An unsupervised physics-informed neural network for finding optimal low-thrust transfer trajectories with the direct method

Goldman, T.C.; Cowan, K.J.

Publication date
2024

Document Version
Accepted author manuscript

Published in
Advances in the Astronautical Sciences

Citation (APA)
Goldman, T. C., & Cowan, K. J. (2024). An unsupervised physics-informed neural network for finding optimal low-thrust transfer trajectories with the direct method. In J. A. Christian, A. Mashiku, P. Valerino, & P. Mehta (Eds.), *Advances in the Astronautical Sciences: Annual Conference procedings* Article AAS 24-224 (Advances in the Astronautical Sciences).

(Preprint) AAS 24-224

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

Thomas Goldman[*], Kevin Cowan[†]

An unsupervised Physics-Informed Neural Network (PINN) 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 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 PINN shows promise as a preliminary design tool for trajectory optimization in challenging dynamical conditions.

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 attractive 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] Unfortunately, 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. 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.[4] A survey on the tools in academia and industry for designing low-thrust 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.[6,7] PINN include physical laws in their architecture or learning process in order to improve the modeling of physical systems. Most commonly, the loss is regularized with a physics term consisting of the Partial Differential Equation (PDE)

[*]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.

that governs the physics of the system. In supervised fashion, PINNs can be configured to perform a physics-aided regression on data or solve inverse problems. This is particularly useful for systems with low availability of data. PINN-based representations exhibit greater confidence in generalizing beyond the observed domain and simultaneously filter errors in the data. In unsupervised form, PINN can solve initial value problems, that include a PDE or 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. Although the majority of research on PINNs is focusing on PDEs from a computational science perspective,[8,9] 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[10] as well as that of smaller bodies.[11] Supervised PINNs have also been constructed to solve orbit determination problems.[12] 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.[13] 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.[14] 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.[15] 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 preliminary design of low-thrust transfers.

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.[15] extended with analytical constraints inspired by Mattheakis et al.[16] 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-INFORMED 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.

$$\begin{aligned}
\text{Minimize} \quad & J = \Phi(\mathbf{z}_0, \mathbf{z}_f, t_0, t_f) + \int_{t_0}^{t_f} L(\mathbf{z}(t), \mathbf{u}(t))\, dt && (1)\\
\text{Subject to} \quad & \dot{\mathbf{z}} = f(\mathbf{z}, \mathbf{u}, t) \quad \mathbf{z} \in \mathbb{R}^N, \mathbf{u} \in \mathbb{R}^Q, t \in [t_0, t_f] && (2)\\
& \mathbf{z}(t_0) = \mathbf{z}_0 && (3)\\
& \mathbf{z}(t_f) = \mathbf{z}_f && (4)\\
& && (5)
\end{aligned}$$

A neural network $\mathcal{N}_{\boldsymbol{p}}(t)$ 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 unconstrained outcomes $\mathbf{z}' \in \mathbb{R}^N$ and $\mathbf{u}' \in \mathbb{R}^Q$. 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 subset of PINN called *physics-constrained neural network* (PCNN).[9]

$$\begin{bmatrix} \mathbf{z}' \\ \mathbf{u}' \end{bmatrix} = \mathcal{N}_{\boldsymbol{p}}(t) : \mathbb{R} \mapsto \mathbb{R}^{N+Q} \tag{6}$$

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} h(\mathbf{z}', t) \\ g(\mathbf{u}', t) \end{bmatrix} \tag{7}$$

Mattheakis et al. use a time-dependent parametric equation with exponential functions to satisfy an initial condition in their PINN.[16] Using similar expressions, this work enforces an initial *and* final condition on the state, see Equation 8.

$$\begin{aligned} \mathbf{z} = h(\mathbf{z}', t) = \quad & e^{-a(t-t_0)}\,\mathbf{z}_0 \\ & + \left[1 - e^{-a(t-t_0)} - e^{a(t-t_f)}\right]\mathbf{z}' \\ & + e^{a(t-t_f)}\,\mathbf{z}_f \end{aligned} \tag{8}$$

where a is a parameter that controls the steepness with which the parametric equations vanish and rise. Figure 1 graphically depicts the various constraint terms in Equation 8, 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.

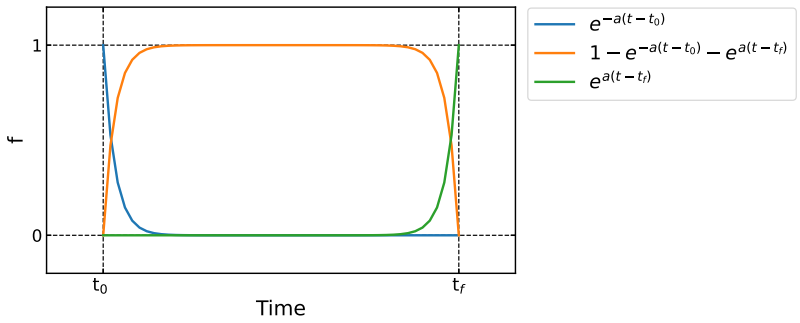

Figure 1. Graphical illustration of the terms in constraint Equation 8 with $a = 10$.

After initialization, the solution as described by the neural network, although satisfying the boundary conditions, is random and nonsensical. In order to solve the OCP, the network has to be trained. The loss to be optimized during training consists of several terms. The first term, $\mathcal{L}_d$ (Equation 10), 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 $\dot{z}_i = f_i(\mathbf{z}, \mathbf{u}, t)$. 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*.

$$\mathcal{L}_i = \frac{1}{M}\sum_{j}^{M}\left[\dot{\boldsymbol{z}}_i(t_j, \boldsymbol{p}) - f_i\big(\boldsymbol{z}(t_j, \boldsymbol{p}), \boldsymbol{u}(t_j, \boldsymbol{p}), t_j\big)\right]^2 \tag{9}$$

$$\mathcal{L}_d = \sum_{i}^{N}\mathcal{L}_i \tag{10}$$

where $j = 1, 2, ..., M$ are the collocation points in a training batch. An additional loss term is added that represents the objective of the optimal control problem $\mathcal{L}_o$ (Equation 11), a procedure proposed by

Mowlavi et al.[15] 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.

$$\mathcal{L}_o = J = \Phi(\mathbf{z}_0, \mathbf{z}_f, t_0, t_f) + \int_{t_0}^{t_f} L(\mathbf{z}(t), \mathbf{u}(t))dt \tag{11}$$

$$\mathcal{L} = \omega_d \mathcal{L}_d + \omega_o \mathcal{L}_o \tag{12}$$

During training, the integral in J has to be numerically estimated from the batch of training data at each training epoch. A training batch $T = \{t_0, t_1, ..., t_M\}$ is quasi-randomly chosen for each training epoch to mimic a continuous sampling space spanning the entire domain of t. 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.[16] The set of times sampled as a training batch is defined as T,

$$T = \left\{ \mathcal{G}(\mu = t_0 + n\Delta t, \sigma = 0.2\Delta t) \,|\, n \in \{0, 1, ..., M\} \right\} \tag{13}$$

$$\Delta t = \frac{t_f - t_0}{M} \tag{14}$$

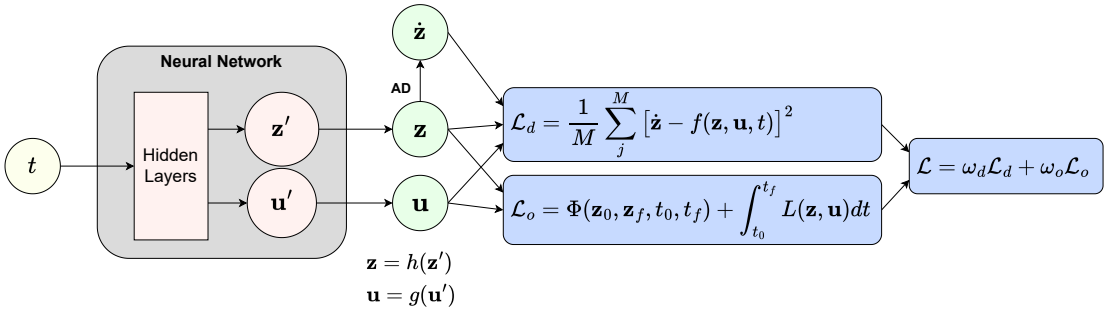

Figure 2. Overview of the Physics-Informed Neural Network to solve two-boundary-value optimal control problems. AD refers to automatic differentiation.

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.

PINN FOR FUEL-OPTIMAL LOW-THRUST TRANSFER

The proposed PINN design to solve continuous optimal control problems will be implemented for finding fuel-optimal 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 PINN 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 magnitude 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 $\hat{\mathbf{e}}_r$ pointing in the outward radial direction and a tangential unit vector $\hat{\mathbf{e}}_\theta$, perpendicular to the radial unit vector, pointing in the direction of increasing polar angle. The velocity is projected on the basis $\{\hat{\mathbf{e}}_r, \hat{\mathbf{e}}_\theta\}$, its entries are then $v_r = \mathbf{v} \cdot \hat{\mathbf{e}}_r$ and $v_\theta = \mathbf{v} \cdot \hat{\mathbf{e}}_\theta$. The thrust will be parameterized in terms of the thrust magnitude u_T and thrust angle u_ϕ, which measures the angle between the tangential unit vector and the thrust vector. See figure 3 for an graphical illustration of the reference frame and coordinates.

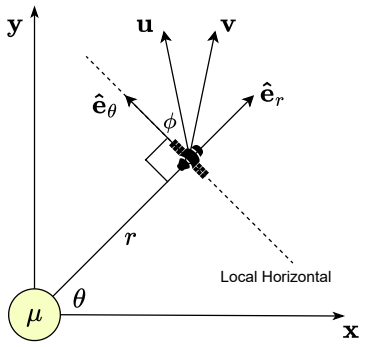

Figure 3. Reference frame and coordinates used to model the transfer trajectory.

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[13]

$$\dot{r} = v_r \tag{15}$$

$$\dot{\theta} = \frac{v_\theta}{r} \tag{16}$$

$$\dot{v_r} = \frac{v_\theta^2}{r} - \frac{\mu}{r^2} + \frac{u_T \sin u_\phi}{m} \tag{17}$$

$$\dot{v_\theta} = -\frac{v_r v_\theta}{r} + \frac{u_T \cos u_\phi}{m} \tag{18}$$

$$\dot{m} = \frac{u_T}{I_{sp} g_0} \tag{19}$$

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.

$$J = m_p = \frac{1}{I_{sp} g_0} \int_{t_0}^{t_f} u_T(t)\, dt \tag{20}$$

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 PINN method

A physics-informed 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, $\mathcal{N}_{\mathbf{p}}(t) = \begin{bmatrix} \mathbf{x}' & m' & \mathbf{u}' \end{bmatrix}^T \in \mathbb{R}^7$, where $\mathbf{x}' = \begin{bmatrix} r' & \theta' & v_r' & v_\theta' \end{bmatrix}^T$ and $\mathbf{u}' = \begin{bmatrix} u_\phi' & u_T' \end{bmatrix}^T$. 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 $\mathbf{u}(t)$.

$$\begin{bmatrix} \boldsymbol{x} \\ m \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} h_x(\mathbf{x}', t) \\ h_m(m', t) \\ g(\mathbf{u}') \end{bmatrix} \tag{21}$$

To adhere to the initial and final state constraint, the function h_x takes the shape of Equation 8, 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

$$m = h_m(\mathcal{N}_m, t) = m_0 - (1 - e^{-a(t-t_0)})\ m_0\ \text{sigmoid}(\mathcal{N}_m) \tag{22}$$

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

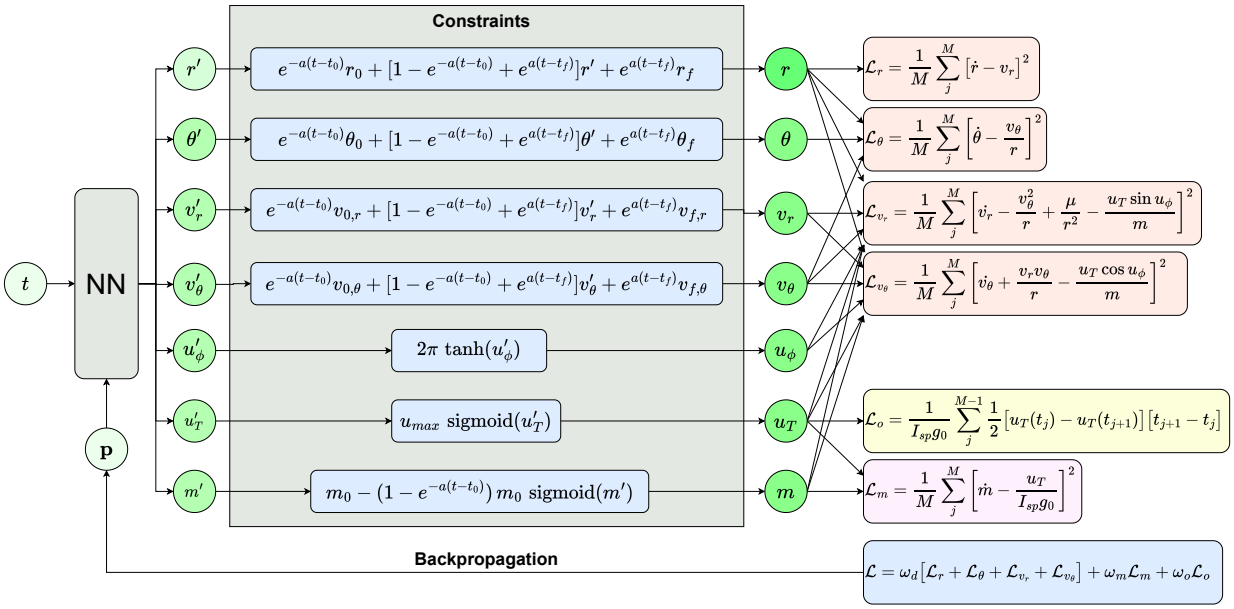

Figure 4. Overview of the Physics-Informed Neural Network implemented for solving a planar optimal-fuel low-thrust transfer trajectory problem.

The control entries u_ϕ' and u_T' are constrained via

$$\boldsymbol{u} = \begin{bmatrix} u_\phi \\ u_T \end{bmatrix} = g(\mathbf{u}') = \begin{bmatrix} 2\pi \ \tanh(u'_\phi) \\ u_{max} \ \text{sigmoid}(u'_T) \end{bmatrix} \tag{23}$$

The sigmoid activation followed by a multiplication with u_{max} makes sure that u_T is constrained between $[0, u_{max}]$. Similarly, u_ϕ is constrained 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 15-18 and a mass loss term, representing Equation 19 which is separately weighted. The objective, Equation 20, will be estimated by means of a trapezoid rule, using the training batch $\{t_1, ..., t_m\}$ at each epoch, which are sampled as a perturbed equidistant grid (Equation 13), and the corresponding outputs of the thrust force magnitude $\{u_T(t_0), ..., u_T(t_M)\}$.

$$\mathcal{L}_o = \frac{1}{I_{sp} g_0} \sum_{j}^{M-1} \frac{1}{2} \big[u_T(t_j) - u_T(t_{j+1})\big] \big[t_{j+1} - t_j\big] \tag{24}$$

$$\mathcal{L} = \omega_d [\mathcal{L}_r + \mathcal{L}_\theta + \mathcal{L}_{v_r} + \mathcal{L}_{v_\theta}] + \omega_m \mathcal{L}_m + \omega_o \mathcal{L}_o \tag{25}$$

The entire network architecture with all constraints and loss terms is schematically presented in Figure 4.

Verification Metrics

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 26), the final euclidean velocity distance dv (Equation 27) and difference in final mass dm (Equation 28).

$$dx = ||\mathbf{x}(t_f) - \mathbf{x}_v(t_f)|| \tag{26}$$

$$dv = ||\mathbf{v}(t_f) - \mathbf{v}_v(t_f)|| \tag{27}$$

$$dm = m(t_f) - m_v(t_f) \tag{28}$$

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 report 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

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 (Parameter in equation 8)	10

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

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.

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.[17] 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[18] python module with TensorFlow[19] as the backend. Numerical integrations of control profiles for verification were executed in the TU Delft Astrodynamics Toolkit (TUDAT).[20] The training of neural networks in large quantities has been performed in a paralleled configuration on CPU nodes* of the DelftBLue supercomputer.[21]

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 $\frac{1\text{AU}}{v_\oplus}$, 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 5 and the corresponding loss evolution in Figure 6. 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.

*Equipped with Intel XEON Gold E5-6226R processors

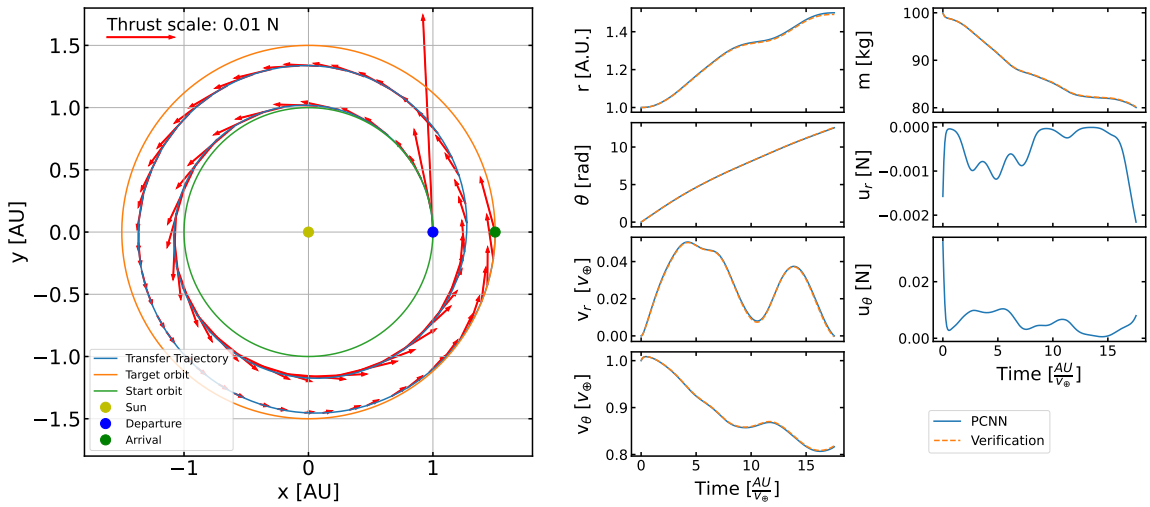

Figure 5. A near optimal-fuel solution for an Earth-Mars low-thrust transfer as acquired by the baseline configuration of the PINN. The verification metrics are $dx = 0.073$ AU, $dv = 0.040$ $\mathbf{v}_\oplus$ and $dm = -0.12$ kg. The propellant mass is $m_p = 19.84$ kg.

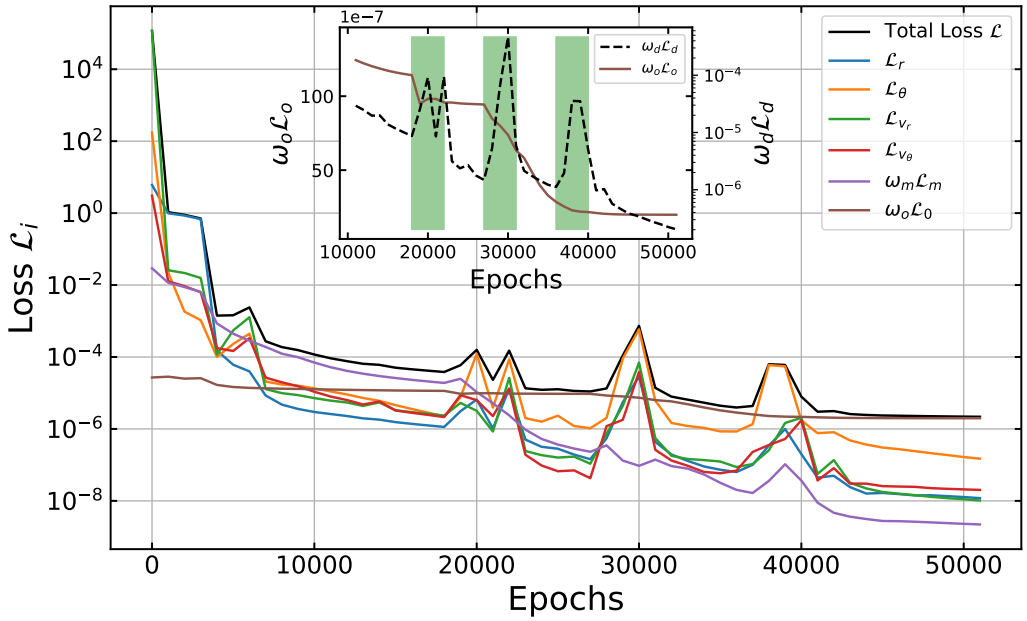

Figure 6. Training loss evolution for the solution presented in Figure 5. The interior panel displays the objective loss (brown) on a linear scale and the dynamics loss (dashed black) on a log scale from epoch 10000 onwards where the green shaded regions indicate a shaking phase.

	Schedule [learning rate, epochs]	Description
Schedule 1	$[10^{-2}$, 3k]* $\rightarrow$ $[10^{-3}$, 5k] $\rightarrow$ $[10^{-4}$, 10k] $\rightarrow$ $[5 \times 10^{-3}$, 4k] $\rightarrow$ $[10^{-4}$, 5k] $\rightarrow$ $[5 \times 10^{-3}$, 4k] $\rightarrow$ $[10^{-4}$, 5k] $\rightarrow$ $[5 \times 10^{-3}$, 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] $\rightarrow$ $[5 \times 10^{-3}$, 4k] $\rightarrow$ $[10^{-4}$, 5k] $\rightarrow$ $[5 \times 10^{-3}$, 4k] $\rightarrow$ $[10^{-4}$, 5k] $\rightarrow$ $[5 \times 10^{-3}$, 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.

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 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 7. 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 8. 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

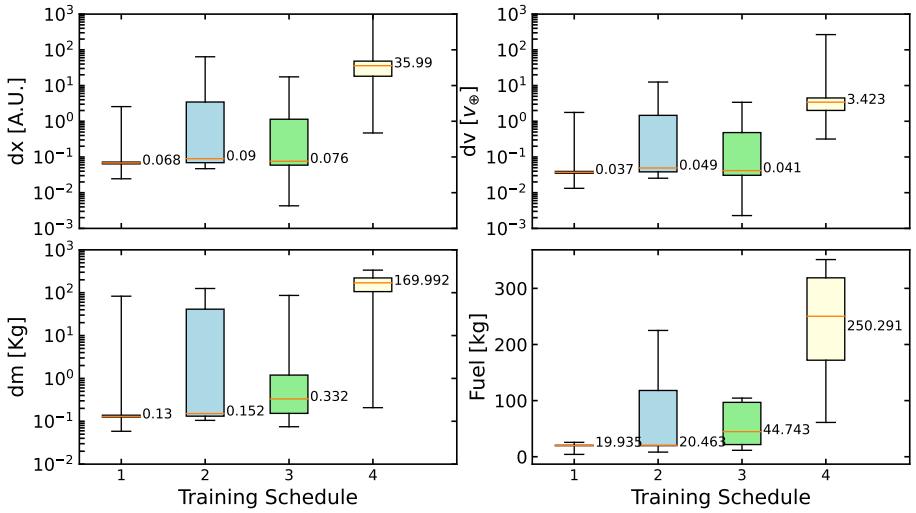

Figure 7. Comparison of learning training schedules as defined in Table 2. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

Schedule 1, both of these solutions have $dx > 2$ AU and $dv > 1$ $v_\oplus$ 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.

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 6. 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 value of the fuel objective and the smaller spread of outcomes for all metrics in Figure 7 of Schedule 1 compared to Schedule 3 indicates that shaking is successful in finding a global minimum not merely in the example of

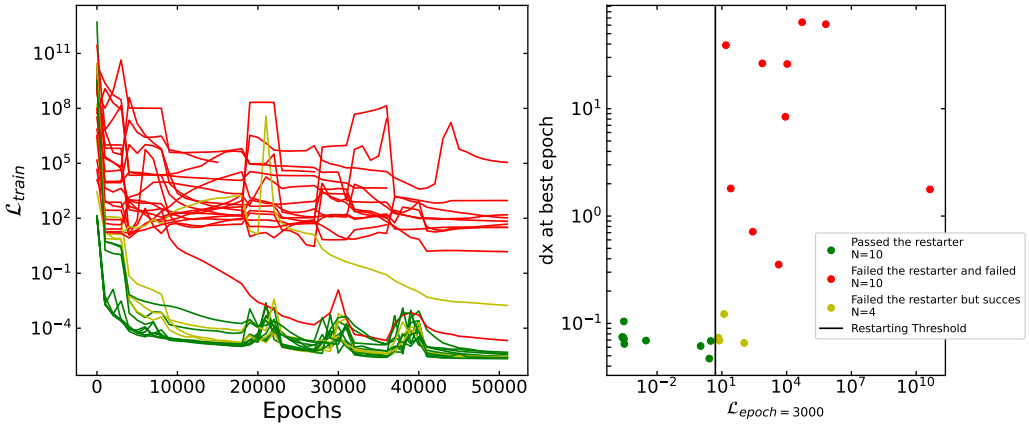

Figure 8. (*Left*) The training loss evolution of the cases trained with Schedule 2, which does not restart. (*Right*) The total test loss at epoch 3000 versus the dx of the solution after training ended. The color code in the legend applies to both panels

Figure 6 but for the vast majority of the cases.

In conclusion, the loss landscape is quite unique and capricious, but with some unusual tools like restarting and shaking, it can be successfully navigated when solving an optimal-fuel transfer problem in the two-body problem.

The loss weights

The total loss in Equation 12 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 9. 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 9. 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 9.

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 $\Delta\tilde{\mathbf{x}}$.

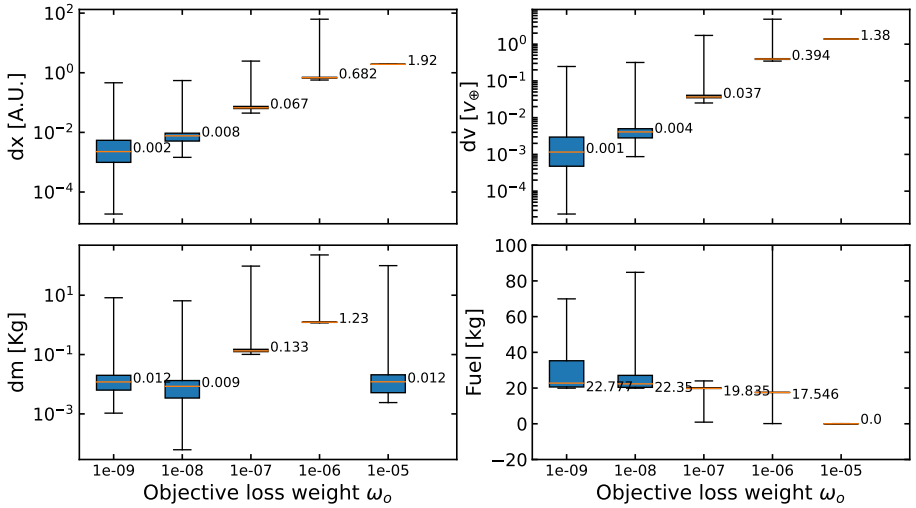

Figure 9. Grid search over the objective weight ω_o. In this box plot the orange lines indicate the median value, the boxes indicate the 1st and 3th quartile and the whiskers indicate the minimum and maximum.

$$S(t) = \frac{\delta \mathbf{x}(t)}{\delta \mathbf{a}} \tag{29}$$

$$\Delta \tilde{\mathbf{x}}(t) = \mathbf{S}(t) \Delta \mathbf{a} \tag{30}$$

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 constrained to end. Inverting the sensitivity matrix allows to solve for the empirical accelerations that closes this gap.

$$\Delta \mathbf{a} = \mathbf{S}^{-1} \Delta \tilde{\mathbf{x}}(t_f) \tag{31}$$

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

$$\Delta V = ||\mathbf{a}||(t_f - t_0) \tag{32}$$

$$\tilde{m}_p = m_0 \left[1 - \exp \frac{\Delta V}{I_{sp} g_0} \right] \tag{33}$$

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 9 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

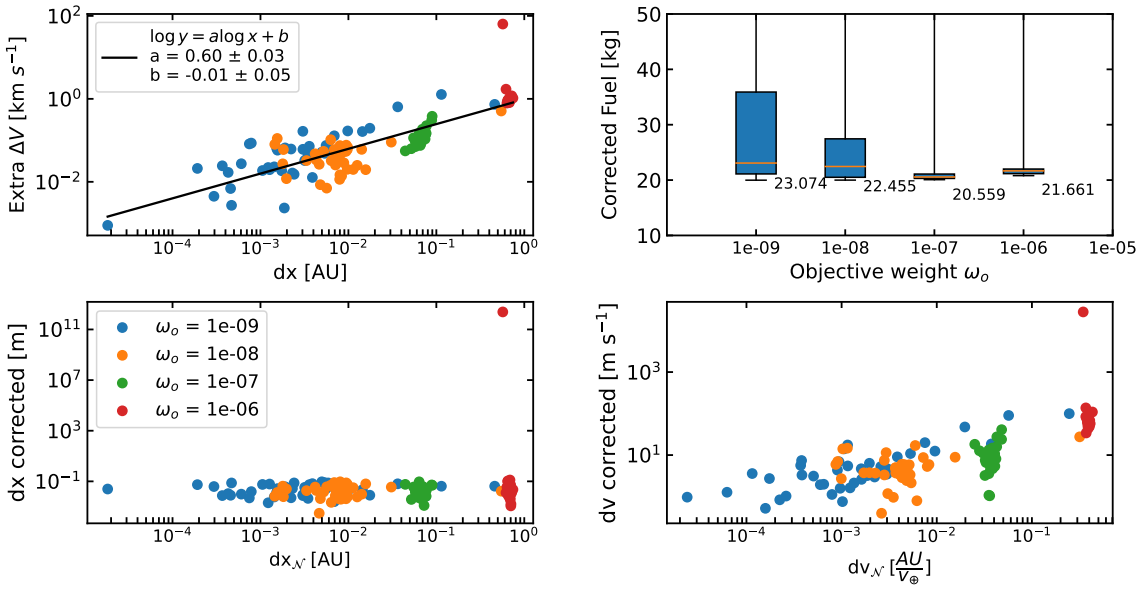

Figure 10. Refinement of dx and dv for the cases in the weight comparison (Figure 9) except for those with $\omega_o = 10^{-5}$. *Upper left*: the relation between the neural networks mismatch in position and the ΔV required to correct for it. *Upper right*: the distributions of total mass after including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

$\omega_0 = 10^{-5}$ is not considered as an option, because the propellant mass is consistently zero, implicating that 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 10. 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 10 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 10 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 insufficient 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 5. In order to test the PINN 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 PINN 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 PINN 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 PINN 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.[22] The PINN 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[23] 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 10. 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.[22] Optimization of the degrees of freedom in order to minimize the ΔV is carried out by the Nelder-Mead optimization scheme[24] implemented in the Pygmo module[25] in combination with a hodographic shaping tool implemented by Stubbig et al.[26] All launch opportunities are solved for 0, 1, 2 revolutions and the optimal amount is chosen. Options filtered from the PINN solutions are also filtered from the HS solutions.

The optimal ΔV gathered from both methods are depicted in the launch opportunity plots in Figure 11 where they are plotted as a function of departure and arrival date. For the PINN method, the position and velocity error metrics are plotted in Figure 12. 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 PINN 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

*notation adapted from Gondelach et al.[22]

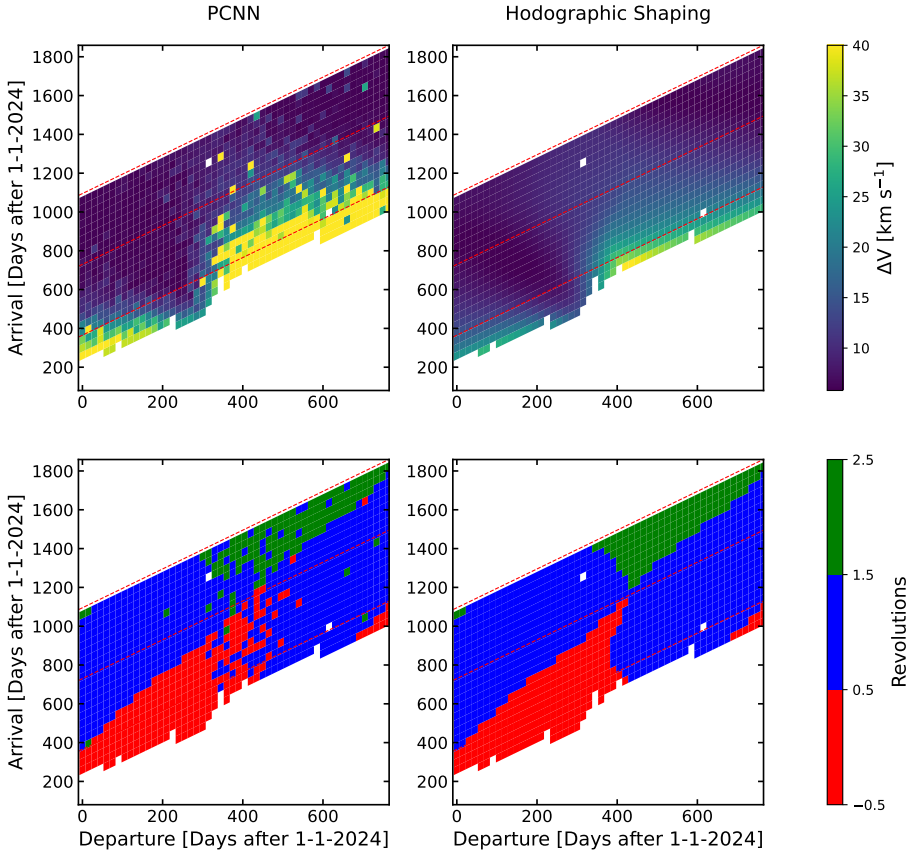

Figure 11. ***Upper row*: ΔV as a function of departure and arrival date for optimal-fuel Earth-Mars rendezvous transfers. *Lower row*: selected amount of revolutions. *Left column* is acquired with the PINN proposed in this work and the *right column* is found by a hodographic shaping method.**

transfers. Furthermore, as the time of flight increases, the ΔV becomes less sensitive to the departure date, as it is capable of finding efficient solutions that include coasting phases. 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 13. 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 14, the residuals in ΔV objectives between the methods are quantified for each launch opportunity. A striking observation is that the HS method outperforms the PINN 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 PINN 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 PINN 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 suffer less from 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 14 shows only cases where the HS solution complies with $u_{max} < 0.1$ N to allow an equal comparison. In this plot, it can be observed, with the excep-

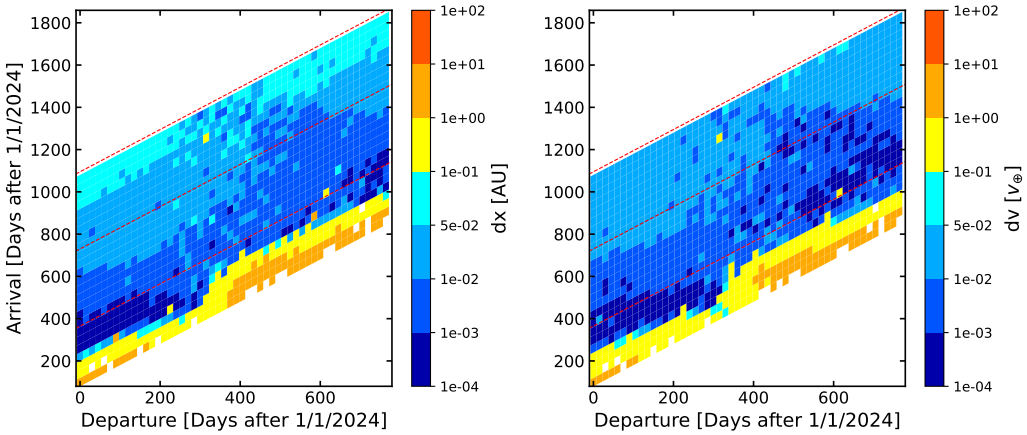

Figure 12. The verification metrics dx and dv of the PINN solution as a function of departure and arrival date for optimal-fuel Earth-Mars rendezvous transfers.

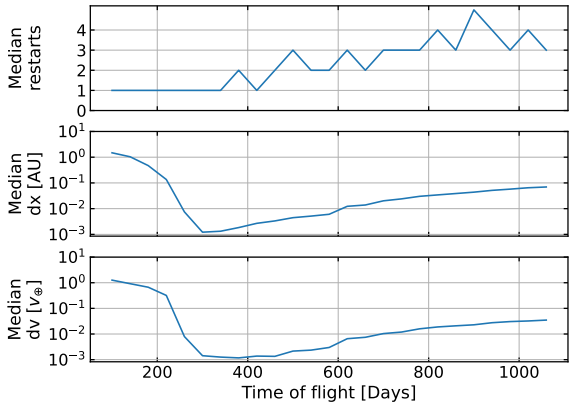

Figure 13. The influence of the time of flight on the accuracy of the solution and the required restarts for the cases in the Earth-Mars launch opportunity plot Figure 11.

tion of some outliers, that the HS method moderately outperforms the PINN 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 PINN 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 PINN in specifically those areas suggests that the method excels in searching solutions to nonlinear problems. All cases together in the right panel of Figure 14 have a median residual of -0.55 km s^{-1}, rendering the PINN method the preferred option for optimality. On the other hand, the average CPU training time per PINN 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 PINN 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 PINN based method, as indicated by the improved optimality of solutions generated by the PINN method for challenging boundary conditions.

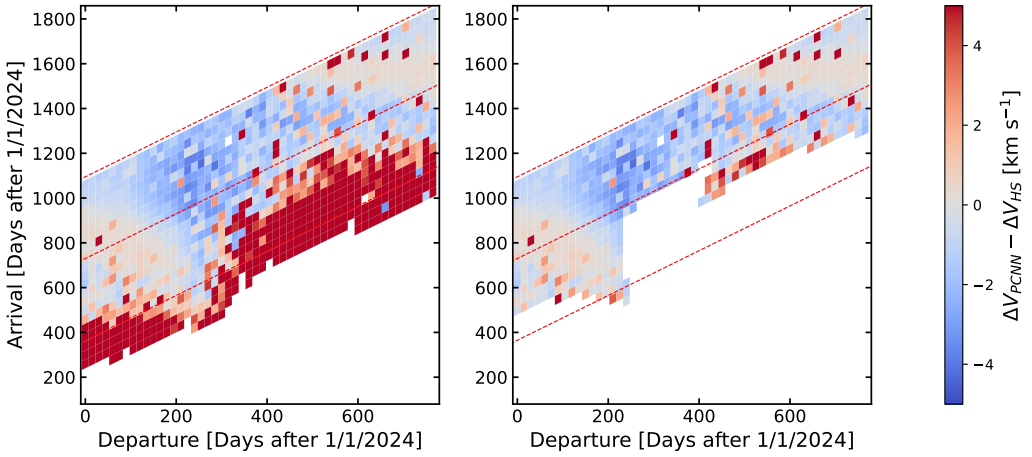

Figure 14. ΔV residuals between the PINN method and a HS method for optimal Earth-Mars transfer and rendezvous opportunities. Right plot excludes cases where the HS method provided solutions with $u_{max} > 0.1$ N, which was a constraint for the PINN method.

DISCUSSION

Physics-Informed 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 into complying 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 suitable in the context of more nonlinear dynamics, like the circular-restricted three-body problem, where the PINNs might be able to excel in providing preliminary concepts of mission designs. Furthermore, the method can naturally handle multi-objective optimization, although it would be challenging to 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.

An inconvenience of the PINN method in its current form, is that it does not inherently solve the OCP to within arbitrary accuracy of the physical model. By selection of the objective loss weight, a trade-off was made between satisfying the physical laws and optimality. Physical laws are not up to debate and can't be traded. In order to find the optimal solution that satisfied the physical model, the objective loss weight was selected in such a way that *after refinement*, the solution exhibited the minimal fitness while satisfying the dynamics to within 1 meter accuracy. This means that the refinement step, using a separate method, has to be performed, before the OCP is considered to be solved. Ideally, the neural network itself produces a solution that already satisfies the physical laws within a pre-defined accuracy. Additionally, 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. 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 robust.

CONCLUSION

An unsupervised Physics-Informed Neural network has been configured to solve optimal control problems with the direct method in order to design fuel-optimal low-thrust transfer trajectories. 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 PINN demonstrates proficiency in identifying near-optimal solutions across a diverse set of constraints. Furthermore, the PINN 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] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the astronautical Sciences*, Vol. 135, No. 1, 2009, pp. 497–528.

[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. 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.