



**POLITECNICO**  
**MILANO 1863**

**SCUOLA DI INGEGNERIA INDUSTRIALE  
E DELL'INFORMAZIONE**

EXECUTIVE SUMMARY OF THE THESIS

# Low-Thrust Spacecraft Transfers Using Physics-Informed Neural Networks

LAUREA MAGISTRALE IN SPACE ENGINEERING - INGEGNERIA SPAZIALE

**Author:** GIUSEPPE EDOARDO ADDARIO

**Advisor:** PROF. FRANCESCO TOPPUTO

**Co-advisors:** CHRISTIAN HOFMANN, ALESSANDRA MANNOCCHI, ANDREA CARLO MORELLI, MATTIA PUGLIATTI

**Academic year:** 2022-2023

## 1. Introduction

The objective of the research is to investigate recent developments in the field of deep learning to perform real-time guidance of low-thrust spacecraft. The methods based on deep learning meet the requirements for designing optimal trajectories on-board. In particular, the methods are suitable for the implementation on spacecraft computers given the limited computing power available. Thus, the ultimate goal of the research is to take a small step forward in the development of spacecraft capable of exploring the Solar System autonomously.

The study is based on physics-informed neural networks, a particular architecture developed to solve partial differential equations [7]. The purpose of physics-informed neural networks is to increase the reliability and to overcome the data dependence of standard networks by exploiting the physics behind the problem in the training process of the networks.

The work focuses on how the Hamilton-Jacobi-Bellman theory of optimal control can be exploited to build physics-informed neural networks. The end goal is to design a neurocontroller, i.e., a neural network-based controller,

capable to guide the spacecraft to its final destination with reliability while minimizing specific performance indices, such as the mass of propellant required.

## 2. Problem Statement

The problem considered is a low-thrust transfer between two planets in the Solar System. The orbits of the planets are assumed to be circular and coplanar. The spacecraft is modeled as a point mass and is subject to the gravitational attraction of the Sun and to the thrust force determined by its propulsion system.

The state of the spacecraft at a given time instant is denoted  $x(t) \in \mathbb{R}^5$  and can be expressed in polar coordinates as follows:

$$x(t) = [r(t), \phi(t), v_r(t), v_\phi(t), m(t)]^T \quad (1)$$

where  $r(t) \in \mathbb{R}_{>0}$  is the radial position,  $\phi(t) \in \mathbb{R}$  is the angular position,  $v_r(t), v_\phi(t) \in \mathbb{R}$  are respectively the radial and tangential components of the velocity and  $m(t) \in \mathbb{R}_{>0}$  is the mass. The motion of the spacecraft, in dimensionless form, is described by the following system of ordinary differential equations:

$$\begin{cases} \dot{r}(t) = v_r(t) \\ \dot{\phi}(t) = \frac{v_\phi(t)}{r(t)} \\ \dot{v}_r(t) = \frac{v_\phi^2(t)}{r(t)} - a_3 \frac{1}{r^2(t)} + a_1 \frac{u(t) \alpha_r(t)}{m(t)} \\ \dot{v}_\phi(t) = -\frac{v_r(t) v_\phi(t)}{r(t)} + a_1 \frac{u(t) \alpha_\phi(t)}{m(t)} \\ \dot{m}(t) = -a_2 u(t) \end{cases} \quad (2)$$

where  $u(t) \in [0, 1]$  is the thrust throttle factor and  $\alpha(t) \in A \subset \mathbb{R}^2$  is the thrust direction, with  $A := \{v \in \mathbb{R}^2 \mid \|v\|_2 = 1\}$ , and the notation  $\alpha(t) = [\alpha_r(t), \alpha_\phi(t)]^T$  is used. The dimensionless coefficients in the dynamics are defined as:

$$a_1 = \frac{T_{max} T_C^2}{m_i L_C}, \quad a_2 = \frac{T_{max} T_C}{I_{sp} g_0 m_i}, \quad a_3 = \mu \frac{T_C^2}{L_C^3} \quad (3)$$

where  $\mu$  is the Sun gravitational parameter,  $L_C$  and  $T_C$  are the characteristic length and time of the problem,  $T_{max}$ ,  $I_{sp}$  and  $m_i$  are the maximum thrust, the specific impulse and the initial mass of the spacecraft and  $g_0$  is the standard gravitational acceleration. The control applied to the spacecraft will be also referred as  $c(t) \in C \subset \mathbb{R}^3$ , where  $c(t) = [u(t), \alpha_r(t), \alpha_\phi(t)]^T$  and clearly  $C := \{(u, v, w) \in \mathbb{R}^3 \mid u \in [0, 1] \wedge (v, w) \in A\}$ . Therefore, the dynamics of the spacecraft can be expressed compactly as  $\dot{x}(t) = f(x(t), c(t))$ , where  $f := \mathbb{R}^5 \times C \rightarrow \mathbb{R}^5$ . The initial conditions of the spacecraft are:

$$x(t_i) = \left[ r_i, \phi_i, 0, \sqrt{\frac{a_3}{r_i}}, m_i \right]^T \quad (4)$$

where  $t_i \in \mathbb{R}$  is the initial time while  $r_i \in \mathbb{R}_{>0}$  and  $\phi_i \in \mathbb{R}$  are the orbital radius and the angular position of the departure planet, respectively. The final state of the spacecraft must respect the condition  $\psi(x(t_f)) = 0$ , with  $\psi := \mathbb{R}^5 \rightarrow \mathbb{R}^4$ :

$$\psi(x(t_f)) = \begin{bmatrix} r(t_f) - r_f \\ \phi(t_f) - \phi_f \\ v_r(t_f) \\ v_\phi(t_f) - \sqrt{\frac{a_3}{r_f}} \end{bmatrix} \quad (5)$$

where  $t_f \in \mathbb{R}$  is the final time and  $r_f \in \mathbb{R}_{>0}$  and  $\phi_f \in \mathbb{R}$  are the orbital radius and the angular position of the arrival planet, respectively. In summary, the final radial and angular positions

and final radial and tangential velocities of the spacecraft are fixed, while the final mass is free. Note that the final time is left free and must be obtained by solving the optimal control problem, so the problem can be classified as a free-time partially fixed endpoint control problem.

## 2.1. Optimal Control Problem

The optimal control problem addressed in the research can be formalized as follows:

$$\min_{c(\cdot) \in C} J(c(\cdot)) = a_2 \int_{t_i}^{t_f} L(c(t)) dt \quad (6a)$$

$$\text{subject to } \dot{x}(t) = f(x(t), c(t)), \quad (6b)$$

$$x(t_i) = x_i \quad (6c)$$

$$\psi(x(t_f)) = 0. \quad (6d)$$

with  $C := \{c := [t_i, t_f] \rightarrow C\}$ . In particular, the running cost,  $L := C \rightarrow \mathbb{R}$ , is defined as:

$$L(c(t)) = u(t) - \epsilon \log [u(t)(1 - u(t))] \quad (7)$$

where  $\epsilon \in \mathbb{R}_{\geq 0}$  is the homotopy coefficient introduced to smooth the problem. Note that if  $\epsilon = 0$ , the cost functional,  $J := C \rightarrow \mathbb{R}$ , then quantifies the propellant mass required for the transfer and the resulting problem is called fuel-optimal. The Hamiltonian of the problem,  $H := \mathbb{R}^5 \times C \times \mathbb{R}^5 \rightarrow \mathbb{R}$ , is defined as:

$$H(p, q, r) = \lambda_0 a_2 L(q) + \langle r, f(p, q) \rangle \quad (8)$$

where  $\lambda_0 \in \mathbb{R}_{>0}$  is the newly introduced costate normalization factor [5]. The necessary conditions to minimize the cost functional, called the Euler-Lagrange equations, take the form:

$$\begin{cases} \dot{x}(t) = \partial_r H(x(t), c(t), \lambda(t)) & (9a) \\ \dot{\lambda}(t) = -\partial_p H(x(t), c(t), \lambda(t)) & (9b) \end{cases}$$

where  $\lambda(t) \in \mathbb{R}^5$  is the costate of the spacecraft:

$$\lambda(t) = [\lambda_r(t), \lambda_\phi(t), \lambda_{v_r}(t), \lambda_{v_\phi}(t), \lambda_m(t)]^T \quad (10)$$

The optimal control is evaluated by applying Pontryagin's Maximum Principle. To this aim, the auxiliary function  $Q := \mathbb{R}^5 \times \mathbb{R}^5 \rightarrow \mathbb{R}^3$  is introduced:

$$Q(p, r) = \arg \min_{\bar{c} \in C} H(p, \bar{c}, r) \quad (11)$$

The optimal control is then obtained by evaluating the previous expression on the state and costate of the spacecraft:

$$c_*(t) = Q(x(t), \lambda(t)) \quad (12)$$

In detail, for the problem under consideration, the following holds [1]:

$$Q(x(t), \lambda(t)) = \begin{bmatrix} 2\epsilon \\ \frac{2\epsilon + S(x(t), \lambda(t)) + \sqrt{4\epsilon^2 + S(x(t), \lambda(t))^2}}{\lambda_{v_r}(t)} \\ -\frac{\|\lambda_v(t)\|}{\lambda_{v_\phi}(t)} \\ -\frac{\lambda_{v_\phi}(t)}{\|\lambda_v(t)\|} \end{bmatrix} \quad (13)$$

where the notation  $\lambda_v(t) = [\lambda_{v_r}(t), \lambda_{v_\phi}(t)]^T$  is used. Note that the function introduced in the previous expression,  $S := \mathbb{R}^5 \times \mathbb{R}^5 \rightarrow \mathbb{R}$ , is called switching function and takes the form:

$$S(x(t), \lambda(t)) = 1 - \frac{\lambda_m(t)}{\lambda_0} - \frac{a_1 \|\lambda_v(t)\|}{a_2 \lambda_0 m(t)} \quad (14)$$

To conclude the formulation of the optimal control problem, the transversality condition, which can be derived from the application of the Euler-Lagrange theorem, dictates that the Hamiltonian and the mass costate evaluated at the final time must be zero, i.e.  $H(x(t_f), c_*(t_f), \lambda(t_f)) = 0$  and  $\lambda_m(t_f) = 0$ .

The problem presented can be reformulated as a two-point boundary value problem and solved with shooting methodologies. In this regard, note that the only unknowns of the problem are the initial costate,  $\lambda_i \in \mathbb{R}^n$ , such that  $\lambda(t_i) = \lambda_i$ , the costate normalization factor and the final time. If these variables are known the dynamical evolution of the state of the spacecraft can be obtained by integrating Equations (9) with the control evaluated with Equation (12). To determine the unknown variables, the shooting function,  $\gamma := \mathbb{R}^5 \times \mathbb{R} \times \mathbb{R}_{>0} \rightarrow \mathbb{R}$ , is introduced:

$$\gamma(\lambda_i, t_f, \lambda_0) = \begin{bmatrix} \psi(x(t_f)) \\ H(x(t_f), c(t_f), \lambda(t_f)) \\ \lambda_m(t_f) \\ \sqrt{\langle \lambda_i, \lambda_i \rangle + \lambda_0^2} - 1 \end{bmatrix} \quad (15)$$

Note that the last condition is introduced to determine the costate normalization factor. To

solve the optimal control problem, it is necessary to find the zeros of the shooting function and in particular, this task can be accomplished with root finding algorithms. In detail, in this research, a modification of the Powell hybrid method, as implemented in the SciPy library of Python, is exploited with a tolerance set to  $xtol = 10^{-12}$ .

## 2.2. Hamilton-Jacobi-Bellman Theory

The cost functional of the problem can be interpreted in a more general sense as a function of the the initial state as well, i.e.  $J : \mathbb{R}^5 \times \mathcal{C} \rightarrow \mathbb{R}$ . In this regard, it is possible to introduce the value function,  $\nu : \mathbb{R}^5 \rightarrow \mathbb{R}$ , which represents the optimal cost of the transfer if the spacecraft starts from a given initial state:

$$\nu(X) = \min_{c(\cdot) \in \mathcal{C}} J(X, c(\cdot)) \quad (16)$$

Note that the value function is time independent due to the properties of the problem. The value function must satisfy the Hamilton-Jacobi-Bellman partial differential equation [2]:

$$\begin{cases} \min_{\bar{c} \in \mathcal{C}} H(X, \bar{c}, \partial_X \nu(X)) = 0 & X \in \mathbb{R}^5 & (17a) \\ \nu(X) = 0 & \forall X \in \mathbb{R}^5 : \psi(X) = 0 & (17b) \end{cases}$$

In particular, from the previous discussion, the minimizer can be evaluated as:

$$\bar{c} = Q(X, \partial_X \nu(X)) \quad (17c)$$

It must be emphasized that the knowledge of the value function is sufficient to evaluate the optimal control given the state of the spacecraft:

$$c_*(t) = Q(x(t), \partial_X \nu(x(t))) \quad (18)$$

Therefore, by exploiting this relation, the spacecraft can be controlled with a feedback control law. In addition, considering Equation (12), it is evident that the following must hold along an optimal trajectory:

$$\lambda(t) = \partial_X \nu(x(t)) \quad (19)$$

Thus, a link is established between the costate introduced in the Euler-Lagrange equations and the value function defined in the Hamilton-Jacobi-Bellman theory.

### 3. Backward Generation Method

The first step to design a neurocontroller consists in constructing a database of optimal trajectories which can be exploited to train the neural networks. In this regard, the straightforward approach would be to perturb the initial state of the nominal trajectory and to solve the corresponding optimal control problem characterized by the new set of initial conditions, thus generating a new optimal trajectory. This approach scales very poorly because of the computational difficulties associated with solving two-points boundary value problems. Therefore, in order to generate such a database, the *backward generation method*, introduced by Izzo et Öztürk [4], is exploited. It should be noted that the method was originally presented for a transfer between two orbits and in this work it is extended to the rendezvous problem discussed in the previous sections.

The first step of the method consists in perturbing the components of the final state and costate of the spacecraft of the nominal trajectory. The perturbations cannot be arbitrary because the following conditions must be respected:

- The final radial and angular positions and the final radial and tangential velocities of the spacecraft must coincide with the arrival planet ones.
- The mass costate must be zero due to the transversality condition.

Therefore, these components cannot be perturbed and are kept constant while the remaining free components are perturbed as follows:

$$\beta_i^{new}(t_f) = \beta_i^{old}(t_f) + \delta_i \quad i = 1, \dots, n_{free} \quad (21)$$

where  $\beta_i(t_f)$  represents a generic free component and  $\delta_i \in \mathbb{R}$  is the associated perturbation:

$$\delta_i = \rho v_i \quad (22)$$

where  $\rho \in \mathbb{R}_{>0}$  is the *perturbation size* and the coefficient  $v_i$  is sampled from a uniform distribution  $\mathcal{U}(-1, 1)$ . There is still a constraint which must be considered: the Hamiltonian, evaluated at the final time, must be zero. In order to meet this constraint all the free components are perturbed, except one denoted  $\beta_j^{old}(t_f)$  and referred as *unknown variable*, so  $\beta_j^{new}(t_f) = \beta_j^{old}(t_f) + \delta_j$  is still to be determined. Let us define the auxiliary function,  $g : \mathbb{R} \rightarrow \mathbb{R}$ , as follows:

$$g(\delta_j) = H(x^{new}(t_f), c_*(t_f), \lambda^{new}(t_f)) \quad (23)$$

The perturbation of the unknown variable is then evaluated by solving the root-finding problem:  $g(\delta_j) = 0$ . The perturbed final state and costate can then be numerically propagated backward in time to obtain an optimal trajectory. Note that in this procedure the costate normalization factor is kept constant, so it is common to all the back-propagated trajectories.

#### 3.1. Trust Band

The optimal trajectories generated by exploiting the backward generation method could be very different in nature with respect to the nominal trajectory. Nevertheless, it is desirable to generate trajectories close to the nominal one so that the neural networks can focus on learning patterns over a limited region of space, and thus smaller databases and simpler networks can be adopted in the training process. To this aim, the perturbation size in Equation (22) is finely tuned so that the initial positions of the back-propagated trajectories lie within a circle centered on the departure planet, the radius of the circle being referred as *maximum distance*. The result is a beam of trajectories which presents a thickness constantly decreasing going from the departure planet to the arrival planet, indeed it is worth remembering that all the trajectories converge to the same final position. The region defined by the beam of trajectories, from which the database samples are obtained, will be referred as *trust band*. To clarify the concept, consider as example the transfer between the fictitious planets *Helicon* and *Terminus*<sup>1</sup>. An example of the effect of the perturbation size is reported in Figure 1 where high, medium and low perturbation sizes are used to generate the optimal trajectories. Note that the concept of trust band can be clearly appreciated in the last figure. It must be noticed that the neural networks are trained on samples belonging to the trust band, so the neurocontroller is expected to be effective in driving the spacecraft from any position (and overall meaningful spacecraft state) within the trust band to the target planet. Therefore, the goal is to keep the spacecraft within the trust band as much as possible to increase the probability of reaching the target

<sup>1</sup>Isaac Asimov, *Foundation Series*.

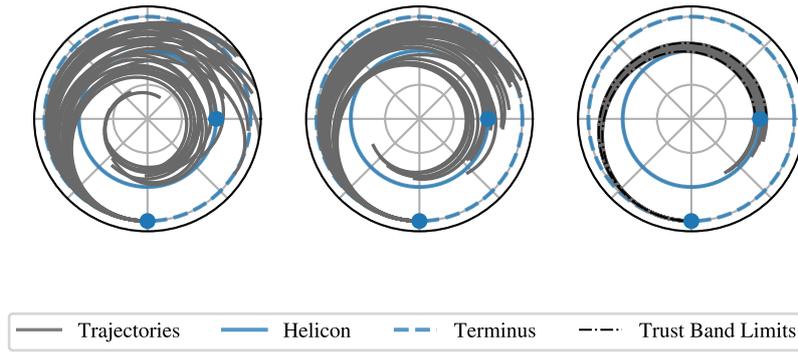


Figure 1: Effects of the perturbation size.

planet. It should be noted, however, that as soon as the spacecraft approaches the arrival planet, the trust band narrows, and thus, if the guidance is not perfect, there is a high probability of leaving this region and thus the neurocontroller may experience a drop in performance because over generalization capabilities are requested to the neural networks.

#### 4. Neural Networks

The second step in designing a neurocontroller encapsulates every aspect related to the creation and the training process of neural networks. Therefore, in this section, the neural networks are introduced and the models exploited in the research are presented. Note that the numerical implementation of neural networks is based on the TensorFlow 2 library of Python.

The neural networks generally consist of an input layer, a number of hidden layers and an output layer. The mathematical function representing a neural network,  $\mathcal{N}(\cdot | \theta) := \mathbb{R}^m \rightarrow \mathbb{R}^n$ , can be defined from the maps of the layers as:

$$\mathcal{N}(x | \theta) = (g^{[l-1]} \circ \dots \circ g^{[1]})(x) \quad (24)$$

where  $\theta \in \mathbb{R}^p$  are the trainable parameters and  $l$  denotes the total number of layers. The map of a generic layer,  $g^{[i]}(\cdot | \theta^{[i]}) := \mathbb{R}^{m_i} \rightarrow \mathbb{R}^{n_i}$ , can be defined as:

$$g^{[i]}(x | \theta^{[i]}) = \sigma^{[i]}(W^{[i]}x + b^{[i]}) \quad (25)$$

where the vector of the parameters of the layer,  $\theta^{[i]} \in \mathbb{R}^{(n_i m_i + n_i)}$ , is composed of the entries of the weights matrix,  $W^{[i]} \in \mathbb{R}^{n_i \times m_i}$ , rearranged in vector form, and the components of the biases vector,  $b^{[i]} \in \mathbb{R}^{n_i}$ . Note that  $n_i$  denotes the number of computational units, called neurons,

presents in the layer. The layer activation function,  $\sigma^{[i]} := \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_i}$ , is simply:

$$\sigma^{[i]}(x) = [\sigma_1^{[i]}(x), \dots, \sigma_{n_i}^{[i]}(x)]^T \quad (26)$$

where the generic  $\sigma_j^{[i]} := \mathbb{R} \rightarrow \mathbb{R}$  is a nonlinear function called the neuron activation function. With regards to the training process, in this study, neural networks are trained with supervised learning. In supervised learning, a labelled database, i.e a collection of inputs and the corresponding correct outputs, is exploited. The training can be then decomposed in a sequence of phases. In the first phase, the inputs contained in the database are fed to the network which in turn predicts the corresponding outputs. In the second phase, the predicted outputs are compared with the correct outputs through an evaluation metric called the loss function. Finally, the network adjusts the parameters to minimize the loss function, i.e to find the best fit of the data included in the database.

##### 4.1. Models

In order to address the spacecraft guidance problem, standard neural networks and physics-informed neural networks are considered. The main difference between them is the way the costate is predicted from the information about the spacecraft state. In particular, standard neural networks directly predict the costate, thus, the neural network function is defined as:  $\mathcal{N}(\cdot | \theta) : \mathbb{R}^5 \rightarrow \mathbb{R}^5$ . In contrast, physics-informed neural networks try to find the best approximation of the value function, namely  $\nu(\cdot) \approx \mathcal{N}(\cdot, \theta)$ , where  $\mathcal{N}(\cdot | \theta) : \mathbb{R}^5 \rightarrow \mathbb{R}$  and the costate is obtained as the gradient of the network function. In the study, four different model

of neural networks are investigated. The first model is a standard neural network denoted  $\mathbb{S}$  while the remaining models are physics-informed networks denoted  $\mathbb{P}_1$ ,  $\mathbb{P}_2$  and  $\mathbb{P}_3$ . Note that the database is common to all networks and it is defined as  $\mathcal{D} = \{(x_*^{(i)}, \lambda_*^{(i)}, \nu_*^{(i)}, u_*^{(i)}, \alpha_*^{(i)})\}_{i=1}^m$ , and collects the states, costates, transfers costs and controls sampled from the optimal trajectories. The loss functions used to trained the standard network is defined as:

$$l_{\mathbb{S}}(\theta | \mathcal{D}) = \frac{1}{5m} \sum_{i=1}^m \|\lambda_*^{(i)} - \mathcal{N}(x_*^{(i)} | \theta)\|_2^2 \quad (27)$$

In contrast, the composite loss functions used to train the physics-informed networks are:

$$l_{\mathbb{P}_1}(\theta | \mathcal{D}) = l_{\nu}(\theta | \mathcal{D}) + \mu_{\lambda} l_{\lambda}(\theta | \mathcal{D}) \quad (28)$$

$$l_{\mathbb{P}_2}(\theta | \mathcal{D}) = l_{\nu}(\theta | \mathcal{D}) + \mu_H l_H(\theta | \mathcal{D}) \quad (29)$$

$$l_{\mathbb{P}_3}(\theta | \mathcal{D}) = l_{\mathbb{P}_1}(\theta | \mathcal{D}) + \mu_H l_H(\theta | \mathcal{D}) \quad (30)$$

where  $\mu_{\lambda}, \mu_H \in \mathbb{R}$  are the weights introduced to balance the different terms, in particular these parameters are set to  $\mu_{\lambda} = 10$  and  $\mu_H = 1$ . The individual terms in the previous relations are defined as follows:

$$l_{\nu}(\theta | \mathcal{D}) = \frac{1}{m} \sum_{i=1}^m [\nu_*^{(i)} - \mathcal{N}(x_*^{(i)} | \theta)]^2 \quad (31)$$

$$l_H(\theta | \mathcal{D}) = \frac{1}{m} \sum_{i=1}^m [H(x_*^{(i)}, Q(x_*^{(i)}, \lambda_{\mathcal{N}}^{(i)}), \lambda_{\mathcal{N}}^{(i)})]^2 \quad (32)$$

$$l_{\lambda}(\theta | \mathcal{D}) = \frac{1}{5m} \sum_{i=1}^m \|\lambda_*^{(i)} - \lambda_{\mathcal{N}}^{(i)}\|_2^2 \quad (33)$$

where  $\lambda_{\mathcal{N}}^{(i)} = \partial_x \mathcal{N}(x_*^{(i)} | \theta)$ , i.e the costate is computed as the gradient of the neural network function evaluated at the given state of the database. Note that the term in Equation (32) imposes the structure of the Hamilton-Jacobi-Bellman equation on a finite number of collocation points while the term in Equation (33) leverages the fact that the gradient of the value function must correspond to the costate introduced in the Euler-Lagrange equations along an optimal trajectory.

## 4.2. Architectures

The hyperbolic tangent function is selected as neuron activation function for all neural networks models. The reason is that, at the preliminary stage, this function proved to be superior

to other activation functions such as the sigmoid and the softplus. The hyperbolic tangent function,  $\tanh := \mathbb{R} \rightarrow [-1, 1]$  is defined as:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (34)$$

In particular, in the neural networks the adaptive version of the function is implemented:

$$\tanh_{adaptive}(x) = \tanh(\rho \kappa x) \quad (35)$$

where the parameter  $\kappa \in \mathbb{R}$  is referred as slope and is adjusted during training, exactly like the other trainable parameters, while  $\rho \in [1, +\infty)$  is a fixed scale factor which is used to accelerates the convergence towards the global minima. In detail, the scale factor is set to  $\rho = 10$  while the initial value of the slope is selected as  $\kappa = 1/10$  such that  $\kappa\rho = 1$ , as suggested by the authors of the adaptive functions. To conclude, the possible architectures for all neural networks models are selected by considering between 4 and 8 hidden layers and 32 and 64 neurons per hidden layer.

## 4.3. Training

In the analysis, the database is divided into training, validation and test datasets with the proportions 60% – 20% – 20%. The standard neural network is trained with ADAM with a learning rate of  $1 \times 10^{-4}$  for a maximum number of iterations set to 20000 while the physics-informed networks are trained for 1000 iterations with ADAM with a learning rate of  $5 \times 10^{-4}$ , and then with L-BFGS for a maximum number of iterations of 2000. The remaining settings of ADAM and L-BFGS are kept at their default values, except for the number of correction pairs to approximate the Hessian matrix, which is set to 50. In training physics-informed networks, ADAM is used to properly initialize the search for minima because L-BFGS tends to stop in the early phase on local minima. In order to avoid the overfitting of the training dataset the training is stopped after the loss function has not decreased by at least  $1 \times 10^{-7}$  in the last 100 iterations on the validation dataset. The adoption of this technique is particularly useful for comparing different neural network models. In fact, the goal is to obtain the best performance for each network, and depending on the model, this may require more or fewer iterations. Therefore,

the methodology is exploited to avoid setting a predetermined number of iterations while still getting the most out of the training process.

#### 4.4. Metrics

The best architecture for each model is selected based on the following metric evaluated on the validation dataset:

$$\Delta_{u-\alpha} = \sqrt{\Delta_u \Delta_\alpha} \quad (36)$$

where the individual components of this metric are defined as follows:

$$\Delta_u = \frac{1}{m} \sum_{i=1}^m |u_*^{(i)} - u_{\mathcal{N}}^{(i)}| \quad (37a)$$

$$\Delta_\alpha = \frac{1}{m} \sum_{i=1}^m \arccos(\langle \alpha_*^{(i)}, \alpha_{\mathcal{N}}^{(i)} \rangle) \quad (37b)$$

and  $u_{\mathcal{N}}^{(i)}$  and  $\alpha_{\mathcal{N}}^{(i)}$  denote the controls evaluated with the predictions of the costate by the networks, i.e.  $c_{\mathcal{N}}^{(i)} = Q(x_*^{(i)}, \lambda_{\mathcal{N}}^{(i)})$ . The physics-informed neural networks can be tested also on the ability to predict the transfer cost from a given state of the spacecraft, and so the following metric is introduced for this purpose:

$$\Delta_\nu = \frac{1}{m} \sum_{i=1}^m |\nu_*^{(i)} - N(x_*^{(i)} | \theta)| \quad (38)$$

With regard to the neurocontroller, it is evaluated on the trajectories included in the test dataset, in particular, it is asked to propagate the initial states of these trajectories. The resulting trajectory is referred as neural trajectory for simplicity. Note that the final integration time of the neural trajectory is selected to coincide with the time of flight of the corresponding test trajectory. To evaluate the performance of the neurocontroller three metrics are introduced. The first metric,  $\Delta_r$ , is the average final distance to the target planet (considering all neural trajectories) and similarly, the second metric,  $\Delta_v$ , is the average euclidean norm of the difference between the spacecraft final velocity and the target planet velocity. The last metric,  $\Delta_J$ , instead is the mean absolute error between the optimal cost of the transfer and the one obtained considering the neurocontroller.

## 5. Case Studies

The physical constants adopted for the case studies are given in Table 1. To formulate

Table 1: Physical constants.

Quantity	Value
Astronomical Unit (AU)	$1.496 \times 10^8$ km
Sun Gravitational Parameter	$1.327 \times 10^{11}$ km <sup>3</sup> /s <sup>2</sup>
Standard Gravitational Acceleration	$9.807 \times 10^{-3}$ km/s <sup>2</sup>
Sun-Earth Distance	1.000 AU
Sun-Venus Distance	0.723 AU
Sun-Mars Distance	1.524 AU

the equations of motion in dimensionless form, the astronomical unit is chosen as characteristic length and the characteristic time is selected so that the Sun's dimensionless gravitational parameter, i.e. the coefficient  $a_3$  in Equations (3), is unitary. The spacecraft considered in the cases studies is characterized by  $T_{max} = 0.30$  N,  $I_{sp} = 3000$  s and  $m_i = 1500$  kg.

## 6. Earth-Venus

The first case study is an optimal transfer from Earth to Venus, considering the homotopy coefficient set to  $\epsilon = 1$  and an angular separation between the planets of 180 deg. The nominal trajectory is characterized by a total time of flight of  $t_f = 1.1755$  yr and a propellant mass consumption of 256.78 kg. To generate the database of optimal trajectories the maximum distance to the departure planet is set to  $5 \times 10^6$  km, the perturbation size is selected as  $10^{-3}$  and the unknown variable is selected to be  $m(t_f)$ . The database is composed by 100 optimal trajectories each sampled at 100 random time instants. The small number of trajectories is selected to speed up the training but the limited amount of data can be theoretically overcome with physics. In fact, the physical constraints imposed during the training virtually enrich the database as neural networks exploit the datapoints to understand the physics of the problem. The positions where the datapoints are sampled along the optimal trajectories are displayed in Figure 2, note that only a fraction of the positions is reported. In particular, the positions are highlighted according to the thrust throttle factor value of the corresponding datapoint. It can be noticed that the thrust bands presents a decreasing thickness in the transition from Earth to Venus. The best architecture for each model is selected based on the metric defined in Equation (36) and the models are then evaluated on the ability to predict the optimal controls and

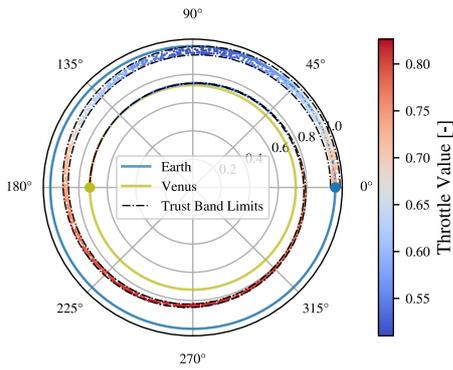


Figure 2: Earth-Venus: datapoints.

the transfers costs from the states included the test dataset. The values of the metrics defined in Equation (37) and in Equation (38) are reported in Table 2. It can be noticed that the models are accurate in the predictions with the only exception of  $\mathbb{P}_2$ .

Table 2: Earth-Venus: models predictions.

	$\Delta_u$ [-]	$\Delta_\alpha$ [deg]	$\Delta_\nu$ [kg]
$\mathbb{S}$ [8/64]*	$7.7 \times 10^{-4}$	0.230	-
$\mathbb{P}_1$ [4/64]	$5.9 \times 10^{-4}$	0.140	0.08
$\mathbb{P}_2$ [4/32]	$8.6 \times 10^{-2}$	16.050	0.61
$\mathbb{P}_3$ [4/64]	$5.8 \times 10^{-4}$	0.136	0.04

\* indicates the best model architecture

The models are implemented in the neurocontroller to guide the spacecraft in closed loop and the position, velocity and optimality errors as defined in the previous sections are reported in Table 3.

Table 3: Earth-Venus: neurocontrollers errors.

	$\Delta_r$ [km]	$\Delta_v$ [km s <sup>-1</sup> ]	$\Delta_J$ [kg]
$\mathbb{S}$ [8/64]*	$1.62 \times 10^5$	$1.69 \times 10^{-2}$	0.05
$\mathbb{P}_1$ [4/64]	$1.46 \times 10^5$	$1.59 \times 10^{-2}$	0.04
$\mathbb{P}_2$ [4/32]	$1.11 \times 10^7$	2.64	13.31
$\mathbb{P}_3$ [4/64]	$1.57 \times 10^5$	$1.60 \times 10^{-2}$	0.04

\* indicates the best model architecture

It can be seen that the final errors are not negligible and a corrective maneuver would be needed to close the gap to rendezvous with the target planet. It is evident that the accumulation of small inaccuracies in the predicted controls leads to significant final position and velocity errors.

In this regard, it must be emphasized that the optimality error does not give a complete picture because the cost of the additional maneuver should be considered. In particular, it can be noticed that physics-informed networks perform slightly better than the standard network. The neurocontroller based on  $\mathbb{P}_2$ [4/32] presents the worst performance: it seems that the network is unable to replicate the optimal controls. The problem is not necessarily due to the way the loss function is conceptually conceived, but it could be the way the function is mathematically formulated. In other words, the issue may be the minimization of this type of loss function. To try to shed light on this, the loss landscape around the point of convergence for  $\mathbb{P}_2$ [4/32] is compared with that of  $\mathbb{P}_3$ [4/64]. It should be emphasized that the loss landscape is a local visualization of the loss function. The loss function is indeed a high-dimensional function, but by applying dimensionality reduction techniques it is possible to obtain a representation that provides useful information about the outcome of the neural network training process [6]. To cross-check the consistency of the analysis, three different landscapes are evaluated for each model and the results are reported in Figure 3. It is evident that the loss landscape of  $\mathbb{P}_2$ [4/32] is jagged and presents more than one depression, in contrast to the loss landscape of  $\mathbb{P}_3$ [4/64] which appears regular and shows a single minimum. The conformation of the landscape of  $\mathbb{P}_2$ [4/32] can be the main obstacle to the minimization of the associated loss function: it is reasonable to assume that the optimizer may get stuck on local minima. To overcome this difficulty, the first possibility might be to consider a more sophisticated type of neural network to capture the complexity of the Hamilton-Jacobi-Bellman equation. The second alternative could be to change the way the partial differential equation is exploited to formulate the loss function. It can be pointed out in this regard how the term based on the relationship between the two optimality principles integrates well in the construction of the loss function while at the same time increasing the reliability of the network.

## 7. Earth-Mars

The second case study is an optimal transfer from Earth to Mars, considering  $\epsilon = 10^{-5}$  and

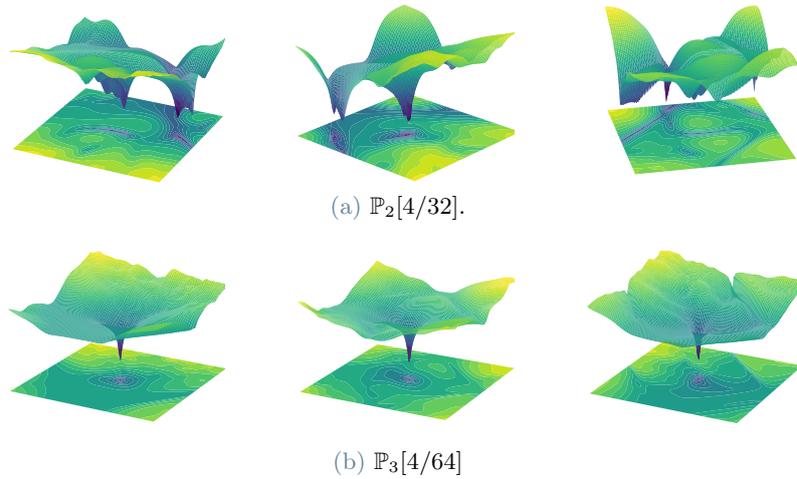


Figure 3: Earth-Venus: loss landscapes.

an angular separation between the planets of 90 deg. Note that the homotopy coefficient is sufficiently small so that the problem can be considered a fuel-optimal transfer. The nominal trajectory is characterized by a time of flight of  $t_f = 1.5662$  yr with a propellant mass consumption of 260.92 kg. To generate the database of optimal trajectories the maximum distance to the initial position is set to  $5 \times 10^6$  km, the perturbation size is selected as  $10^{-4}$  and the unknown variable is selected to be  $\lambda_\phi(t_f)$ . Also in this case study, the database consists of 100 trajectories each sampled at 100 random time instants. The database can be visualized in Figure 4. Note in particular the evolution of the thickness of the trust band, which shows a bulge in the central portion of the transfer. In fuel-

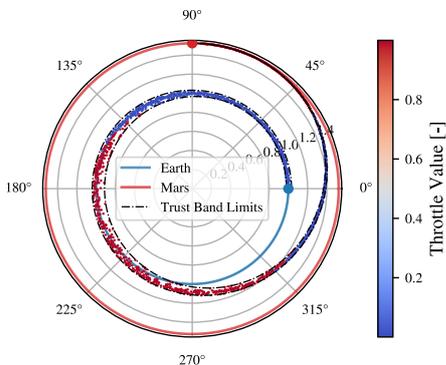


Figure 4: Earth-Mars: datapoints.

optimal problems, it turned out to be challenging to control the spread of trajectories, as even small perturbation sizes lead to trajectories very different from the nominal one. In this regard,

the main cause could be the structure of the equations in the formulation of the fuel-optimal problem. The best architectures for each model are evaluated on the ability to predict the controls on the test dataset and in addition, the physics-informed networks are asked to predict the propellant mass consumption. The results of this analysis are reported in Table 4.

Table 4: Earth-Mars: models predictions.

	$\Delta_u$ [-]	$\Delta_\alpha$ [deg]	$\Delta_\nu$ [kg]
$\mathbb{S}$ [8/64]*	0.169	0.053	-
$\mathbb{P}_1$ [4/64]	0.038	0.044	0.12
$\mathbb{P}_2$ [8/64]	0.311	2.974	0.09
$\mathbb{P}_3$ [8/64]	0.024	0.045	0.06

\* indicates the best model architecture

It can be seen that the networks present good prediction capabilities and in particular physics-informed networks can estimate the transfer costs with high accuracy. Therefore, this type of networks may also be useful in the preliminary design of missions to obtain a rough estimate of the required mass associated to different transfer options. With regard to the model  $\mathbb{P}_2[8/64]$ , the same considerations made for the previous case study apply. The performance of the neurocontrollers based on the various models is reported in Table Table 5. It can be seen that the final errors are even more pronounced than in the previous case study. Nevertheless, it is interesting to note that the neurocontrollers based on the physics-informed network  $\mathbb{P}_1[4/64]$  and  $\mathbb{P}_3[4/64]$

perform much better than the one based on the standard neural network.

Table 5: Earth-Mars: neurocontrollers errors.

	$\Delta_r$ [km]	$\Delta_v$ [km s <sup>-1</sup> ]	$\Delta_J$ [kg]
$\mathbb{S}$ [8/64]*	$6.96 \times 10^7$	5.30	78.66
$\mathbb{P}_1$ [4/64]	$5.36 \times 10^6$	$1.98 \times 10^{-1}$	3.16
$\mathbb{P}_2$ [8/64]	$2.16 \times 10^8$	5.88	85.76
$\mathbb{P}_3$ [4/64]	$4.59 \times 10^6$	$1.74 \times 10^{-1}$	6.91

\* indicates the best model architecture

In addition, focusing on the neurocontrollers based on  $\mathbb{P}_1$ [4/64] and  $\mathbb{P}_3$ [4/64], it is evident that the Hamiltonian-related term considered in the loss function of  $\mathbb{P}_3$ [4/64] turn out to be beneficial. The trajectories propagated by the neurocontrollers are shown in Figure 5. Note that grey and orange colors indicate whether the trajectory is inside or outside the trust band, respectively.

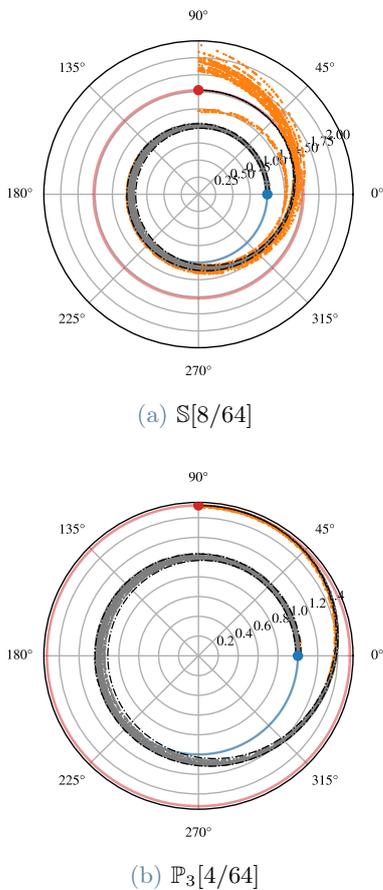


Figure 5: Earth-Mars: neural trajectories.

It can be seen that the neurocontroller based on  $\mathbb{S}$ [8/64] takes the spacecraft out of the trust band even before the end of the first burning arc. The main consequence is an increase in the final errors because, it should be remembered, the networks are not trained to predict the controls outside the band. In this case, the problem can be clearly appreciated in Figure 6: the neurocontrollers based on  $\mathbb{S}$ [8/64] is unable to correctly replicate the second ignition of the engine. On the other hand, it can be seen that  $\mathbb{P}_3$ [4/64] is able to capture the optimal time evolution of the thrust throttle factor. Note that the comparison is based on the best neural trajectories in term of the final position error and the thrust angle,  $\beta(t) \in (-\pi/2, \pi/2)$ , is defined as:

$$\beta(t) = \arctan \left( \frac{\alpha_r(t)}{\alpha_\phi(t)} \right) \quad (39)$$

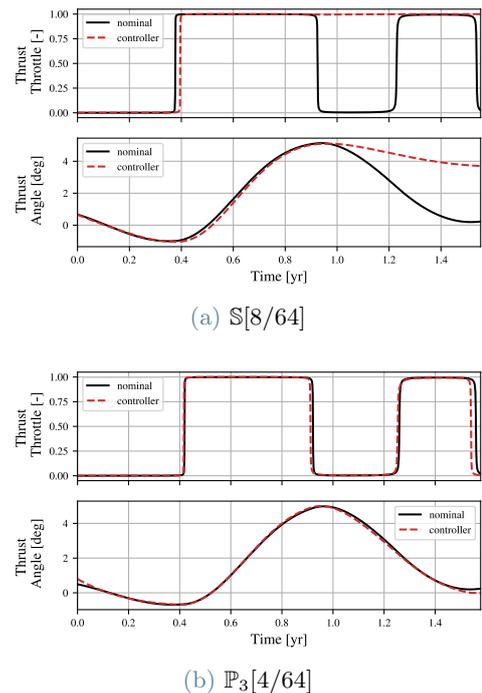


Figure 6: Earth-Mars: controls comparison.

## 8. Conclusions

In this research, the reliability issue of standard neural networks is addressed by following two complementary approaches. First, the trust band concept is introduced to specify the range of validity of using neural networks in low-thrust transfers. Second, the mathematical structure of

the optimality principles is imposed as soft constraint in the training process of the networks. The introduction of physics turned out to be extremely advantageous: the networks locally learn the structure of the Hamilton-Jacobi-Bellman equation. Therefore, physics-informed neural networks prove to be more effective in guiding spacecraft. In addition, these networks can predict with accuracy the transfers costs and this is a valuable additional feature with respect to standard networks.

To conclude, in all the case studies the neural networks-based controllers are unable to guide the spacecraft to the final destination with the accuracy required for operational purposes. Nevertheless, these results are consistent with those found in literature, and therefore the accuracy of terminal guidance should be addressed in future research.

## References

- [1] R. Bertrand and R. Epenoy. New smoothing techniques for solving bang-bang optimal control problems—numerical results and statistical interpretation. *Optimal Control Applications and Methods*, 23(4):171–197, 2002.
- [2] A.E. Bryson and Y.C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. A Halsted Press book. Hemisphere Publishing Corporation, 1975.
- [3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [4] Dario Izzo and Ekin Öztürk. Real-time guidance for low-thrust transfers using deep neural networks. *Journal of Guidance, Control, and Dynamics*, 44(2):315–327, 2021.
- [5] Fanghua Jiang, Hexi Baoyin, and Junfeng Li. Practical techniques for low-thrust trajectory optimization with homotopic approach. *Journal of Guidance, Control, and Dynamics*, 35(1):245–258, 2012.
- [6] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [7] 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*, 378:686–707, 2019.