for four different configurations of the network.



(a) Adapted from Mattheakis et al. [39]



(b) This work. Implementation build on top of DeepXDE+TensorFlow

Figure B.2: Errors on the produced canonical coordinate x and p and the total energy for the best configuration (sin activation and $f(t) = 1 - e^{-1}$).

B.1.2. Verification of physical models in the PCNN

In order to verify that the acceleration model of the two-body problem is correctly implemented in the dynamical loss terms and the network is in fact generally capable of solving these dynamics, the solution of a circular orbit will be created. Secondly, in order to verify that the control and objective loss is implemented correctly, it will be demonstrated that the network converges to a Hohmann transfer if permitted by the constraints.

B.1.2.1.Circular Orbit

Consider a 100 kg spacecraft in a circular orbit around the Earth at an altitude of 500 km. Only the central point mass gravity of the Earth is acting on the spacecraft. The length is scaled by the radius of the orbit $R_\oplus + 500$ km and the time is scaled such that one orbital period corresponds to 2π units of time. As a consequence, the velocity is scaled to the circular velocity at 500 km altitude. The initial state of the spacecraft in polar coordinates in these non-dimensional units is

$$\mathbf{z}_0 = \begin{bmatrix} r_0 \\ \theta_0 \\ v_{r,0} \\ v_{\theta,0} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{B.7}$$

The analytical solution to this initial value problem in polar coordinates can be written as

$$\mathbf{z}(t) = \begin{bmatrix} r(t) \\ \theta(t) \\ v_r(t) \\ v_\theta(t) \end{bmatrix} = \begin{bmatrix} 1 \\ t \\ 0 \\ 1 \end{bmatrix} \tag{B.8}$$

The opportunity is taken to also verify the implementation of the coordinate transformation between polar and cartesian coordinates. Transformation to cartesian coordinates are used to convert a control profile to cartesian coordinates for the verification-numerical-integration and subsequently calculate the metrics dx and dv and plot the transfer in cartesian coordinates in Figure 4. According to the transformations described in Appendix A.2, the analytical solution in cartesian coordinates is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r(t)\cos\theta(t) \\ r(t)\sin\theta(t) \end{bmatrix} \tag{B.9}$$

$$= \begin{bmatrix} \cos t \\ \sin t \end{bmatrix} \tag{B.10}$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \cos\theta(t) & -\sin\theta(t) \\ \sin\theta(t) & \cos\theta(t) \end{bmatrix} \begin{bmatrix} v_r(t) \\ v_\theta(t) \end{bmatrix} \tag{B.11}$$

$$= \begin{bmatrix} -\sin t \\ \cos t \end{bmatrix} \tag{B.12}$$

The orbit will be described by the neural network in terms of polar coordinates, like suggested in the paper. However, compared to figure 3 in the paper, the control output, mass output and objective loss terms are discarded. Obviously there is also no final state enforced. As a neural network the baseline configuration is used with training scheme 3 from table 2 in the paper. Because the loss terms are not competing, a learning rate schedule without shakes is chosen. The residuals of the network solution compared to the analytical solution for both cartesian coordinates and polar coordinates are plotted in Figure B.3. An excellent agreement between the PCNN solution and the analytical solution is found for both coordinate systems with errors in the range of 10^{-5} for both position and velocity, which corresponds to 68 m and 0.076 m s^{-1} in dimensional units, respectively. This allows to conclude that the dynamics in the loss is correctly implemented, the coordinate transformations are correctly implemented and the PCNN design is in fact capable of solving two-body problem dynamics.

B.1.2.2.Hohmann Transfer

The spacecraft considered for the circular orbit above will be equipped with a chemical engine capable of delivering a maximum thrust of $u_{\max} = 1000$ N with $I_{sp} = 400$ s. Just like for the circular orbit, the position and velocities will be scaled by the initial values r_0 and v_0, by selection of the time scaling. The final state of the spacecraft will represent a 3000 km altitude orbit. In non-dimensional units, the final state vector is written in equation B.13.

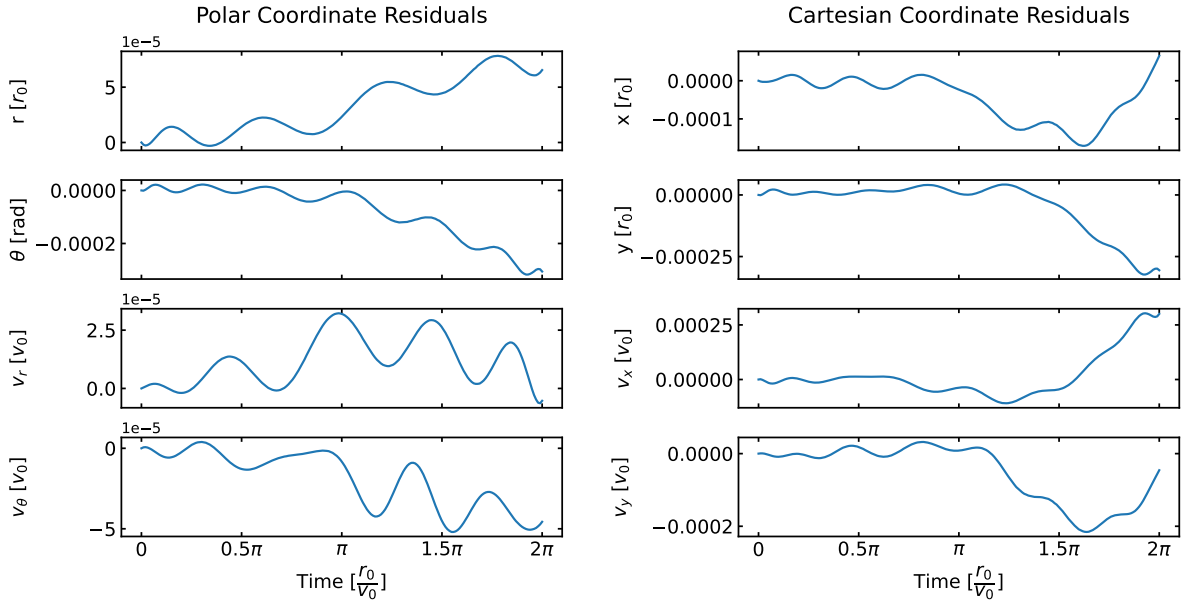


Figure B.3: Solution acquired by the PCNN to a circular orbit in the two-body problem compared to the analytical solution for polar coordinates and cartesian coordinates.

$$\mathbf{z}_f = \begin{bmatrix} r_f \\ \theta_f \\ v_{r,f} \\ v_{\theta,f} \end{bmatrix} = \begin{bmatrix} 1.364 \\ \pi \\ 0 \\ 0.856 \end{bmatrix} \tag{B.13}$$

The time of flight is set accordingly to 60.69 minutes or 4.037 non-dimensional time units. With the high amount of available thrust, the boundary conditions and time of flight, the network should converge towards a Hohmann transfer, which is an analytical optimal transfer solution that assumes the capability of performing instantaneous velocity increments. As configuration of the network, the baseline configuration is used with an objective weight $\omega_0 = 10^{-6}$. The resulting transfer trajectory is depicted in Figure B.4. The metrics for this transfer, as acquired by comparison to the numerical integration of the thrust profile, are $dx = 0.00786\ r_0$, $dv = 0.00411\ v_0$ and $dm = 0.00538$ kg. In dimensional units, this corresponds to $dx = 54.02$ km and $dv = 31.34$ m s^{-1}. The control profile accumulated to $\Delta V = 1.370$ km s^{-1} which takes this engine $m_p = 29.47$ kg of propellant. For comparison, a true Hohmann transfer, which performs the burning of propellant in instantaneous moments, takes $\Delta V = 1.08$ km s^{-1} or $m_p = 24.22$ kg. The burn profile produced by the PCNN indeed approaches the two instantaneous burns in the tangential direction as expected from the Hohmann transfer, demonstrating that the neural network is capable of converging to a near-optimal solution of optimal control problems in the two-body problem. It can be concluded that the extension of the neural network to include a thrust force and an objective loss term has been correctly implemented.

B.1.3. Continuous verification of PCNN solutions

The previous section demonstrated that the PCNN is capable of finding solutions to the two-body problem (circular orbit) and near-optimal solution of a simple transfer problem (Hohmann transfers). Neural networks are driven by statistics and can be highly sensitive to initialization of the weights and biases, which make them inherently unreliable. In this work that means that even though it has been demonstrated that the PCNN method can produce the correct results in cases with analytical solutions, it is not guaranteed that it will do so for every random initialization or in slightly different situations, like more elliptical orbits or low-thrust transfers. Although the verification in the previous sections provides proof that the method is correctly implemented, it does not suggest that this method always produces physically valid solutions. Verification and qualification testing of neural networks

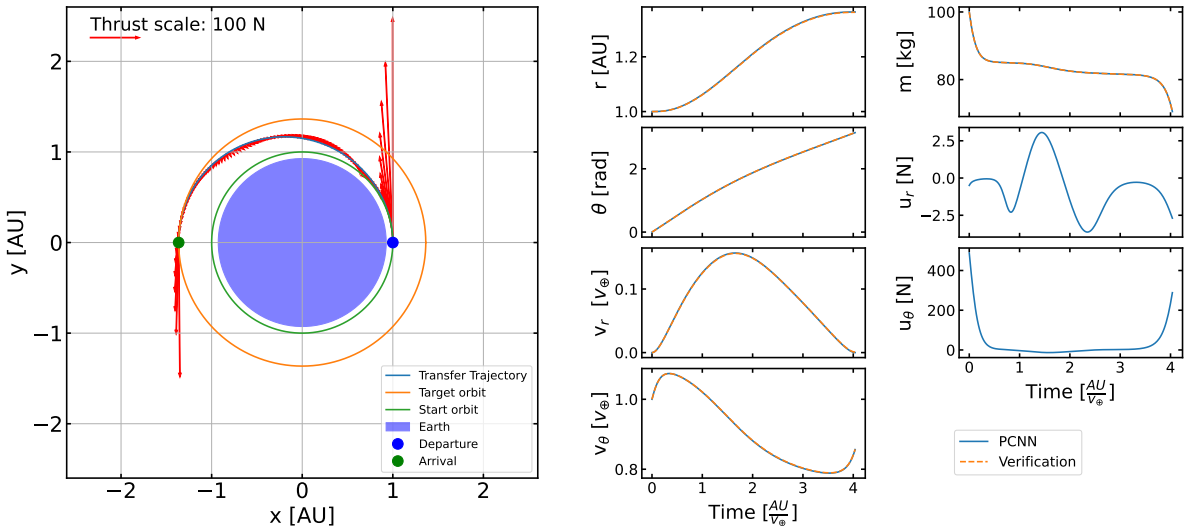


Figure B.4: PCNN solution for a LEO to MEO transfer orbit with chemical engines that approaches the Hohmann transfer with two instantaneous burns, one at periapsis and at apoapsis.

in the context of aerospace systems is in fact a major issue and methods to deal with this are actively being researched. Especially in the context of real-time systems, like flight computers, a rigorous demonstration of the performance is required, which is difficult to carry out for a neural network. The system under development in this work is not intended to be a real-time system, but a tool for preliminary design of trajectories. During this design phase, more time is available to perform explicit verification steps on the spot. For that reason, an approach of continuous verification by comparison to traditional methods has been implemented in order to check for compliance with physical laws. This has been explicitly explained in the section 'Verification of solutions' of the paper. A short summary: the fixed initial state in combination with a control profile produced by the PCNN are numerically integrated by a Runge-Kutta 7(8) variable step-size integrator in order to check whether the produced position and velocity evolution match those found by the network. In doing so, three metrics are defined to quantify compliance with the physical model: dx, dv and dm. In all analysis of the neural network solutions reported on in the paper, these metrics are assessed. *It is recommended to always perform this verification step when using this method and never take the solution provided as the truth.* Especially considering that it has been found that the network is highly sensitive to initialization.

This section will expand on the continuous verification approach by providing a quantification on the discretization error of the numerical integrator selected to perform the verification. Numerical integration has been used in two circumstances in the paper.

1. Integrate the initial state + control profile to obtain $\{\mathbf{x}_v(t_k)\}_{k=0}^{k=M}$ which can be used to calculate the verification metrics dx, dv and dm. This has been performed for every PCNN solution obtained [2].

    - **Integrator 1** Runge-Kutta-Fehlberg (RKF) 7(8) variable time step integrator with relative and absolute tolerance set to 10^{-10}.
    - **Interpolator** scipy.interpolator.cubicspline in order to compare the states produced by the neural networks to the states acquired by the RKF 7(8) integrator at the test time grid points (For example in figure 4 of the paper).

2. Integrate the sensitivity matrix and state in order to iteratively find an empirical RSW-acceleration that closes the gap in dx and dv. This has only been performed in order to quantify the additional ΔV required to close the neural network's error in position and velocity which was then used to select the most appropriate objective weight.

    - **Integrator 2** Runge-Kutta 4 constant time step integrator with time step set to 3600 s.

[2]Moreover, during this research this step was performed at every test evaluation (every 1000 epochs) during PCNN training allowing to see metrics evolve as training progresses.

- **Interpolator** None. No comparison between sets of states was required in order to do this. Only the final state of the integration was compared to the target state to find dx corrected and dv corrected in figure 9 of the paper. As such the integrator was ordered to stop exactly at the termination condition $t = t_f$.

These integrators will be assessed by applying them on the case presented in Figure 4 of the paper. That particular case is 1000 days time of flight, which is approximately the maximum time of flight considered throughout the work. In order to asses the numerical error on these two integrators first a benchmark will be established. As a benchmark, the Dormand-Prince integrator RKDP 8(7) is used with constant time step of 1000 s. The error on the benchmark will be quantified by comparing it to the same integrator with 2000 s time step. Figure B.5 presents the euclidean position difference $||\epsilon_r||$ between these two integrations. The position difference is plotted at the test time points, which is an equidistant time grid of 200 time points between t_0 and t_f that includes the boundaries. A lagrange interpolator of order 7 is used to get the states at these specific moments in time. The smooth error evolution implicates that the interpolator is of sufficiently large order and that truncation error is dominating allowing us to quantify the error of the 1000 s benchmark integrator. The final error is 0.489 m.

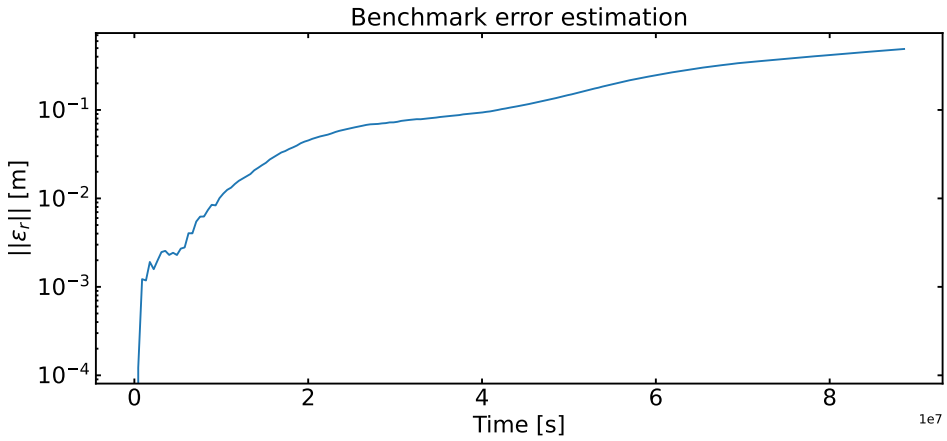


Figure B.5: Numerical error quantification of the benchmark integrator. Euclidean position difference between a Dormand-Prince integrator RKCP 8(7) with time step 1000 s and with 2000 s. The final error is 0.489 m.

Integrator 1 is now used to integrate the control profile and initial state of the same case, Figure 4 from the paper. Here two interpolations are performed, once by the SciPy cubic spline interpolator and once by the lagrange interpolator of order 7. The euclidean position differences by comparison to the benchmark at the training time points is plotted in Figure B.6. From the lagrange interpolation it is concluded that error due to the numerical integrator is dominated by truncation error. The cubic spline interpolation results in an additional error compared to the more accurate lagrange method. The final error comes down to 7.24×10^2 km. The smallest errors found in a neural network solution in the paper are on the order of $dx = 0.002$ AU (figure 8, $\omega_o = 10^{-9}$), which is around 1.5×10^5 km. The error due to numerical integration and interpolation is thus three orders of magnitude smaller verifying that this numerical integrator is suitable for quantifying the verification metrics of the neural network solutions.

Finally, integrator 2 is considered. Again the case from figure 4 in the paper is numerically integrated and compared to the benchmark. Lagrange interpolation with order 7 was used to acquire the states at the test time grid and calculate the euclidean position differences with the benchmark, presented in figure B.7. The final error is 0.60 m, which is on the same order as the error of the benchmark. It is therefore concluded that the numerical error on this integrator is on the order of meters. In the paper this integrator is used to correct the dx and dv metrics to smaller than 1 m, see figure 9. The numerical error is on the same order here. This is no issue, considering that this is an interplanetary mission scenario where accuracy on the order of meters is more than enough anyway.

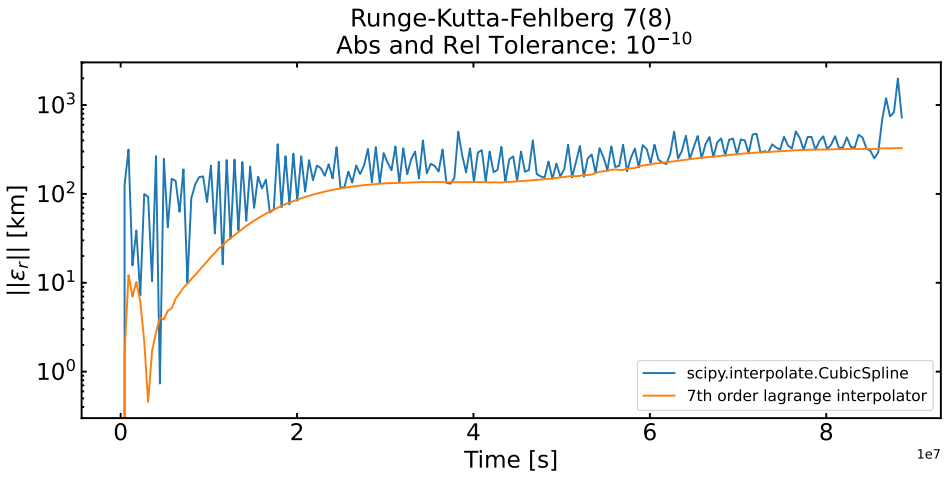


Figure B.6: Numerical error quantification of the RKF7(8) variable time-step integrator. Euclidean position difference by comparison to a Dormand-Prince integrator RKCP 8(7) with constant time step 1000 s and with varying interpolator.

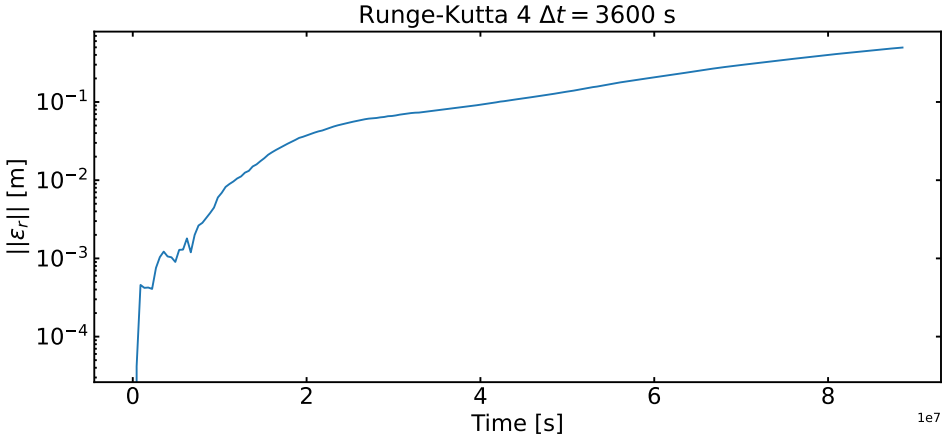


Figure B.7: Numerical error quantification of the RK4 constent time-step integrator with 3600 s time steps. Euclidean position difference by comparison to a Dormand-Prince integrator RKCP 8(7) with constant time step 1000 s after applying a lagrange interpolator of order 7.

B.1.4. Verification of the Hodographic Shaping implementation

In the paper, the optimality of transfer trajectories found by the PCNN is validated by comparison to solutions acquired by a hodographic shaping (HS) method. A python tool [51] to perform hodographic shaping developed by Leon Stubbig as part of his master thesis has been used. Stubbig comprehensively verifies the tool by comparing its output to the results published in the thesis by David Gondelach, who invented the hodographic shaping method. For that reason, no further explicit verification of the tool will be repeated here. However, the correct handling of the tool will shortly be verified here. The problem under consideration at Stubbig and Gondelachs work is a three-dimensional version of the problem considered in this work. So also a low-thrust transfer and rendezvous between Earth and Mars solved for a large amount of departure dates and times of flight. As a consequence the ΔV map for different launch opportunities in the paper (Figure 10, upper right) is also presented in Stubbig's work, albeit for 3D trajectories. Stubbigs tool has been prompted to also recreate the 3D transfer trajectories. The resulting launch opportunity plot is rearranged into a departure date versus time of flight configuration and shown next to Stubbigs version in Figure B.8. The identical colorcode indicates that the tool has been correctly prompted.

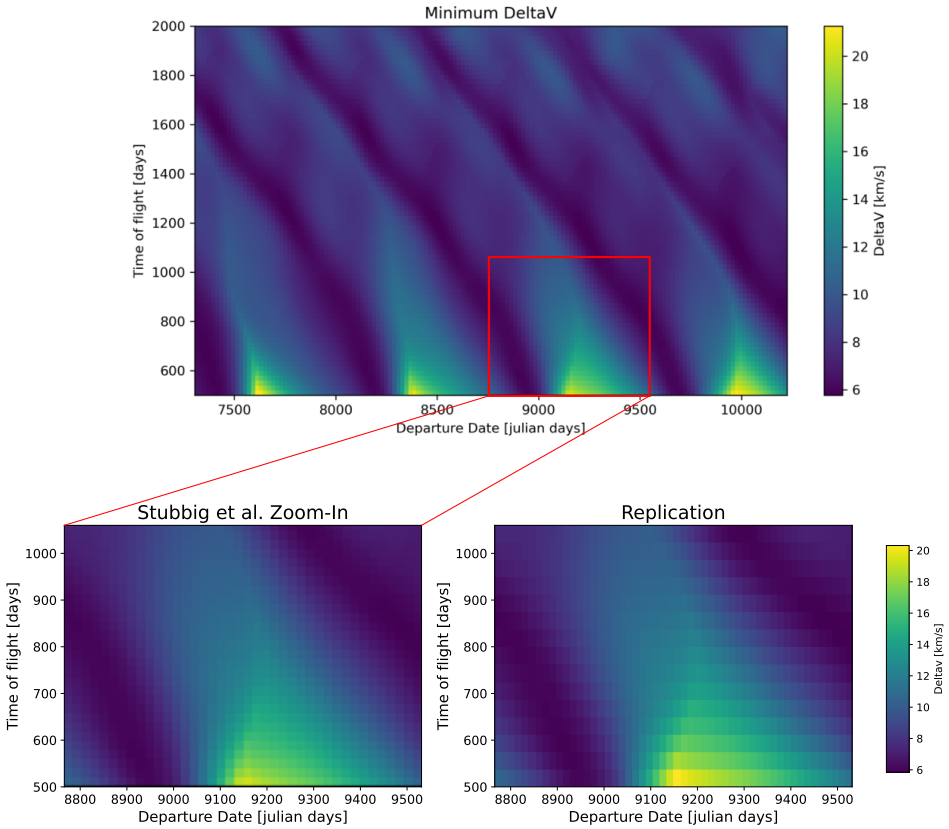


Figure B.8: *Upper:* launch opportunities adapted from Stubbig [51]. *Lower left:* a zoom in on the upper plot to the region considered in this work. *Lower right* Solution acquired by the authors of this work by prompting the Hodographic shaping tool [51] in order to replicate the launch opportunity plot in Stubbig's master thesis.

B.2. Validation

Validation is the process of checking that the system fulfills its intended purpose. The purpose of the Physics-Constrained Neural Network is to find preliminary near-optimal solutions for low-thrust transfer options. In order to validate to what degree the PCNN is capable of carrying out this task, its results have been compared to the results acquired by a well-established preliminary low-thrust trajectory design method: hodographic shaping. This comparison is considered as the validation step of this work and all aspects of this are extensively and quantitatively discussed in the paper.

C

Additional Findings

The conference paper reports on the matters that have been extensively researched and are accompanied with in-depth evidence that can be communicated with great confidence to the scientific community. Throughout this research, additional aspects have been explored, which have not reached the level of understanding required to be included in a conference paper but are still valuable to be communicated in the context of a master thesis to other students and staff. For example, for a considerable period, cartesian coordinates were used to describe the low-thrust transfer. It turned out that that resulted in considerable challenges in describing the transfers, which will be elaborated on here. Secondly, several choices on the neural network design that affected the performance less significantly compared to the training procedure have been researched through grid searches. These will also be reported on in this section.

C.1. Cartesian Coordinates

The paper describes the transfer problem in terms of polar coordinates, but it is equally possible to describe this physical situation in terms of cartesian coordinates like the one defined in Figure A.1a. However, as will be shortly demonstrated in this appendix, the physics constrained neural network experiences some distinct challenges with cartesian coordinates. The equations of motion in cartesian coordinates are written in equations C.1-C.4.

$$\dot{x} = v_x \tag{C.1}$$

$$\dot{y} = v_u \tag{C.2}$$

$$\dot{v_x} = -\frac{\mu}{r^3}x + \frac{||u|| \cos\beta}{m} \tag{C.3}$$

$$\dot{v_y} = -\frac{\mu}{r^3}x + \frac{||u|| \sin\beta}{m} \tag{C.4}$$

where β is the angle measured from the x-axis to the thrust vector $\mathbf{u}$. The same constraint equations are used as in the paper for the entries of the state. The control is again parameterized in the magnitude, which is the same, and an angle, which is now β instead of ϕ. In the control constraint layer of the network, these two thrust entries are treated analogously to the polar coordinate alternatives. Finally, the treatment of the mass output is identical as with polar coordinates. A slightly altered configuration[1] of the network has been used with cartesian coordinates to solve two situations each 40 times. Situation 1 is an Earth-Mars transfer and rendezvous over precisely 0.5 revolutions in 255 days and situation 2 is over precisely 1 revolution in 500 days. The same spacecraft as in the paper is considered. The resulting metrics are summarized in Figure C.1. The 1 revolution situation is behaving quite poorly as it is nearly never able to comply with physics and yields massive amounts of fuel. The 0.5 revolution situation has a better median value, but still shows a large spread in the metrics.

[1]With a [1, 20, 20, 20, 20, 20, 7] fully-connected architecture instead

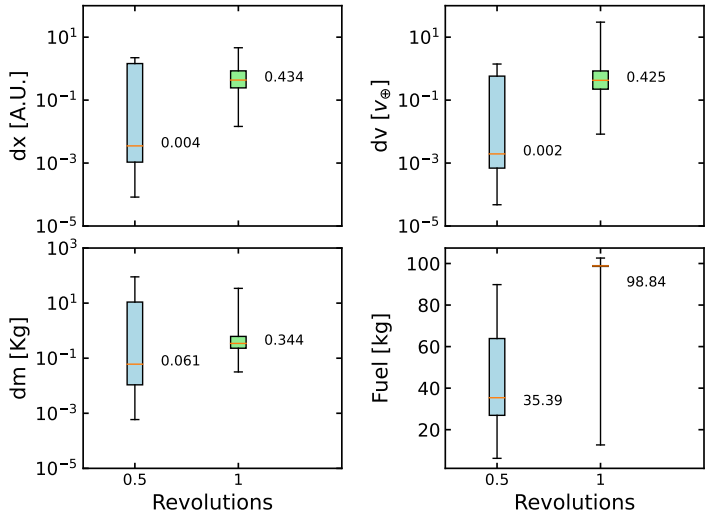


Figure C.1: Verification metrics for two transfers optimized with cartesian coordinates, each 40 times. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

Closer inspection on individual cases reveal that both situations suffer from a separate local minimum that is very hard to escape. A failed case for both situations is plotted in Figure C.2 and Figure C.3. For the 0.5 revolution orbit, the network gets stuck in the option of performing the transfer clockwise, opposite of its initial orientation. Unfortunately, it is not straightforward to recognize early in training that this situation is being created. A restarting mechanism as implemented for polar coordinates is therefore an unattractive solution for this problem. When considering a full orbit transfer, like in the second situation, the network is not aware that it can perform a revolution for the vast majority of initializations. Instead, a route in the vicinity of both boundaries is created that tries to pass time by flying away for a bit and then turning around. In the description in terms of polar coordinates, the amount of revolutions is specified in the final angle θ_f, guiding the network into a certain solution. This is not the case for cartesian coordinates, as there is no natural method to embed the amount of revolutions into the constraints x_f and y_f. For both situations, the gradient descent optimization is incapable of 'lifting the orbit over the sun' as this requires to bridge a large mountain in the loss-valley. These nasty local minima have resulted in the decision to not work with cartesian coordinates in this work.

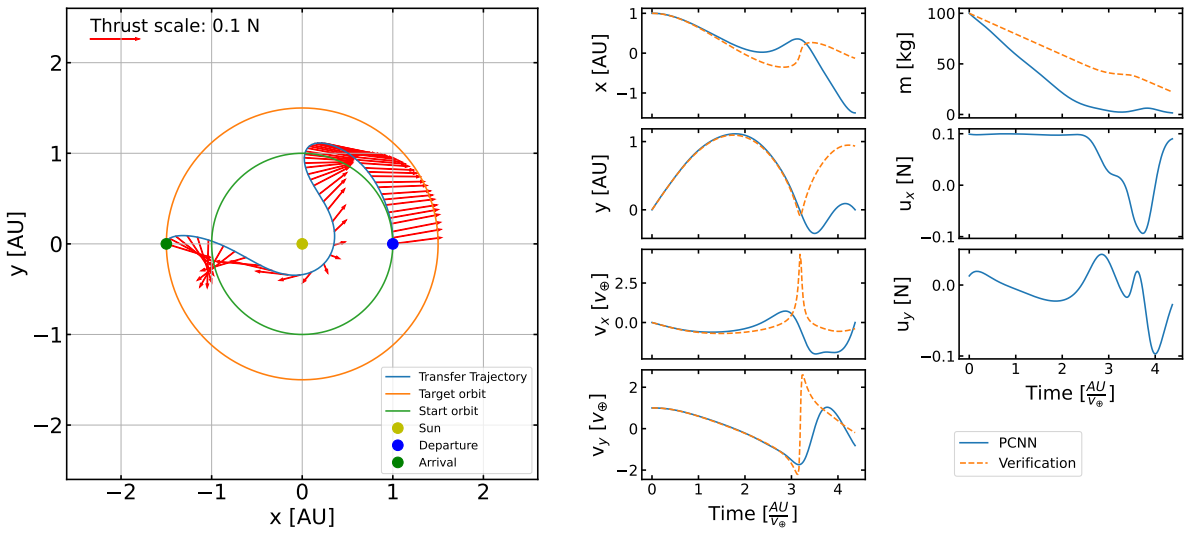


Figure C.2: Planar Earth-Mars 0.5 revolution transfer and rendezvous solved with cartesian coordinates getting stuck in a local minimum. Metrics for this case are $dx = 1.65$ AU, $dv = 0.73\, v_\oplus$, $dm = -20.67$ kg and $m_p = 77.76$ kg.

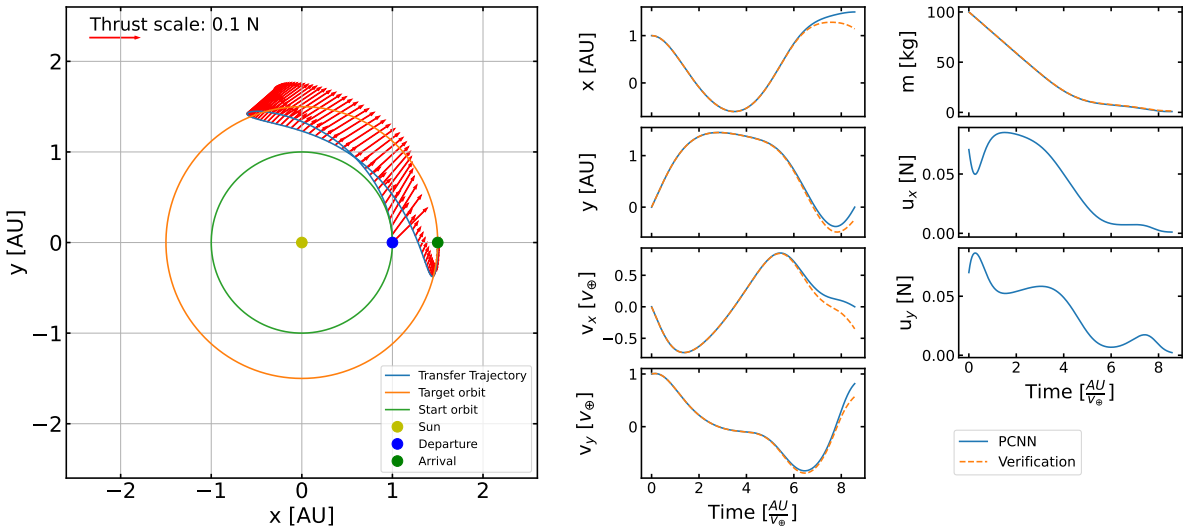


Figure C.3: Planar Earth-Mars 1 revolution transfer and rendezvous solved with cartesian coordinates getting stuck in a local minimum. Metrics for this case are $dx = 0.43$ AU, $dv = 0.43\, v_\oplus$, $dm = -0.56$ kg and $m_p = 98.59$ kg.

C.2. Grid search: Network Architectures

The network architecture determines how much information can be contained in the network and thus pertains to the accuracy capable of achieving. At the same time it might be possible that certain architectures have similarities with the underlying physical process under consideration and can better capture the dynamics. However, the architectures considered in this work are rather straightforward compared to non-straightforward dynamics to describe, so this was not anticipated to be the case. Generally, larger networks have more freedom in the shapes they can take on, which allows them to make more creative time evolutions of state and control entries. Two large grid searches have been employed in comparing basic neuron and layer size combinations as well as parallel networks in a bundle compared to single feed forward neural networks. The baseline configuration of the network from the paper was applied to the same 2-revolution transfer and rendezvous to Mars, with various architectures. A bundle of parallel networks refers to the architecture of having a separate network map time to each entry in the state, each entry in the control and the mass. In the example handled in the paper, that meant 7 parallel networks made up a bundle. Figure C.4 compares network sizes for a single fully connected neural network and figure C.5 compares bundles of networks. The notation $[1,\ 10,\ 10,\ 10,\ 7]$ describes a neural network architecture with an input layer with 1 neuron (time), an output layer with 7 neurons and 3 hidden layers with each 10 neurons. In the plots, the architectures are sorted by total number of parameters in the weight matrices and bias vectors, the figure in parenthesis indicates the amount of parameters. Consider first the fully connected cases. Some observations can be made.

- Generally, more parameters allows for a more physically correct description of the trajectory as indicated by the decreasing median values for dx and dv for increasing network size.
- No specific architectures stand out, although it appears that single hidden layer options are generally unfavorable. Examples are the 50 and 20 neurons options with 1 hidden layer, which are worse then the architectures with similar amount of parameters and much worse than when including more hidden layers with this amount of neurons.
- For very large networks, with more than 10k parameters, the accuracy start to decrease again. This is attributed to the heavily overparameterized design space. The networks most likely require longer training or larger training batches in order to successfully converge to a solution.

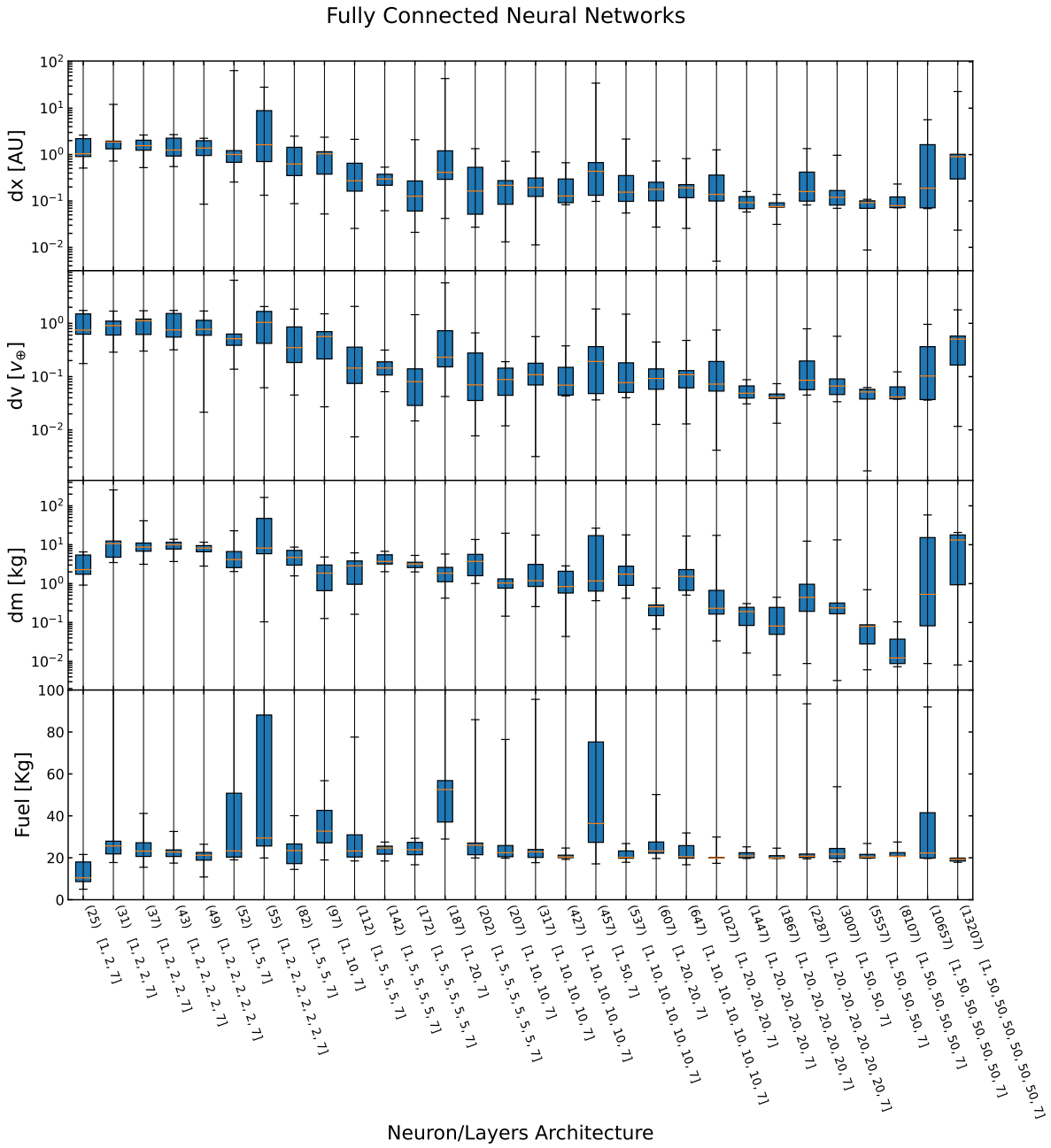


Figure C.4: Grid search over fully-connected neural network architectures. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

Now consider the parallel networks. Similar as for the fully-connected networks, the parallel networks correspond better to the physics when the network has more parameters. Again, the single hidden layer options are not performing well. An interesting observation here is that all the networks with more than 200 parameters have a very comparable performance that is very consistent in their adherence to the physics and yield superior results for the fuel objective. These architectures are the most favored option out of all considered. A bundle with parallel [1, 10, 10, 10, 10, 1] architectures has been used in the baseline configuration, but the other options with more than 200 parameters could have been selected equally well. The most favorable of the fully-connected neural networks is the one with the [1, 20, 20, 20, 20, 20, 7] architecture. For comparison, the metrics of this network are compared to the metrics of the architecture used in the baseline configuration in table C.1. The consistency and slightly superior values of dx and dv for the parallel network was prioritized in this work. However, fully-connected neural networks are nearly equally capable of finding solutions to transfer problems and they can do so much

more efficiently, making them an attractive option worthwhile for further investigation.

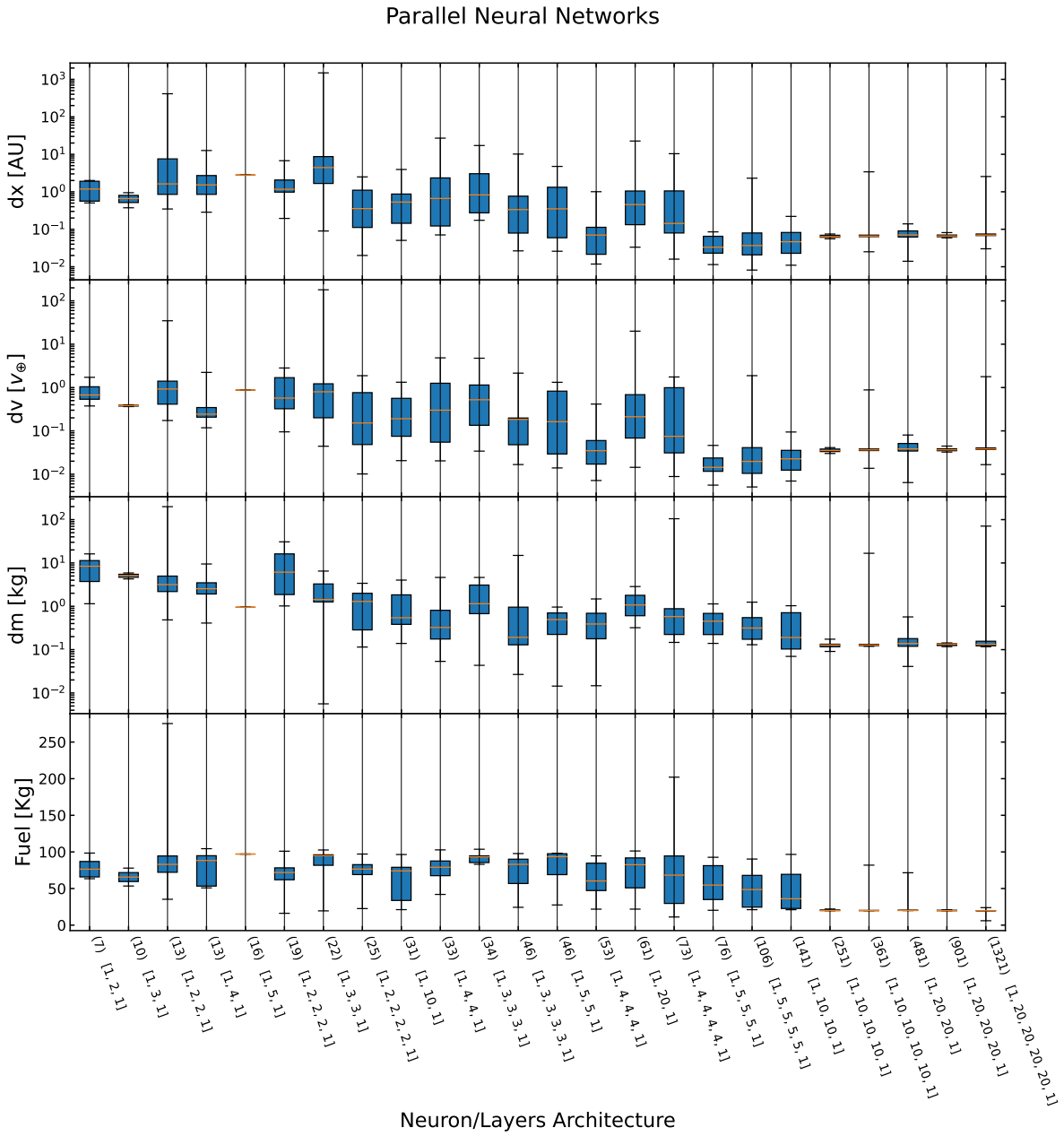


Figure C.5: Grid search over parallel bundles of neural network architectures. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

	Fully-connected [1, 20, 20, 20, 20, 7]	Bundle [1, 10, 10, 10, 1]
Median dx [AU]	0.076	0.064
Median dv [$v_\oplus$]	0.041	0.035
Median dm [kg]	0.081	0.129
Median propellant mass m_p [kg]	19.93	19.96
Median CPU time [s]	169	619

Table C.1: Comparison between the champion of the fully connected architectures and the champion of the bundled architectures.

C.3. Grid search: Network Activation

The baseline configuration from the paper was making use of the sine activation function. Several alternatives (relu, sigmoid, tanh) have been tested on the 2-revolution optimal-fuel Earth-Mars transfer and rendezvous problem in a grid search with 40 repetitions per activation function. Figure C.6 compares the metrics dx, dv, dm and the propellant mass for this grid search. It is evident that the relu is not able to capture the dynamics encountered in the transfer problem and the sigmoid performs quite unreliably. Both tanh and sine are suitable options and there is no particular reason that the sine has been chosen instead of the tanh function in this work.

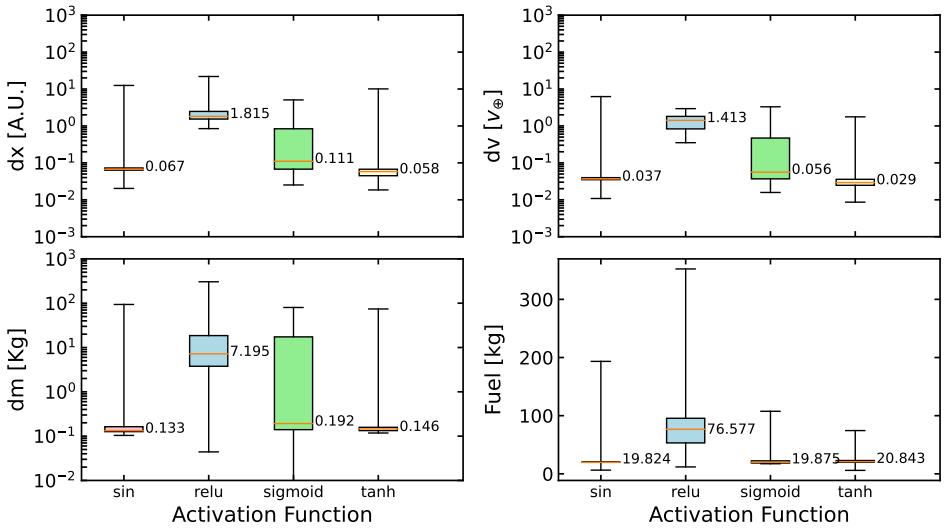


Figure C.6: Grid search over activation functions. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

C.4. Grid search: Constraint steepness parameter

The steepness parameter a that is present in the constraint equation was set to 10 in the baseline configuration. A range of values between 1 and 50 has been tested on the 2-revolution optimal-fuel Earth-Mars transfer and rendezvous problem in a grid search with 40 repetitions per value of a. Figure C.7 compares the metrics dx, dv, dm and the propellant mass. It can be quite confidently concluded that the PCNN's ability to describe solutions to the transfer problem is not sensitive to the choice of steepness parameter as long as it is not too large. Smaller than 20 that is.

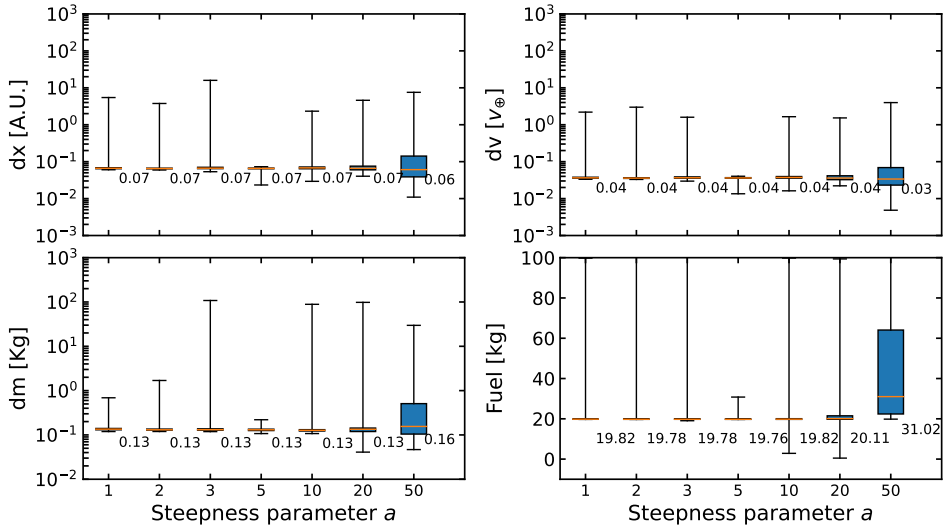


Figure C.7: Grid search over steepness parameter a. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

C.5. Grid search: Training sample

The batch size N of training points per epoch was set to 200 in the baseline configuration. A range of values between 10 and 1000 has been tested on the 2-revolution optimal-fuel Earth-Mars transfer and rendezvous problem in a grid search with 40 repetitions per batch size N. Figure C.7 compares the metrics dx, dv, dm and the propellant mass. Small batch size is not favorable because the network hasn't quite learned to satisfy the dynamical constraints. From batch size 200 and larger, the metrics barely improve with increasing batch size. It has been demonstrated in the paper that the values for dx and dv originate from the battle between the dynamic and objective loss terms and are not due to the network's inability to capture the dynamics. This balance (dictated by the objective loss weight ω_o) is deliberately chosen to converge to optimal solutions. Due to this competition, having more collocation points per batch does not result in a more accurate solution. The median CPU time to train a network with $N = 200$ training point batch size was $t_{\text{CPU}} = 911.4$ s, for $N = 500$ it was $t_{\text{CPU}} = 1035.3$ s and for $N = 1000$ it was $t_{\text{CPU}} = 1443.8$ s. The trade-off was made to choose for the more efficient option of $N = 200$ considering that the other options have nearly identical metrics.

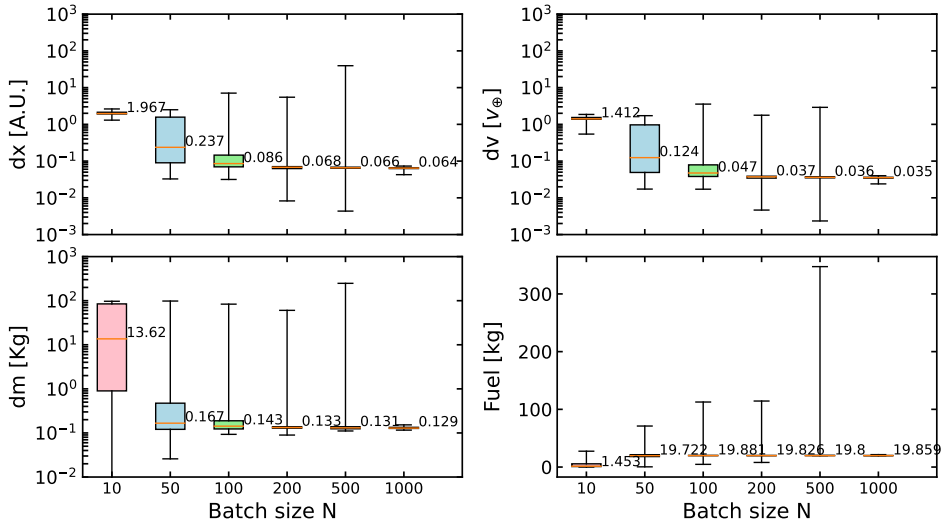


Figure C.8: Grid search over training batch size N. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

D

Extended Conclusions and Recommendations

The paper has highlighted the most impactful and direct recommendations for future work and provided conclusions that have been drawn from the results. In this chapter, the original research questions developed to guide this master thesis are reiterated and explicitly answered. Furthermore, recommendations are listed, which are split into short-term and long-term.

D.1. Answering the research questions

Q1 ***How can a Physics-Informed Neural Network (PINN) be configured to aid spacecraft trajectory optimization?***

This thesis started out with the broad goal to assess the current PINNs proposed in literature and find new ways to use them in trajectory optimization. Originally one road was selected for exploration. That was the option of configuring the PINN as a method that acts analogously to a numerical integrator. Networks like this had been proposed, and they could be assessed on the two-body problem, something that had not been done before. Once the integration of the two-body problem by a PINN was implemented successfully, the idea for an extension of the method to solve optimal control problems by adding a final constraint, adding the control parameters as an output to the network and including the objective as a loss term was conceived. This opportunity provided a much better outlook to aid trajectory optimization, the goal of the research, compared to the poorly-performing PINN for numerical integration. Subsequently it was found that this had already been proposed (very) recently, but was only applied in the context of partial differential equations and without analytical constraints. The second sub question was constructed to shape a research in this type of PINN capable of solving optimal control problems in a direct approach. Low-thrust transfers were identified as an area within mission analysis that can benefit from this new type of method, because of its continuous control nature and the limitations in existing tools, specifically lack of multi-objective capability, difficulties in searching large design spaces and lack of efficiency.

Answer: A method to solve optimal control problems in a direct approach with a physics-informed neural network is developed and applied to optimal-fuel low-thrust transfer trajectory problems. The method excels at finding solutions in non-linear circumstances and has the potential of being developed into a framework that can perform multi-objective optimization and search large design spaces.

Q1.1 ***How can a PINN best be configured to act as an integrator of initial value problems in orbital mechanics and what is its performance?***

PINNs can be configured to solve initial value problems with analytically constrained initial conditions and this has been performed on the two-body problem in an early phase of this

research. An example of this is found in appendix B.1.2.1, where the solution of a circular low-earth orbit has been created by the PINN in order to verify the implementation of the dynamical model. No further investigation has been conducted into the performance in terms of accuracy or efficiency. No conclusions can be made based on this research other than that it is at least possible to find the solution of a circular orbit.

Q1.2 ***How can a PINN best be configured to act as a trajectory optimization algorithm and what is its performance?***

A configuration is established where an unsupervised physics-informed neural network produces the states and control as a function of time. It is trained by having a primary loss term reflect the physical model and a secondary loss term reflect the objective of the optimal control problem. An initial and final constraint is analytically enforced. Time of flight is kept constant. Carefully selected weights of the loss terms and tuning of learning rate schedule are key ingredients in successfully converging the neural networks. The networks are highly sensitive to initialization, but a failed initialization can be recognized early on. Accuracy can be traded for optimality (by balancing of the loss terms). It has been found that the method is capable of retrieving near-optimal solutions for a wide range of phase angles and times of flight of an Earth-Mars transfer and rendezvous problem. Moreover, it excels in finding solutions when the initial and final constraint imply a more non-linear solution. On the other hand, it is inefficient, with CPU times on the other of 100s of seconds.

Q1.2.1 *How are suitable network architectures and training procedures established?*
The method is not very sensitive to the architecture of the neural network. Both a bundle of network and a single fully-connected network are possible options. The network should be sufficiently large to capture the dynamics. Non-linear activation functions, like the sine or hyperbolic tangent, are required. On the other hand, the network is highly sensitive to the training procedure and initialization. In terms of training procedure, a learning rate schedule is established that performs quite consistently, yet not fully consistent. A unique feature in this schedule is the temporary increase of learning rate, dubbed a 'shake', that can guide the training in handling the competing loss terms and escape local minima. Failed initializations occur frequently, but can be recognized by high loss values early on in training and can thus be circumvented by re-initializing the network. Additional work is required in order to uncover what determines if an initialization is suitable or not.

Q1.2.2 *What reference frame/coordinates systems are successful?*
The work has taken place entirely in a 2D space. Polar coordinates with a velocity vector in a basis spanned by a unit vector pointing radially outward and an orthogonal unit vector pointing in the direction of the initial motion (tangential) have been extensively tested. Additionally, limited analysis on cartesian coordinates has been carried out. The networks strictly performed better in polar coordinates. Cartesian coordinates suffer from more local minima that are difficult to avoid.

Q1.2.3 *How can constraints be included in the PINN?*
In this work constraints are enforced analytically. This works fine as long as the constraint layer is a differentiable function. No testing has been performed on soft-constraint in the form of penalizing loss terms.

Q1.2.4 *How can control be included in the PINN?*
Control can be added as an output to the network that is subsequently included in the equations of motion that are established in the dynamics loss terms. A natural method for control parameter constraints is implemented by using activation functions that map the output to a value in the range [0, 1] and then multiplying with the maximum allowed value to scale the output.

Q1.2.5 *How can objectives be included in the PINN?*
Objectives can be included as an additional loss term that is carefully weighted with respect to the dynamical loss terms. For the example considered in this work, optimal-fuel, the fitness is calculated by estimating the integral over the control magnitude with a trapezoid rule. The collocation points are sampled in a perturbed equidistant time grid in order to fairly estimate this integral.

Q1.2.6 *How is the network performing for different scenarios?*
Because of the continuous nature of the control output, interplanetary low-thrust transfer trajectories were chosen as the situation to test the method on. Because of the constant time of flight nature of the method, an optimal-fuel objective was selected in the description of the optimal control problem. A great range of phase angles and times of flight for an Earth-Mars transfer and rendezvous have been optimized. Comparing the results to solutions acquired by a hodographic shaping method demonstrates that the proposed PINN is capable of finding near-optimal solutions. Specifically, a maximum improvement of 4.5 km s^{-1} and a median improvement of 0.55 km s^{-1} was observed. For constraints that imply a more non-linear solution, due to an awkward phase angle with respect to the time of flight, the PINN excels compared to the hodographic shaping method.

D.2. Short-term development

Some ideas for short-term development that attempt to tackle some of the more urgent problems and open questions are the following.

- **Try the method on different scenarios.** Primarily, the circular-restricted three problem is of specific interest, due to the presence of more nasty nonlinearities compared to the two-body problem. This will really push the method to its limits and might set it apart from other methods. Finding optimal low-thrust transfers in that setting is also of special interest due to the upcoming Lunar Gateway space station. Alternatively, ascent and descent trajectories are an interesting test case and the addition of perturbations, like solar radiation pressure or perturbing bodies, can certainly be looked into.
- **Combine Hodographic-Shaping with PCNN.** The PCNN and shape-based methods can complement each other by leveraging the PCNN to fine-tune an initial solution obtained through a shape-based approach. For example by using data points from a solution acquired from hodographic shaping as data input to the PCNN. The PCNN can then be trained in a supervised manner, refining the solution via the dynamics and objective loss terms. This way, the PCNN is guided into the correct direction earlier in training. Alternatively, it might be possible to combine them by having the internal network architecture of the PCNN be derived from the base functions of successful shape-based methods, like hodographic shaping.
- **Improve efficiency** Although it will most likely never be as quick as other methods, like shape-based optimization, some improvements might be possible. Adaptive sampling of collocation points, like proposed by Wu et al. [52], are an option to create more efficient training batches. Alternatively, adaptive weights might have a positive effect on the efficiency [47]. A more creative idea is to have more tailored network architectures per entry that might boost both efficiency and accuracy. For example, the time evolution of the angle θ is generally quite linear for the problem solved in this work and it might be more suitable to have that reflected by the layers, neurons and activation function for that entry (e.g. fewer layers and fewer neurons and a relu activation function).
- **The method should be made more reliable.** Specifically, the restarting mechanism is unusual and ideally temporary. A more fundamental understanding of the distinction between failed and non-failed initializations should be developed. This is a complicated issue due to the black box nature of the neural network. A way to start this, is with a comprehensive comparison of initialization methods or trying to find correlations between the values of the weights/biases and the convergence success.
- **Try an adaptive loss weight scheme.** Not only will this serve to more efficiently navigate the loss space, it also builds towards multi-objective optimization. Some algorithms are presented in the

literature [47], but it does not have to be difficult to create something new. For example, in the context of the work presented here, it might be a very good idea to add a few more iterations at the end with a smaller objective loss weight, in an attempt to better comply with the physical model and close the gap in dx and dv. This is the simplest form of a 'changing loss weight scheme'.

- **Try a time-objective.** This can for example be constructed by scaling the time input with a trainable parameter representing the open final time parameter. In other words, only sample inputs in the range $[0, 1]$ and multiply with a final time trainable parameter before insertion into the neural network. That same trainable parameter can then be defined as the objective.

D.3. Long-term development

Here, a broad view on a more long-term development of the method proposed by this master thesis will be expressed.

Generalization

Personally, I would very much like to know whether a universal configuration of the network and accompanying training procedure can be established that finds solutions to a great range of optimal control problems. For example, can the method also handle ascent and descent trajectories and problems within the three-body problem with the same network architecture, training procedure and coordinate system? Perhaps this becomes possible when making the networks really large and training them on Graphical Processing Units (GPU). It is also interesting to look more into weight sharing. Perhaps a set of weights and biases or a method to construct them can be designed that can act as initialization for a certain class of optimal control problems, like a fuel-optimal low-thrust transfer. This could potentially be a solution to the encountered failed initializations.

Multi-objective optimization

The PCNN method is inherently capable of performing multi-objective optimization, by adding objective loss terms. In the scope of this thesis, it has not been possible to address this feature. Difficulties will arise from competing loss terms, which has already been proven to be a complication when considering only two competing losses, dynamics and a single objective. However, if it is possible to navigate the loss landscape, a trade-off can be made by selection of the weight parameters in order to determine where to end up on a Pareto front. It could be imagined that in a very far developed state, the method can handle many objectives where the designer merely selects the weights by determining what objectives have priority over others, and the network converges to a balanced solution. If such a high-level tool can actually function properly, space engineers would be able to make quick-trade offs efficiently in the context of a concurrent design phase as they only insert the relative importance into the tool. Multi-objective capability is therefore an important next step in the assessment of this method.

Searching large design spaces

The network is capable of searching a large design space in the sense that it narrows down a completely free shape to a physically valid near-optimal solution of an optimal control problem. When low-thrust tools require the option to search large design spaces, it is meant more along the lines that tools should be able to search large launch windows, or many different gravity assist sequences. The current implementation of the PCNN is not suitable for searching many combinations in a combinatorial problem through a brute force approach, due to the inadequate efficiency. However, one could potentially automate the search over launch opportunities by keeping the departure time and arrival time open as trainable parameters and connecting an ephemeris to obtain states of target bodies during training at these variable times, thus theoretically finding a single optimal solution within the entire launch opportunity space. If navigated successfully, which is again anticipated to be very complex, this approach can completely circumvent the double loop required to search launch windows and time of flight options. This could be combined with the possibility of doing multi-objective optimization with a time of flight and ΔV objective. Moreover, one could consider a predefined series of gravity assists at planetary bodies. The PINN description of the solution can be analytically constrained to perform the gravity assists at these bodies where the times of closest approach are left open as trainable parameters. As a consequence, only a loop over sequences has to be performed and no inner loops on launch dates,

arrival dates and time of flight of individual legs has to be carried out, greatly increasing the design space being searched. Description of such a trajectory probably requires a large neural network. In order to move towards such implementations, training of much larger network architectures by GPU can be explored.

These types of extensions demonstrate how the PCNN can be extended in a modular fashion towards a method that finds entire mission scenarios satisfying sets of requirements instead of optimizing single building blocks of a mission, like an optimal-fuel transfer.

Final remarks

This closing section has provided a vision on what the PCNN can be developed into in a far future. What most ideas to extend the PCNN method presented here have in common, is that they rely on the advancements of navigating intricate loss landscapes in physics-informed neural networks. Specifically, competing loss terms and the existence of many local minima are major obstacles. Overcoming these issues should be researched in the context of astrodynamics-related implementations of PINNs. At the same time, the flood of literature being produced surrounding PINNs should be closely observed, in order to identify training procedures, loss weight algorithms or other innovations that make PINN training easier. If you are reading this after great advancements have been published in the field of PINNs, please consider looking into using them to extend this work, I promise that it will be an incredible journey.

References

[1] Bassel Al Homssi et al. “Next generation mega satellite networks for access equality: Opportunities, challenges, and performance”. In: *IEEE Communications Magazine* 60.4 (2022), pp. 18–24.

[2] Harry Jones. “The recent large reduction in space launch cost”. In: 48th International Conference on Environmental Systems. 2018.

[3] Wayne A Shiroma et al. “CubeSats: A bright future for nanosatellites”. In: *Central European Journal of Engineering* 1 (2011), pp. 9–15.

[4] Massimo Bandecchi et al. “The ESA/ESTEC concurrent design facility”. In: *Proceedings of EUSEC* 9 (2000), p. 2000.

[5] Joseph A Starek et al. “Spacecraft autonomy challenges for next-generation space missions”. In: *Advances in control system technology for aerospace applications*. Springer, 2015, pp. 1–48.

[6] Tommaso Ghidini. “Materials for space exploration and settlement”. In: *Nature materials* 17.10 (2018), pp. 846–850.

[7] Stéphane Mazouffre. “Electric propulsion for satellites and spacecraft: established technologies and novel approaches”. In: *Plasma Sources Science and Technology* 25.3 (2016), p. 033002.

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

[9] Charles E Garner et al. “Ion propulsion: an enabling technology for the dawn mission”. In: *AAS/AIAA Spaceflight Mechanics Meeting*. AAS 13-342. 2013.

[10] CT Russell and CA Raymond. “The dawn mission to Vesta and Ceres”. In: *The dawn mission to minor planets 4 vesta and 1 ceres*. Springer, 2011, pp. 3–23.

[11] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

[12] Anil V Rao. “A survey of numerical methods for optimal control”. In: *Advances in the astronautical Sciences* 135.1 (2009), pp. 497–528.

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

[14] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[15] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[16] David Silver et al. “Mastering the game of go without human knowledge”. In: *nature* 550.7676 (2017), pp. 354–359.

[17] Feng-Lei Fan et al. “On interpretability of artificial neural networks: A survey”. In: *IEEE Transactions on Radiation and Plasma Medical Sciences* 5.6 (2021), pp. 741–760.

[18] Jing Wei et al. “Machine learning in materials science”. In: *InfoMat* 1.3 (2019), pp. 338–358.

[19] Benjamin A Toms, Elizabeth A Barnes, and Imme Ebert-Uphoff. “Physically interpretable neural networks for the geosciences: Applications to earth system variability”. In: *Journal of Advances in Modeling Earth Systems* 12.9 (2020), e2019MS002002.

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

[21] Stefano Silvestrini and Michèle Lavagna. “Deep Learning and Artificial Neural Networks for Spacecraft Dynamics, Navigation and Control”. In: *Drones* 6.10 (2022), p. 270.

[22] Carlos Sánchez-Sánchez, Dario Izzo, and Daniel Hennes. “Learning the optimal state-feedback using deep networks”. In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2016, pp. 1–8.

[23] Carlos Sánchez-Sánchez and Dario Izzo. "Real-time optimal control via deep neural networks: study on landing problems". In: *Journal of Guidance, Control, and Dynamics* 41.5 (2018), pp. 1122–1135.

[24] Dario Izzo and Ekin Öztürk. "Real-time guidance for low-thrust transfers using deep neural networks". In: *Journal of Guidance, Control, and Dynamics* 44.2 (2021), pp. 315–327.

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

[26] Daniel Hennes, Dario Izzo, and Damon Landau. "Fast approximators for optimal low-thrust hops between main belt asteroids". In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2016, pp. 1–7.

[27] Alessio Mereta, Dario Izzo, and Alexander Wittig. "Machine learning of optimal low-thrust transfers between near-earth objects". In: *International Conference on Hybrid Artificial Intelligence Systems*. Springer. 2017, pp. 543–553.

[28] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[29] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations". In: *IEEE transactions on neural networks* 9.5 (1998), pp. 987–1000.

[30] George Em Karniadakis et al. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440.

[31] Salvatore Cuomo et al. "Scientific machine learning through physics–informed neural networks: Where we are and what's next". In: *Journal of Scientific Computing* 92.3 (2022), p. 88.

[32] Enrico Schiassi et al. "Physics-informed neural networks for optimal planar orbit transfers". In: *Journal of Spacecraft and Rockets* 59.3 (2022), pp. 834–849.

[33] John Martin and Hanspeter Schaub. "Physics-informed neural networks for gravity field modeling of the Earth and Moon". In: *Celestial Mechanics and Dynamical Astronomy* 134.2 (2022), p. 13.

[34] John Martin and Hanspeter Schaub. "Physics-informed neural networks for gravity field modeling of small bodies". In: *Celestial Mechanics and Dynamical Astronomy* 134.5 (2022), p. 46.

[35] Andrea Scorsoglio, Luca Ghilardi, and Roberto Furfaro. "A Physic-Informed Neural Network Approach to Orbit Determination". In: *The Journal of the Astronautical Sciences* 70.4 (2023), pp. 1–30.

[36] Saviz Mowlavi and Saleh Nabi. "Optimal control of PDEs using physics-informed neural networks". In: *Journal of Computational Physics* 473 (2023), p. 111731.

[37] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[38] Johannes Lederer. "Activation functions in artificial neural networks: A systematic overview". In: *arXiv preprint arXiv:2101.09957* (2021).

[39] Marios Mattheakis et al. "Hamiltonian neural networks for solving equations of motion". In: *Physical Review E* 105.6 (2022), p. 065305.

[40] Jiuxiang Gu et al. "Recent advances in convolutional neural networks". In: *Pattern recognition* 77 (2018), pp. 354–377.

[41] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. "Recurrent neural networks for time series forecasting: Current status and future directions". In: *International Journal of Forecasting* 37.1 (2021), pp. 388–427.

[42] Atilim Gunes Baydin et al. "Automatic differentiation in machine learning: a survey". In: *Journal of Marchine Learning Research* 18 (2018), pp. 1–43.

[43] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[44] Pengzhan Jin et al. “SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems”. In: *Neural Networks* 132 (2020), pp. 166–179.

[45] Michael M Bronstein et al. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[46] Aditi Krishnapriyan et al. “Characterizing possible failure modes in physics-informed neural networks”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26548–26560.

[47] Rafael Bischof and Michael Kraus. “Multi-objective loss balancing for physics-informed deep learning”. In: *arXiv preprint arXiv:2110.09813* (2021).

[48] William M Folkner et al. “The planetary and lunar ephemerides DE430 and DE431”. In: *Interplanetary Network Progress Report* 196.1 (2014), pp. 42–196.

[49] Lu Lu et al. “DeepXDE: A deep learning library for solving differential equations”. In: *SIAM Review* 63.1 (2021), pp. 208–228. DOI: 10.1137/19M1274067.

[50] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[51] Leon Stubbig. “Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories”. In: (2019).

[52] Chenxi Wu et al. “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115671.