

NEURAL APPROXIMATORS FOR LOW-THRUST TRAJECTORY TRANSFER COST AND REACHABILITY

Zhong Zhang*, Yilin Zou[†], and Francesco Toppeto[‡]

In trajectory design, fuel consumption and trajectory reachability are two key performance indicators for low-thrust missions. This paper proposes general-purpose pretrained neural networks to predict these metrics. The contributions of this paper are as follows: Firstly, based on the confirmation of the Scaling Law applicable to low-thrust trajectory approximation, the largest dataset is constructed using the proposed homotopy ray method, which aligns with mission-design-oriented data requirements. Secondly, the data are transformed into a self-similar space, enabling the neural network to adapt to arbitrary semi-major axes, inclinations, and central bodies. This extends the applicability beyond existing studies and can generalize across diverse mission scenarios without retraining. Thirdly, to the best of our knowledge, this work presents the current most general and accurate low-thrust trajectory approximator, with implementations available in C++, Python, and MATLAB. The resulting neural network achieves a relative error of 0.78% in predicting velocity increments and 0.63% in minimum transfer time estimation. The models have also been validated on a third-party dataset, multi-flyby mission design problem, and mission analysis scenario, demonstrating their generalization capability, predictive accuracy, and computational efficiency.

INTRODUCTION

Low-thrust electric propulsion has gained significant attention due to its higher specific impulse and improved efficiency over traditional high-thrust chemical propulsion. As a result, it has been widely used in real-world missions such as Deep Space 1,¹ Hayabusa,² BepiColombo,³ and Psyche.⁴ Despite these advantages, low-thrust propulsion poses significant challenges for mission designers. Unlike high-thrust systems, optimizing low-thrust trajectories involves solving more complex optimal control problems. During preliminary mission design, it is often necessary to evaluate a large number of trajectory candidates under varying mission constraints. As a result, computing millions of low-thrust trajectories becomes a common requirement that can be computationally prohibitive.

As physicist Lev Landau noted, “the most important part of doing physics is the knowledge of approximation,” a principle that equally applies to this case. To address such computational burden, approximate methods are commonly employed in low-thrust mission design to estimate key performance indicators such as fuel consumption and trajectory feasibility, thereby avoiding the need to solve complex optimal control problems directly. These approximate methods can then be broadly classified into three categories:⁵ analytical approaches, database-driven techniques, and neural network-based models.

Classical analytical techniques—such as Lambert two-impulse solutions,^{6–8} Edelbaum’s formulas and their refinements,⁹ shape-based trajectory methods,^{10,11} and other simplifications of system dynamics^{12–15}—offer closed-form or semi-analytical approximations. While computationally efficient, these methods often suffer from limited accuracy and generality, for instance, being restricted to specific cases such as near-circular or near-coplanar transfers.

Database-driven methods offer high accuracy and efficiency for well-defined problem classes;^{16,17} however, their generalization capability is limited, as performance depends heavily on the coverage and data generation method.¹⁸ Furthermore, in many mission scenarios, task parameters cannot be fully specified in advance, often necessitating frequent database updates, which further limits the adaptability of these methods.

*Postdoctoral Fellow, Department of Aerospace Science and Technology, Politecnico di Milano, 20156 Milan, Italy.

[†]Ph.D. candidate, Department of Aerospace Engineering, Tsinghua University, Beijing, China.

[‡]Professor, Department of Aerospace Science and Technology, Politecnico di Milano, 20156 Milan, Italy; Senior Member AIAA.

In recent years, an increasing number of researchers have turned to machine learning techniques, particularly deep learning, to develop efficient surrogate models for trajectory evaluation. Deep neural networks, as universal function approximators,¹⁹ have been shown capable of capturing complex nonlinear relationships with high accuracy and have been applied to astrodynamics tasks. Zhu et al.²⁰ proposed a learning-based approach to assess low-thrust transfer feasibility and optimal fuel consumption within a specified time window. Their method integrates a classification network for feasibility with a regression network for fuel cost, and experimental results showed that the networks replicated outcomes of expensive trajectory optimizations. Li et al.²¹ used neural networks to approximate three optimal transfer costs: minimum time-of-flight, minimum fuel consumption, and minimum Δv for J_2 -perturbed multi-impulse transfers. More recently, Guo et al.²² targeted time-optimal, multi-leg, multi-asteroid rendezvous missions by embedding a neural network to predict the minimum time-of-flight for transfers between asteroids. To enhance precision on critical trajectories, they emphasized short-duration transfers—limiting the dataset and applying custom loss functions to improve accuracy. Acciarini et al.²³ compared neural and analytical methods, emphasizing their advantages and publicly releasing their training dataset to foster open research in neural network applications. Above studies highlight the effectiveness of neural models as efficient approximators, capable of providing real-time transfer-cost estimates to support complex mission design. Similar deep learning techniques have also been applied successfully in other astrodynamics tasks, such as orbit uncertainty, periodic orbit generation in multi-body dynamics, on-board guidance, and autonomous navigation.^{24–29}

Despite significant progress in recent years, existing research still faces several limitations. First, validation procedures for neural networks rely on self-generated or closely related datasets. Such validation strategies tend to artificially inflate performance measures due to the shared data distribution between training and test sets. Second, existing models are designed for certain mission configurations, such as fixed specific impulse or a narrow thrust range, with insufficient exploration of their generalizability across varied scenarios. A general model that eliminates the need for data regeneration and retraining when addressing new missions would significantly accelerate application. Third, the lack of publicly available models hinders the reproducibility and verifiability of existing studies, thereby limiting the credibility of neural network-based approaches among mission designers.

This paper aims to develop robust neural network models that can be applied across diverse space mission scenarios without the need for regenerating or retraining, thereby significantly reducing time in practical mission design. To this end, the contributions of this paper are threefold. First, a dataset of 100 million samples is constructed using the proposed homotopy ray method, with its distribution aligned with typical mission design requirements. Second, to enhance model generalization, the data is transformed into a self-similar space, enabling the neural network to generalize across varying semi-major axes, inclinations, and central bodies. Third, to the best of the authors’ knowledge, this will be the first publicly available low-thrust trajectory approximator, with implementations provided in C++, Python, and MATLAB. This open access facilitates comparative analysis and encourages broader adoption and trust in neural network methods within the space community.

In addition to strong performance on the internal test set, the proposed approach is further evaluated using a third-party dataset. It is further applied to the multi-asteroid flyby problem from the 4th Global Trajectory Optimization Competition (GTOC4), where it helps to yield the best-known solution to date. Finally, the models are benchmarked against a direct optimal control method via pork-chop plot analysis in an asteroid rendezvous scenario.

DATA GENERATION

This section presents a data generation method for producing large-scale, mission-oriented trajectories. An indirect method is first used to solve individual trajectories, followed by an efficient sampling strategy called the Homotopy Ray Method to generate a diverse set of solutions. In practice, trajectory data is biased toward regions of interest, such as those with low fuel consumption or minimal transfer time. The proposed method concentrates on densely sampling trajectories in these preferred regions.

Two types of data are generated for two separate neural networks: one for fuel-optimal problems and the other for time-optimal problems. Note that trajectory reachability can essentially be framed as a time-optimal control problem. If the given transfer time is shorter than the time-optimal value, the trajectory is unreachable; conversely, it is reachable when the transfer time exceeds this value. When the transfer time equals the time-optimal solution, the trajectory lies at the boundary of the reachable set, as shown in²⁰. Thus, the reachability analysis in²⁰ can be regarded as a special case of time-optimal prediction. Compared to a binary classification of reachable versus unreachable, predicting the minimum transfer time offers richer physical insights. Accordingly, reachability prediction in this work explicitly

corresponds to solving the time-optimal control problem.

Data generation was implemented in C++ and executed on a workstation featuring an AMD EPYC 7452 processor (2.6 GHz, 64 cores) and 256 GB of memory.

Indirect Method for Optimal Control

This subsection describes the indirect method used to solve two classical low-thrust optimal control problems. In both cases, Pontryagin's minimum principle is applied by introducing the costate vector to transform the original optimal control problem into a two-point boundary value problem (TPBVP), which is then solved by a shooting method.

Problem Formulations In all the problem formulations, we assume that the maximum thrust $T_{\max}$ and specific impulse I_{sp} are constants (i.e., independent of the distance between spacecraft to the central body). The state vector $\mathbf{x}$ represents the spacecraft spatial state. $\mathbf{x}_0$ and $\mathbf{x}_f$ denote the initial and final state, respectively.

For the time-optimal problem, the goal is to minimize the transfer time.

Before introducing the indirect method for solving optimal control problems, four practical techniques are used to facilitate the solution process.

1. Instead of using Cartesian coordinates $(\mathbf{r}, \mathbf{v})$ or classical orbital elements $(a, e, i, \Omega, \omega, f)$, the state vector $\mathbf{X}$ is represented using modified equinoctial elements (MEE),³⁰ which are non-singular and well-suited for solving majority of low-thrust trajectory optimization problems.³¹

$$\begin{aligned} p &= a(1 - e^2), & f &= e \cos(\omega + \Omega), & g &= e \sin(\omega + \Omega), \\ h &= \tan(i/2) \cos \Omega, & k &= \tan(i/2) \sin \Omega, & L &= \omega + \Omega + v, \end{aligned} \quad (1)$$

where a , e , i , Ω , ω , and v denote the semi-major axis, eccentricity, inclination, right ascension of the ascending node, argument of perigee, and true anomaly, respectively. Defining the spacial state vector as $\mathbf{x} = [p, f, g, h, k, L]^T$, the dynamics are expressed as

$$\dot{\mathbf{x}} = \mathbf{D}(\mathbf{x}) + \frac{T_{\max}}{m} \mathbf{M}(\mathbf{x}) \boldsymbol{\alpha} u \quad (2)$$

where $\mathbf{D}$ represents the gravitational dynamics, $\boldsymbol{\alpha}$ is the unit vector indicating the thrust direction, u is the normalized thrust magnitude, i.e. $\mathbf{u} = \boldsymbol{\alpha} u$ and $u \in [0, 1]$, $\mathbf{M}$ is the transformation matrix linking the control to the MEE rates. Detailed matrix expressions for $\mathbf{M}$ and $\mathbf{D}$ can be found in Appendix. Readers interested in the various formulations of these expressions may refer to the works of^{27,31,32} for further details.

2. To address the discontinuity of bang-bang control in fuel-optimal problems, a logarithmic homotopy method is applied to smooth the control profile, thereby improving convergence and solution efficiency.³³
3. To facilitate the initialization of the costate vector, a normalization technique is employed by introducing a scalar multiplier λ_0 , such that the magnitude of the initial costate vector $\boldsymbol{\lambda}_0$ is set to 1. This does not alter the nature of the optimal control problem, but improves the ability to estimate reasonable initial guesses for the costate values.³⁴ Therefore, the fuel-optimal cost function is reformulated as

$$J_{\text{fuel}} = \lambda_0 \int_{t_0}^{t_f} L_{\text{fuel}}(\mathbf{x}, \mathbf{u}) dt = \lambda_0 \int_{t_0}^{t_f} \frac{T_{\max}}{I_{\text{sp}} g_0} \{u - \epsilon \ln[u(1 - u)]\} dt \quad (3)$$

with $\epsilon = 1 \times 10^{-5}$ in all generation process, which effectively smooths the control profile and maintains the high fidelity.³³ The time-optimal cost function is similarly defined as

$$J_{\text{time}} = \lambda_0 \int_{t_0}^{t_f} 1 dt \quad (4)$$

4. In the time-optimal problem, the final states are represented as functions of t_f to enhance feasibility and are enforced as equality constraints $\mathbf{x}(t_f) = \mathbf{x}_f(t_f)$, in contrast to the fixed terminal states in the fuel-optimal problem $\mathbf{x}(t_f) = \mathbf{x}_f$.

To summary up, the fuel-optimal and time-optimal problem can be expressed as

Fuel-Optimal Problem:

$$\min_{\mathbf{u}} J_{\text{fuel}} = \lambda_0 \int_{t_0}^{t_f} \frac{T_{\max}}{I_{\text{sp}} g_0} \{u - \varepsilon \ln[u(1-u)]\} dt, \quad (5)$$

$$\text{s.t. } \dot{\mathbf{x}}(t) = \mathbf{D}(\mathbf{x}) + \frac{T_{\max}}{m} \mathbf{M}(\mathbf{x}) \boldsymbol{\alpha} u(t), \quad (6)$$

$$\dot{m}(t) = -\frac{T_{\max} u(t)}{I_{\text{sp}} g_0}, \quad (7)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad m(t_0) = m_0, \quad (8)$$

$$\mathbf{x}(t_f) = \mathbf{x}_f. \quad (9)$$

Time-Optimal Problem:

$$\min_{\mathbf{u}, t_f} J_{\text{time}} = \lambda_0 \int_{t_0}^{t_f} 1 dt, \quad (10)$$

$$\text{s.t. } \dot{\mathbf{x}}(t) = \mathbf{D}(\mathbf{x}) + \frac{T_{\max}}{m} \mathbf{M}(\mathbf{x}) \boldsymbol{\alpha} u(t), \quad (11)$$

$$\dot{m}(t) = -\frac{T_{\max} u(t)}{I_{\text{sp}} g_0}, \quad (12)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad m(t_0) = m_0, \quad (13)$$

$$\mathbf{x}(t_f) = \mathbf{x}_f(t_f). \quad (14)$$

Indirect method and TPBVP Formulation The indirect method is employed to solve the above two optimal control problem, which involves transforming the original problem into a two-point boundary value problem (TPBVP) using Pontryagin's minimum principle.

For the fuel-optimal problem, introducing the costate vector $\boldsymbol{\lambda}_x(t)$, $\lambda_m(t)$ and scalar multiplier λ_0 , the Hamiltonian is constructed as

$$H_{\text{fuel}} = \boldsymbol{\lambda}_x^T \dot{\mathbf{x}} + \lambda_m \dot{m} + \lambda_0 L_{\text{fuel}}(\mathbf{x}, \mathbf{u}) = \frac{T_{\max}}{m} \boldsymbol{\lambda}_x^T \mathbf{M} \boldsymbol{\alpha} u + \boldsymbol{\lambda}_x^T \mathbf{D} - \lambda_m \frac{T_{\max}}{I_{\text{sp}} g_0} \boldsymbol{\alpha} u + \lambda_0 \frac{T_{\max}}{I_{\text{sp}} g_0} \{u - \varepsilon \ln[u(1-u)]\}. \quad (15)$$

The costate dynamics are then given by

$$\begin{aligned} \dot{\boldsymbol{\lambda}}_x &= -\frac{\partial H}{\partial \mathbf{x}} = -\left(\frac{T_{\max}}{m} \boldsymbol{\lambda}_x^T \frac{\partial \mathbf{M}}{\partial \mathbf{x}} \boldsymbol{\alpha} u + \boldsymbol{\lambda}_x^T \frac{\partial \mathbf{D}}{\partial \mathbf{x}}\right) \\ \dot{\lambda}_m &= -\frac{\partial H}{\partial m} = \frac{T_{\max}}{m^2} \boldsymbol{\lambda}_x^T \mathbf{M} \boldsymbol{\alpha} u. \end{aligned} \quad (16)$$

where the detailed expressions of partial derivative of $\mathbf{M}$ and $\mathbf{D}$ to $\mathbf{x}$ can be referred to the Appendix.

Now, the control is unknown in costate dynamics Eq. (16). According to Pontryagin's Minimum Principle, the Hamiltonian must be minimized respect to control, which leads to the optimal thrust direction and magnitude:

$$\boldsymbol{\alpha}^* = -\frac{\mathbf{M}^T \boldsymbol{\lambda}_x}{\|\mathbf{M}^T \boldsymbol{\lambda}_x\|}, \quad (17)$$

$$u^* = \frac{2\epsilon}{\rho + 2\epsilon + \sqrt{\rho^2 + 4\epsilon^2}}. \quad (18)$$

where ρ is the switching function and expressed by:

$$\rho = 1 - \frac{I_{\text{sp}} g_0 \|\mathbf{M}^T \boldsymbol{\lambda}_x\|}{\lambda_0 m} - \frac{\lambda_m}{\lambda_0} \quad (19)$$

For rendezvous problems, the state vector must satisfy the boundary conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f. \quad (20)$$

Since the terminal mass is free in the problem, the transversality condition is provided by

$$\lambda_m(t_f) = 0. \quad (21)$$

In addition, as suggested in,³⁴ the augmented costate vector

$$\boldsymbol{\lambda}(t) \triangleq \begin{bmatrix} \boldsymbol{\lambda}_x(t) \\ \lambda_m(t) \\ \lambda_0 \end{bmatrix}$$

is normalized at the initial time:

$$\|\boldsymbol{\lambda}(t_0)\| = 1. \quad (22)$$

Consequently, the fuel-optimal control problem is converted into the following two-point boundary value problem:

$$\Phi_{\text{fuel}}[\boldsymbol{\lambda}(t_0)] = \begin{bmatrix} \mathbf{x}(t_f) - \mathbf{x}_f \\ \lambda_m(t_f) \\ \|\boldsymbol{\lambda}(t_0)\| - 1 \end{bmatrix} = \mathbf{0}. \quad (23)$$

This TPBVP is solved via a shooting method.³⁵ Once the optimal initial costate $\boldsymbol{\lambda}_x(t_0)$ and final time t_f are determined, the state and costate equations (2) and (16) can be integrated to obtain the complete trajectory and corresponding control law. Finally, the remaining mass is computed as

$$m(t_f) = m_0 - \dot{m} \cdot (t_f - t_0). \quad (24)$$

For the time optimal problem, only the differences are presented because of majority similar derivations. The Hamiltonian is defined as

$$H_{\text{time}} = \frac{T_{\max}}{m} \boldsymbol{\lambda}_x^T \mathbf{M} \boldsymbol{\alpha} u + \boldsymbol{\lambda}_x^T \mathbf{D} - \lambda_m \frac{T_{\max}}{I_{\text{sp}} g_0} \boldsymbol{\alpha} u + \lambda_0. \quad (25)$$

The costate dynamics and the optimal control direction remain the same as in fuel-optimal problem Eqs.(16-17), while the normalized magnitude of optimal control keeps the constant to 1 in time-optimal problem.

With reference to the transversality conditions provided in,³⁶ the optimal final time t_f is determined by

$$H_{\text{time}}(t_f) - \boldsymbol{\lambda}_x(t_f) \cdot \dot{\mathbf{x}}_f = H_{\text{time}}(t_f) - \boldsymbol{\lambda}_L(t_f) \left(\frac{\sqrt{\mu p}}{r^2} \right) = 0. \quad (26)$$

As a result, the time-optimal control problem is converted into the following two-point boundary value problem:

$$\Phi_{\text{time}}[\boldsymbol{\lambda}(t_0); t_f] = \begin{bmatrix} \mathbf{x}(t_f) - \mathbf{x}_f \\ \lambda_m(t_f) = 0 \\ H_{\text{time}}(t_f) - \boldsymbol{\lambda}_L(t_f) \left(\frac{\sqrt{\mu p}}{r^2} \right) \\ \|\boldsymbol{\lambda}(t_0)\| - 1 \end{bmatrix} = \mathbf{0}. \quad (27)$$

Homotopy Ray Method for Large-Scale Data Generation

To tackle the challenge of generating large-scale datasets suitable for trajectory design, the Homotopy Ray Method is proposed. The primary goal is to efficiently generate mission-design-oriented trajectory samples that exhibit both low fuel consumption and close to the boundary of reachable trajectories. This data distribution plays a critical role in numerical trajectory optimization and will be illustrated using a practical example later.

Scaling Law in Low-Thrust Trajectory Approximation The necessity of large-scale datasets is first examined by verifying whether the scaling law applies to low-thrust trajectory estimation. The Scaling Law, originally introduced by OpenAI during research on large language models, reveals that increasing the scale of computation, model size, and dataset can directly lead to performance improvements.³⁷ However, this phenomenon has not been thoroughly explored in the context of low-thrust trajectory optimization. In fact, reference²⁰ suggests that dataset size an efficient trade-off point for dataset scaling.

This paper examines the impact of dataset and model size, using fuel-consumption estimation as a representative case. The neural network inputs, outputs, and training methodology are described in detail later.

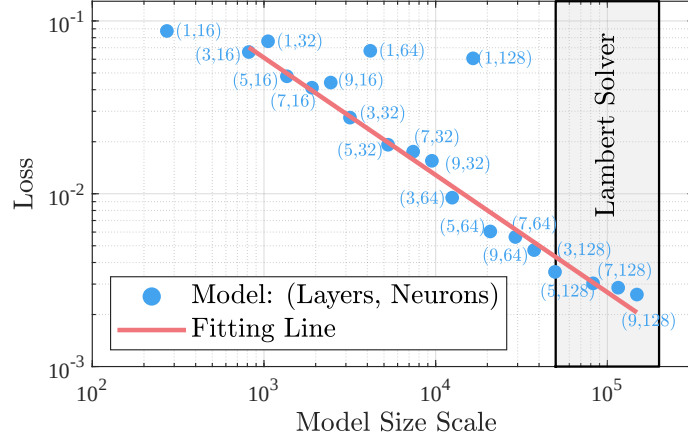


Figure 1. Model size scaling law in low-thrust approximation

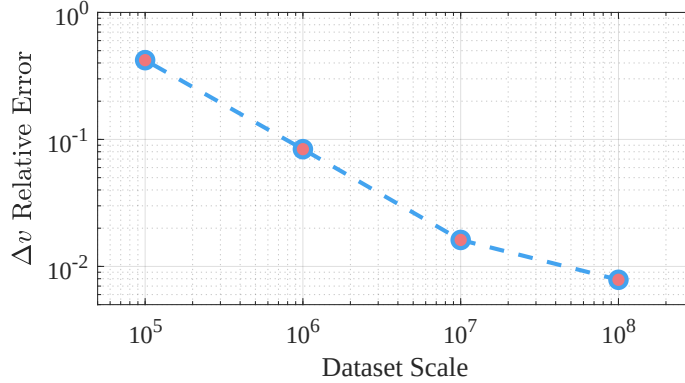


Figure 2. Dataset size scaling law in low-thrust approximation

As shown in Figures 1 and 2, both dataset and model size have a significant impact on performance. Figure 1 illustrates how the loss of model decreases as its size increases, where each blue dot represents a neural network configuration defined by the number of layers and neurons. The shaded area indicates the range in which the model's inference time is on the same order of magnitude comparable to that of the Lambert solver.⁶ Notably, performance exhibits an approximately linear growth on a logarithmic scale, supporting the validity of the Scaling Law in this domain. For each case, multiple model configurations were tested, and the best-performing hyperparameters were selected. During the experiments, it was also observed that the growth in these two dimensions is independent; increasing the dataset size does not affect the choice of model size, and vice versa. These findings underscore the importance of both dataset and model size in low-thrust trajectory optimization.

However, model size cannot be increased indefinitely, as larger models lead to longer inference times, which becomes a bottleneck when used as a submodule within complex optimization frameworks. The model size is selected

to ensure its inference time remains on the same order of magnitude as that of a Lambert solver, as shown in the gray region in Fig. 1, thereby maintaining a balance between predictive efficiency and mission-level computational throughput.

As a result, an appropriate model size—namely, 9 hidden layers with 128 neurons each—is selected, and the subsequent focus shifts to scaling up the dataset. It is worth noting that increasing dataset size only affects training time, not inference time. Since training is conducted once offline and the model is reused subsequently, training time is generally not a concern for end users.

Large-Scale Data Generation This section aims to expand the dataset size to enhance solution quality and convergence robustness. A two-phase method is proposed: it begins with low-fuel trajectories and gradually deforms them toward near-infeasible solutions, utilizing previously obtained effective initial costates to facilitate faster convergence of the shooting method. The method consists of two primary components, as illustrated in Fig. 3: (1) generation of an initial orbit that lies within a Keplerian neighborhood, and (2) homotopic continuation toward the boundary of infeasible trajectories.

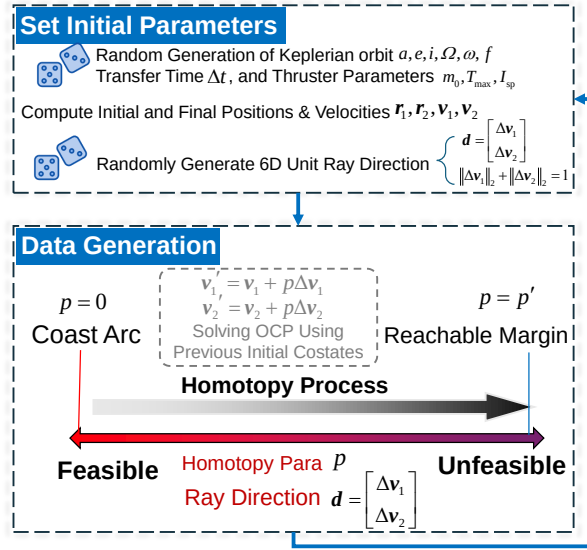


Figure 3. Schematic diagram of the homotopy ray method.

1. Firstly, random initial guesses are generated through a method termed Keplerian Orbital Neighborhood Sampling. Due to the thrust limitations of low-thrust propulsion systems, arbitrarily assigned initial and final states often lead to infeasible transfers. As a result, randomly sampling initial and terminal states results in considerable computational waste and rarely includes fuel-efficient trajectories. An intuitive idea is that, in certain extreme cases, when the initial and final states lie on the same natural Keplerian orbit, the spacecraft can complete the transfer without any propulsion effort, ensuring feasibility regardless of the thrust magnitude.

Under the two-body assumption, such orbits correspond to classical Keplerian trajectories. During data generation, a Keplerian orbit is randomly sampled with orbital elements $(a, e, i, \Omega, \omega, f)$ and a transfer duration Δt , which is then converted into the corresponding initial and final position-velocity vectors $(\mathbf{r}_1, \mathbf{v}_1, \mathbf{r}_2, \mathbf{v}_2)$.

2. Secondly, a homotopy ray method is developed to generate large-scale, mission-oriented trajectory datasets. During the data generation process, the algorithm exploits the concept of homotopy by reusing the initial costate values from previously solved trajectories, and progressively deforms feasible trajectories toward infeasible ones.

Since Keplerian trajectories are guaranteed to be feasible, the process begins with such a trajectory. A random six-dimensional normalized direction vector is generated as $\mathbf{d} = [\Delta \mathbf{v}_1; \Delta \mathbf{v}_2]$, and a homotopy parameter p is used to control the deviation of the boundary velocities from the original Keplerian values. That is, in the

optimal control formulation, the boundary velocities are set as $\mathbf{v}'_1 = \mathbf{v}_1 + p\Delta\mathbf{v}_1$ and $\mathbf{v}'_2 = \mathbf{v}_2 + p\Delta\mathbf{v}_2$. By gradually increasing the homotopy coefficient p , a sequence of trajectory data is obtained along the direction defined by d , until the resulting trajectory becomes infeasible for both problems and extreme large value for time-optimal problem. By repeating first and second procedure with different Keplerian orbits and directions d , a large amount of high-quality trajectory data can be efficiently generated.

The specific algorithmic implementation details can be found in Algorithm 1. For a randomly selected Keplerian orbit, normalized velocity increments $\Delta\mathbf{v}_1$ and $\Delta\mathbf{v}_2$ are generated such that their squared norms sum to one. A small initial scaling factor p is set to 10 m/s, and the scaled increments $p\Delta\mathbf{v}_1$ and $p\Delta\mathbf{v}_2$ are added to the original velocity vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. The initial costate vector $\boldsymbol{\lambda}_0$ is obtained by solving the shooting equation Eq. (23) or Eq. (27) with multiple random guesses. The value of p is then gradually increased, reusing the previously obtained costate vector $\boldsymbol{\lambda}_0$ to solve the shooting problem directly until it becomes unsolvable.

Algorithm 1 Homotopy Ray Method

```

1: repeat
2:   Randomly generate initial position  $\mathbf{r}_1$ , velocity  $\mathbf{v}_1$ , terminal position  $\mathbf{r}_2$ , velocity  $\mathbf{v}_2$ , transfer time  $\Delta t$ , initial mass  $m$ , specific impulse  $I_{sp}$  and thrust  $T_{max}$ 
3:   Randomly generate normalized perturbations  $\Delta\mathbf{v}_1$  and  $\Delta\mathbf{v}_2$  such that  $\|\Delta\mathbf{v}_1\|^2 + \|\Delta\mathbf{v}_2\|^2 = 1$ 
4:   Initialize the homotopy parameter  $p_{start}$  and solve the initial costate  $\boldsymbol{\lambda}_0$  for  $p_{start}$ 
5:   Initialize the empty set of solved trajectories  $\mathcal{S}_{solved}$ 
6:   repeat
7:     Compute perturbed velocities:  $\mathbf{v}'_1 = \mathbf{v}_1 + p\Delta\mathbf{v}_1$ ,  $\mathbf{v}'_2 = \mathbf{v}_2 + p\Delta\mathbf{v}_2$ 
8:     Solve the TPBVP using the current costate guess  $\boldsymbol{\lambda}_0$ 
9:     if the TPBVP is solved successfully then
10:      Update:  $p \leftarrow p + \Delta p$ 
11:      Set  $\boldsymbol{\lambda}_0$  to the newly obtained costate for the next iteration
12:      Store the trajectory in  $\mathcal{S}_{solved}$ 
13:     else
14:      Reduce the step size:  $\Delta p \leftarrow \Delta p/2$ 
15:     end if
16:   until the problem becomes unsolvable:  $\Delta p$  falls below a pre-defined threshold
17:   Select trajectories from  $\mathcal{S}_{solved}$  to move to the Data Sets  $\mathcal{S}$ 
18: until sufficient data is generated
19: Output Data Sets  $\mathcal{S}$ 

```

Note that the unsolvable trajectory corresponds to different physical meanings under the two optimal control formulations. For the fuel-optimal problem, the value p' is defined as the boundary of the reachable set. As discussed at the beginning of this section, the computed time-optimal transfer time t_f equals Δt in this case. For the time-optimal problem, the trajectory often corresponds to a large fuel consumption value, since the engine is at full thrust at all times. This may even result in an eccentricity greater than one and cause numerical difficulties. To avoid such singular cases, an early termination condition is added for the time-optimal problem.

Finally, the reason for imposing an equality constraint on the terminal position Eq. (14) in the time-optimal problem, rather than fixing it at a specific point, is clarified. Based on the homotopy ray method, the initial trajectory must be feasible. Fixing the terminal position may lead to an infeasible trajectory. This is because, in the time-optimal problem, the engine operates at full thrust continuously. If the initial transfer time exceeds the time required to consume all the propellant, the trajectory will naturally fail to reach the terminal point. To mitigate this risk, the terminal position equality constraint is formulated to ensure the feasibility of the initial trajectory.

Data Distribution Characteristics The homotopy ray method not only accelerates data generation, but also reveals an approximately monotonic relationship between the velocity increment and $\Delta\mathbf{v}_1$, $\Delta\mathbf{v}_2$, as shown in Fig. 4. Although this is just a specific case, the figure illustrates the variation in fuel consumption Δv as $\Delta\mathbf{v}_1$ changes while keeping $\Delta\mathbf{v}_2 = 0$, demonstrating the monotonic property of the velocity increment.

This monotonic behavior also motivates the use of $\Delta\mathbf{v}_1$ and $\Delta\mathbf{v}_2$ as input features instead of velocity vectors directly, as they better capture the underlying structure of the problem. This is consistent with observations reported

in,^{20,21,23} which used Lambert solver solutions as the inputs to improve network performance. Details of the input features are provided in the next section.

Another advantage of this approach lies in the adaptively denser sampling of low-fuel-consumption trajectories and unreachable regions, which aligns well with the requirements of practical trajectory optimization. Such a sampling distribution is difficult to achieve through random generation. This advantage will be further demonstrated in the result section.

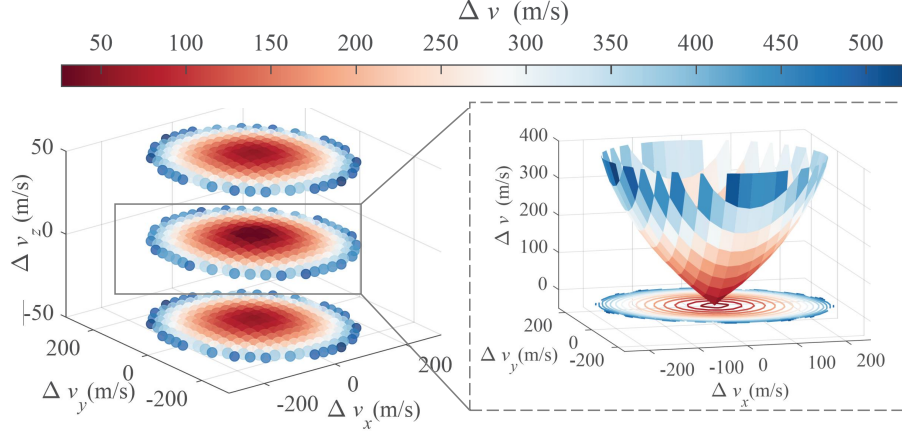


Figure 4. Graph illustrating the relationship between the speed increment and the corresponding variation in departure velocity.

INPUT AND OUTPUT DATA ANALYSIS

This section aims to reduce input dimensionality through lossless transformations, in order to improve model performance. To this end, dimensionality reduction is first introduced in a self-similar space. Next, the model's performance under different input configurations is evaluated. Finally, the input features of the two models along with the intermediate data processing are summarized.

Specifically, the input parameters for Eq. (5) and Eq. (10) include the following: $\mathbf{r}_1, \mathbf{v}_1, \mathbf{r}_2, \mathbf{v}_2, \Delta t, m_0, T_{\max}, I_{\text{sp}}, \mu$. The total input dimension of this formulation is 17. Note that the gravitational acceleration at sea level g_0 , which is typically fixed at 9.80665 m/s^2 , remains unchanged. Minor discrepancies in numerical precision can be compensated by adjusting the specific impulse I_{sp} , since g_0 and I_{sp} always appear as a product.

This section aims to leverage physical insights and empirical analysis to identify a minimal set of independent variables. The goal is to determine the optimal combination of input features that preserves model performance.

Dimensionality Reduction in Self-Similar Space

This subsection introduces the concept of self-similar mapping, which leverages the inherent rotational invariance and dimensional invariance of the problem. It is important to note that this process involves data preprocessing to reduce the input dimensionality of the neural network, rather than data augmentation. Since these transformations are physically lossless, such dimensionality reduction is both reasonable and justified.

Rotational Invariance A central gravitational field exhibits inherent rotational symmetry, meaning that the system's dynamics depend solely on the relative distance to the center rather than on its absolute orientation in space. In other words, if both the initial and final positions and their corresponding velocity vectors are rotated by the same angle, their relative geometrical relationships remain unchanged. The physical constraints that govern trajectory optimization also remain unaffected.

Consequently, the optimal solution, whether minimizing the fuel consumption or the transfer time, remains invariant under coordinate system rotation. This rotational invariance offers flexibility in coordinate selection for trajectory planning and guarantees that the resulting solution is independent of the chosen reference frame.

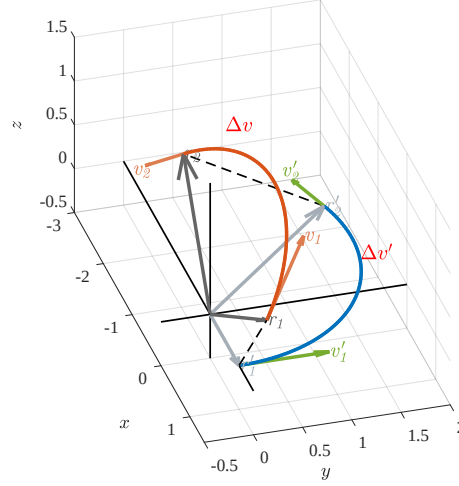


Figure 5. The diagram illustrates the rotational invariance.

As illustrated in Fig. 5, the system dynamic equations remain invariant under coordinate rotation in a dimensionless space. A rotation applied to the initial and final states $\mathbf{r}_1, \mathbf{v}_1, \mathbf{r}_2, \mathbf{v}_2$ yields the transformed states $\mathbf{r}'_1, \mathbf{v}'_1, \mathbf{r}'_2, \mathbf{v}'_2$. Since the dynamic equations are invariant under rotation, the velocity increment of the transformed problem is equal to that of the original one, namely, $\Delta v = \Delta v'$. The same conclusion applies to time-optimal solutions.

There are several strategies for coordinate rotation. In this study, the rotation is performed by aligning the departure position with the x-axis and the arrival position with the xy-plane. This transformation effectively reduces the problem dimensionality by three without any loss of physical fidelity.

Since transfer trajectories in different planes can be expressed in a unified formulation, this approach improves the generality and consistency of model prediction.

The detailed procedure is outlined in Algorithm 2. The core idea involves rotating the initial position onto the x-axis, followed by a secondary rotation about the x-axis to place the final position within the xy-plane. Among the two possible orientations resulting from the second rotation, the one that yields a positive projection of the initial velocity vector onto the y-axis is selected.

Algorithm 2 Rotation $\mathbf{r}_1, \mathbf{v}_1, \mathbf{r}_2, \mathbf{v}_2$

Require: Initial states: $\mathbf{r}_1, \mathbf{v}_1$; Final states: $\mathbf{r}_2, \mathbf{v}_2$

Ensure: Rotated states: $\mathbf{r}'_1, \mathbf{v}'_1; \mathbf{r}'_2, \mathbf{v}'_2$

- 1: Compute $\mathbf{n} \leftarrow \text{Cross}(\mathbf{r}_1, \mathbf{r}_2)$
 - 2: Normalize: $\mathbf{n} \leftarrow \mathbf{n} / \|\mathbf{n}\|$
 - 3: Remove normal component from $\mathbf{v}_1$: $\mathbf{v}_1 \leftarrow \mathbf{v}_1 - (\text{Dot}(\mathbf{v}_1, \mathbf{n}))\mathbf{n}$
 - 4: Set target direction: $\mathbf{e}_x \leftarrow [1, 0, 0]$
 - 5: Compute rotation matrix $R_1 \leftarrow \text{RotationMatrix}(\mathbf{r}_1, \mathbf{e}_x)$
 - 6: Obtain intermediate states: $\mathbf{r}_1^* \leftarrow R_1 \mathbf{r}_1, \quad \mathbf{v}_1^* \leftarrow R_1 \mathbf{v}_1, \quad \mathbf{r}_2^* \leftarrow R_1 \mathbf{r}_2, \quad \mathbf{v}_2^* \leftarrow R_1 \mathbf{v}_2$
 - 7: Compute correction angle: $\phi \leftarrow -\arctan 2(v_{1z}^*, v_{1y}^*)$
 - 8: Compute $R_x \leftarrow \text{RotateX}(\phi)$
 - 9: Compose total rotation: $R_{\text{total}} \leftarrow R_x R_1$
 - 10: Obtain final states: $\mathbf{r}'_1 \leftarrow R_{\text{total}} \mathbf{r}_1, \quad \mathbf{v}'_1 \leftarrow R_{\text{total}} \mathbf{v}_1, \quad \mathbf{r}'_2 \leftarrow R_{\text{total}} \mathbf{r}_2, \quad \mathbf{v}'_2 \leftarrow R_{\text{total}} \mathbf{v}_2$
 - 11: **return** $\mathbf{r}'_1, \mathbf{v}'_1, \mathbf{r}'_2, \mathbf{v}'_2$
-

Dimensional Invariance This subsection discusses the process of non-dimensionalization, a commonly used technique in trajectory optimization. A typical approach involves normalizing the length and time units by defining 1 AU and 1 year as the respective reference units.³⁴ Another widely used method scales the gravitational parameter so that

the solar gravitational constant is normalized to unity.²⁷

In this work, a similar normalization scheme is adopted, as shown in Fig. 6. From the perspective of neural network input design, the length unit is defined as the magnitude of the departure position, which normalizes it to unity. Additionally, the gravitational parameter is scaled to unity. Since both the length and gravitational parameter become fixed after normalization, this further reduces two input dimensions for the neural network.

Dimensionless Transformation

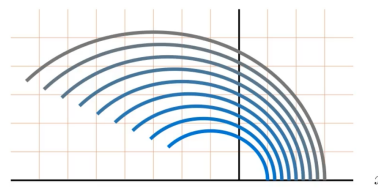
Given: $L_{\text{old}}, \mu_{\text{old}}$

Units: $L_0 = L_{\text{old}}, T_0 = \frac{L_0^3}{\mu_{\text{old}}}, V_0 = \frac{L_0}{T_0}, A_0 = \frac{L_0}{T_0^2}$

New Values: $t_{\text{new}} = \frac{t_{\text{old}}}{T_0}, v_{\text{new}} = \frac{v_{\text{old}}}{V_0}, a_{\text{new}} = \frac{a_{\text{old}}}{A_0}, L_{\text{new}} = 1, \mu_{\text{new}} = 1$

Original Equations

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= -\frac{\mu}{r^3} \mathbf{r} + \frac{T_{\text{max}} u \boldsymbol{\alpha}}{m} \\ \dot{m} &= -\frac{T_{\text{max}} u}{I_{\text{sp}} g_0} \end{aligned} \quad \text{Dimensionless}$$



$$\tilde{t} = \frac{t}{T_0}, \quad \tilde{\mathbf{r}} = \frac{\mathbf{r}}{L_0}, \quad \tilde{\mathbf{v}} = \frac{\mathbf{v}}{L_0/T_0}, \quad \tilde{m} = \frac{m}{m_0}$$

$$\begin{aligned} \frac{d\tilde{\mathbf{r}}}{d\tilde{t}} &= \tilde{\mathbf{v}} \\ \frac{d\tilde{\mathbf{v}}}{d\tilde{t}} &= -\frac{1}{\tilde{r}^3} \tilde{\mathbf{r}} + \beta u \boldsymbol{\alpha} \frac{1}{\tilde{m}} \\ \frac{d\tilde{m}}{d\tilde{t}} &= -\gamma u \end{aligned}$$

$$\text{Where } \beta = \frac{T_{\text{max}} T_0^2}{L_0}, \quad \gamma = \frac{T_{\text{max}} T_0}{I_{\text{sp}} g_0 m_0}.$$

Figure 6. The diagram illustrates the dimensional invariance.

Consequently, the family of trajectories illustrated in the lower left of Fig. 6 can be treated as a single representative case in the normalized space, further enhancing the generalization capability of the proposed model. Furthermore, since only the ratio between the maximum thrust T_{max} and the initial mass m_0 affects the trajectory, the initial acceleration $a_s = T_{\text{max}}/m_0$ is introduced to further reduce the dimensionality of the corresponding input. In summary, the original problem involves 17 independent variables. After applying rotational invariance (3 dimensions) and non-dimensionalization (3 dimensions), the independent input dimensionality is reduced to eleven.

Comparison of Different Types of Inputs

The previous subsection discussed the minimal dimensionality of input variables. This subsection further presents a performance comparison of models under different input configurations.

Table 1. Input Parameters and Descriptions

Input	Description
coe	$\text{coe}_1, \text{coe}_2$
mee	$\text{mee}_1, \text{mee}_2$
rv	$\mathbf{r}_1, \mathbf{v}_1, \mathbf{r}_2, \mathbf{v}_2$
rv _{rotate}	$\mathbf{r}'_1, \mathbf{v}'_1, \mathbf{r}'_2, \mathbf{v}'_2$
Ref. ²³	$\mathbf{r}_1 - \mathbf{r}_2, \mathbf{v}_1 - \mathbf{v}_2, \Delta \mathbf{v}_1, \Delta \mathbf{v}_2$
Lambert _{cart}	$e, f, \Delta \mathbf{v}_1, \Delta \mathbf{v}_2$ (in $[x, y, z]^T$)
Lambert _{sph}	$e, f, \Delta \mathbf{v}_1, \Delta \mathbf{v}_2$ (in $[r, \theta, \varphi]^T$)
t_{Lambert}	$(\ \Delta \mathbf{v}_1\ + \ \Delta \mathbf{v}_2\)/a_s$

Several input configurations are defined, as summarized in Table 1. There are three noteworthy aspects regarding these configurations: Firstly, the transfer time Δt , initial acceleration a_s , and specific impulse I_{sp} are shared across all input configurations and are thus omitted from the table. Secondly, all input variables are non-dimensionalized. Variables that become constant after self-similar transformation are excluded from the final input set to avoid confusion.

For example, under the $\text{rv}_{\text{rotate}}$ configuration, the departure position $\mathbf{r}'_1$ is always mapped to a unit vector $[1, 0, 0]^T$ and is therefore omitted. Thirdly, all angular quantities are encoded using the standard sin/cos formulation to preserve periodicity and facilitate learning.

Table 2. Δv Prediction Performance Comparison of Different Input Configurations

Input	Train Loss	Test Loss	Test Loss	e_{abs} , m/s	e_{rel} , (%)
coe	0.0012	0.0034	0.0032	3.76	6.21
mee	0.0010	0.0030	0.0034	3.94	6.49
rv	0.0096	0.0120	0.0113	13.21	27.52
$\text{rv}_{\text{rotate}}$	0.0005	0.0026	0.0024	2.83	4.71
Ref. ²³	0.0001	0.0021	0.0019	2.26	1.30
Lambert _{cart}	0.0002	0.0019	0.0017	2.00	3.58
Lambert _{sph}	0.0002	0.0020	0.0018	2.11	1.07
$\text{rv}_{\text{rotate}}, \text{Lambert}_{\text{sph}}$	0.0002	0.0021	0.0019	2.19	1.18

The performance of different input configurations on two separate networks, one for Δv prediction and the other for Δt prediction, is summarized in Tables 2 and 3. Several key observations can be made:

1. Without incorporating Lambert solver information, the $\text{rv}_{\text{rotate}}$ configuration clearly outperforms other representations, suggesting that dimensionality reduction substantially improves model performance. Among the commonly used orbital element forms, MEE shows the best performance, while the position–velocity representation performs the worst.
2. With Lambert solver information included, performance improves across all configurations, indicating that this information is highly beneficial.
3. Comparing configurations that incorporate Lambert information reveals that models using only independent variables may suffice to achieve competitive performance. Adding redundant information—such as combining $\text{rv}_{\text{rotate}}$ with Lambert inputs—offers no further benefit and may even slightly degrade it, possibly due to increased input dimensionality making training more difficult. This indirectly supports the effectiveness of dimensionality reduction in input design.
4. Across all representations, spherical coordinates consistently outperform Cartesian position–velocity representations, possibly due to their stronger physical interpretability.

Table 3. Δt Prediction Performance Comparison of Different Input Configurations

Input	Train Loss	Test Loss	Test Loss	e_{abs} , days	e_{rel} , (%)
coe	0.00025	0.00124	0.00070	0.3093	0.300
mee	0.00021	0.00065	0.00056	0.2497	0.235
rv	0.00341	0.00324	0.00414	1.8325	2.046
$\text{rv}_{\text{rotate}}$	0.00010	0.00043	0.00042	0.1842	0.119
Ref. ²³	0.00006	0.00038	0.00038	0.1682	0.100
Lambert _{cart}	0.00006	0.00039	0.00039	0.1719	0.102
Lambert _{sph}	0.00008	0.00036	0.00038	0.1674	0.100
$\text{rv}_{\text{rotate}}, t_{\text{Lambert}}$	0.00006	0.00043	0.00038	0.1670	0.107
Lambert _{sph}, t_{\text{Lambert}}}	0.00004	0.00038	0.00034	0.1504	0.094
$\text{rv}_{\text{rotate}}, \text{Lambert}_{\text{sph}}, t_{\text{Lambert}}$	0.00004	0.00038	0.00035	0.1533	0.095

Based on the above findings, the Lambert_{sph} configuration is selected as the final input configuration. Even though introducing t_{Lambert} leads to further performance improvement in the time-optimal problem, Lambert_{sph} is ultimately selected to ensure consistency across both models.

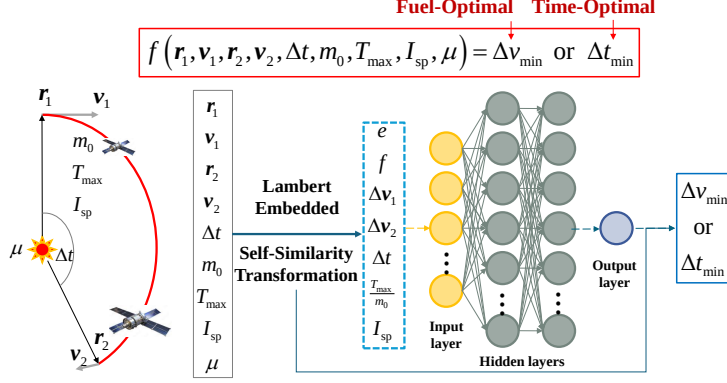


Figure 7. The input and output of the prediction model.

Finally, the overall input–output pipeline is illustrated in Fig. 7. The raw mission parameters are first processed through rotational invariance and non-dimensionalization, reducing the problem dimensionality. Then, a Lambert solver computes the two-impulse transfer characteristics, which are used as inputs to the neural network. The network outputs dimensionless results, which are subsequently rescaled using the original velocity and time units to obtain the final velocity increment and time-optimal transfer duration.

NEURAL NETWORK TRAINING

This section investigates how training-related parameters affect the predictive performance of deep neural networks. The discussion is organized into two parts. The first part evaluates various network architectures, each tested under multiple hyperparameter settings, with only the best result reported per architecture. The second part analyzes the sensitivity of the optimal architecture to individual hyperparameters.

The dataset is split into training, validation, and test sets using a 96%/2%/2% partition. Each model is trained for 10,000 epochs, with the validation set evaluated after each epoch. The model with the best validation performance is retained to prevent overfitting. The retained model is then evaluated on the test set to assess generalization capability. To efficiently evaluate a large number of parameter combinations, these experiments are conducted on a dataset containing 100,000 samples.

All training is performed on a single NVIDIA RTX 4090 GPU (24 GB) with access to 6 CPU cores and 60 GB of memory. The training is implemented using Python 3.12 and PyTorch 2.5.1, with CUDA version 12.4.

Neural Network Structure

Table 4. Ranges of Hyperparameters

Hyperparameter	Range
n_{layer}	1, 3, 5, 7, 9
n_{neuron}	8, 16, 32, 64, 128
η	0.1, 0.01, 0.001, 0.0001
wd / η	0.1, 0.01
B	3200, 6400, 12800, 25600, 51200

The architecture of the network is a critical factor influencing predictive performance. In this subsection, the number of hidden layers (n_{layer}) and the number of neurons per hidden layer (n_{neuron}) are treated as tunable hyperparameters, with their ranges provided in Table 4. Additional hyperparameters are introduced in the next subsection. Each hidden layer uses the ReLU activation function, and a linear activation is applied at the output layer.

The performance of the Δv and Δt models under different network n_{layer} and n_{neuron} is shown in Tables 5 and 6, respectively. Each architecture is fine-tuned over other hyperparameters, and only the best-performing result is reported. The test loss of the Δv model has already been presented in Fig. 1. Table 6 also presents the performance of the Δt model across various architectural configurations. It can be observed that the performance of the Δt model continues to improve as the number of layers and neurons increases, satisfying the scaling law as well.

Table 5. Δv Performance for Different Layer and Neuron Configurations

Layers	Neurons = 8			Neurons = 16			Neurons = 32			Neurons = 64			Neurons = 128		
	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s
1	0.090	0.092	156	0.085	0.087	154	0.073	0.076	155	0.063	0.067	154	0.057	0.061	155
3	0.073	0.072	223	0.058	0.066	223	0.026	0.028	225	0.006	0.009	225	0.001	0.004	225
5	0.070	0.074	291	0.044	0.048	292	0.017	0.019	294	0.004	0.006	293	0.001	0.003	295
7	0.070	0.073	360	0.038	0.041	363	0.013	0.018	363	0.003	0.006	362	0.001	0.003	363
9	0.070	0.070	425	0.040	0.044	426	0.012	0.016	431	0.002	0.005	430	0.001	0.003	431

Table 6. Δt Performance for Different Layer and Neuron Configurations

Layers	Neurons = 8			Neurons = 16			Neurons = 32			Neurons = 64			Neurons = 128		
	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s	Train Loss	Test Loss	Time, s
1	0.154	0.156	157	0.137	0.146	155	0.125	0.126	156	0.081	0.084	155	0.0352	0.0385	156
3	0.131	0.129	224	0.091	0.096	224	0.031	0.036	225	0.004	0.005	225	0.0007	0.0012	226
5	0.131	0.136	291	0.072	0.083	290	0.013	0.017	293	0.002	0.003	291	0.0005	0.0011	292
7	0.125	0.126	356	0.066	0.072	355	0.009	0.011	358	0.001	0.002	357	0.0006	0.0012	358
9	0.117	0.121	421	0.055	0.062	422	0.007	0.010	427	0.001	0.002	426	0.0004	0.0009	426

Hyperparameter Tuning

During hyperparameter tuning, the Δv and Δt models are trained by jointly optimizing three key hyperparameters: the learning rate (η), the weight decay ratio (w_d/η), and the batch size (B).

The AdamW optimizer, a variant of Adam that incorporates weight decay, is adopted for its robustness in training.^{38,39} The learning rate η and the weight decay ratio w_d/η are treated as tunable hyperparameters, with their respective search ranges summarized in Table 4. A OneCycleLR scheduler is employed to dynamically adjust the learning rate during training.⁴⁰

All hyperparameter tuning experiments are conducted using the same network architecture, consisting of 9 hidden layers with 128 neurons each. The test results are reported in Tables 7 and 8. Given the extensive number of tested cases, only the best-performing settings (highlighted in bold) and corresponding ablation results, where a single hyperparameter is varied while the others are fixed at their optimal values, are shown for clarity.

The results indicate that the learning rate η has the greatest impact on model performance, while the influence of the weight decay ratio w_d/η is comparatively minor. Due to the low input dimensionality and moderate network size, GPU parallelization enables efficient full-batch processing. As a result, training time scales inversely with the batch size B , and smaller batches generally lead to better performance. This introduces a trade-off between computational efficiency and generalization when training on large datasets. In this study, a batch size of $B = 6400$ is adopted to accommodate training on datasets as large as 100 million samples.

Finally, these results further demonstrate that careful hyperparameter selection remains essential. Poor choices, such as an excessively large learning rate, can still cause orders-of-magnitude degradation in model performance.

RESULTS

This section presents the final training results of the proposed model. It further evaluates the model through three complementary tasks: validation on third-party datasets, application to the GTOC4 multi-flyby asteroid trajectory design problem, and porkchop plot generation for asteroid rendezvous analysis.

Table 7. Hyperparameter tuning for Δv

η	wd/ η	B	Train Loss	Test Loss	Time, s
0.1	0.01	3200	0.0219	0.0226	424
0.001	0.01	3200	0.0140	0.0160	424
0.0001	0.01	3200	0.0223	0.0247	421
0.01	0.01	3200	0.0062	0.0079	424
0.01	0.1	3200	0.0060	0.0082	424
0.01	0.01	6400	0.0067	0.0092	221
0.01	0.01	12800	0.0081	0.0103	115
0.01	0.01	25600	0.0090	0.0125	62
0.01	0.01	51200	0.0103	0.0127	37

Table 8. Hyperparameter tuning for Δt

η	wd/ η	B	Train Loss	Test Loss	Time, s
0.1	0.1	3200	0.0194	0.0212	426
0.001	0.1	3200	0.0092	0.0114	425
0.0001	0.1	3200	0.0153	0.0183	427
0.01	0.1	3200	0.0033	0.0044	426
0.01	0.01	3200	0.0033	0.0045	426
0.01	0.1	6400	0.0035	0.0045	223
0.01	0.1	12800	0.0043	0.0053	115
0.01	0.1	25600	0.0051	0.0068	62
0.01	0.1	51200	0.0063	0.0080	37

The objective is to demonstrate the model’s generalization ability, applicability to real mission design, and computational efficiency. Validation on third-party datasets assesses the model’s generalization under varying data distributions and verifies the effectiveness of the proposed data generation methodology. The GTOC4 multi-flyby design task serves as a benchmark to evaluate the model’s integration within trajectory optimization, highlighting its accuracy in sequential mission planning. Finally, the model is applied to porkchop plot generation for asteroid rendezvous, enabling direct comparison in computational efficiency and usability against traditional optimal control methods.

Training Result

The final model is trained on a dataset containing 100 million samples, where generating each 1-million-sample dataset takes approximately three days. The training hardware is the same, and training on the full 100-million-sample dataset also takes around three days to complete.

To assess the applicability and performance of the Δv model, we compare our results against those reported in existing studies, as summarized in Table 9. As shown, our model outperforms previous approaches in terms of both mean absolute error (MAE) and mean relative error (MRE), whether evaluated by Δv or the final mass m_f . Specifically, the model achieves an average relative error of 0.78% for Δv . It should be noted that all comparative results are derived from the respective published test sets, which may overestimate performance due to the likely similarity between training and testing data distributions in those works.

More significantly, Table 9 also highlights the broad applicability of our model. Unlike prior methods that impose constraints on orbital elements—such as semi-major axis (a), eccentricity (e), or inclination (i)—our model is general-purpose and supports arbitrary celestial bodies. Furthermore, the propulsion parameters (a_s , I_{sp}) span a wide operational range, covering most practical propulsion configurations. This versatility makes the model suitable for a wide range of mission scenarios, offering substantial convenience compared to earlier models that are often tailored to specific tasks. Mission designers and planetary scientists can directly apply our model without the need for retraining

Table 9. Δv Model Performance Compared to Existing Models

Parameter	Δv Model			
	Zhu ²⁰	Li ²¹	Acciarini ²³	This paper
a , AU	2.0 $\sim$ 3.0	2.0 $\sim$ 3.5	0.9 $\sim$ 4.0	any
e	0 $\sim$ 0.4	0 $\sim$ 0.1	0 $\sim$ 0.48	0 $\sim$ 1.0
i , deg	0 $\sim$ 20	0 $\sim$ 10	0 $\sim$ 30	any
Δt , n	0 $\sim$ 0.48	0 $\sim$ 0.3	0 $\sim$ 12.65	0 $\sim$ 0.99
a_s (min), m/s ²	1.5×10^{-4}	1.5×10^{-4}	7.5×10^{-5}	2.5×10^{-6} [a]
a_s (max), m/s ²	3.8×10^{-4}	3.0×10^{-4}	8.6×10^{-4}	1.2×10^{-2} [a]
I_{sp} , s	3000	3000	4000	700 $\sim$ 9000[a]
Δv MAE, m/s	/	/	96.16	3.38
Δv MRE, %	/	/	2.98%	0.78%
mf MAE, kg	/	/	/	0.42
mf MRE, %	0.37%	0.50%	/	0.014%

^a The value is calculated at 1 AU.

or regenerating datasets. Nevertheless, it is important to note that the current model supports only single-revolution transfers; extending it to multi-revolution cases remains a key direction for future work.

Table 10. Δt Model Performance

Parameter	Δt Model
a , AU	any
e	0 $\sim$ 1.0
i , deg	any
Δt , n	0 $\sim$ 0.99
a_s (min), m/s ²	2.5×10^{-6} [a]
a_s (max), m/s ²	1.2×10^{-2} [a]
I_{sp} , s	700 $\sim$ 9000[a]
Δt min MAE, days	2.56
Δt min MRE, %	0.63%

^a The value is calculated at 1 AU.

For the Δt model, the evaluation results are reported in Table 10. Due to the use of equality constraints in the time-optimal control formulation, only our model’s results are presented. The model achieves an average relative error of 0.63%, demonstrating strong capability in assessing transfer feasibility.

Third-Party Dataset Validation

This subsection discusses the evaluation of the model’s generalization capability. As discussed in the previous subsection, evaluating the model using its own test set inevitably leads to overestimation, because the test and training sets share the same data distribution. Therefore, a third-party dataset was used for validation. The dataset, provided by,²³ contains one million samples and is currently the only publicly available dataset in this field, representing a significant contribution to the community. The authors also released their model’s prediction results on this dataset, enabling direct comparison. The performance of our model was evaluated on the same dataset and compared with their results. The same evaluation metrics, MAE and MRE, were employed.

As shown in Fig. 8, the model’s performance across various Δv was evaluated using a moving average window method. Specifically, the ground-truth Δv values were first sorted in ascending order, and the sorted dataset was then divided into consecutive windows, each containing 1000 samples. For each window, the average of the Δv values and the corresponding MAE were computed. It can be observed that the dataset from²³ does not contain samples with Δv less than 200 m/s. This is likely attributed to their data generation method, which relies on randomly selecting initial

and final positions and velocities, making low-fuel-consumption samples rare. As a result, their model performs better for Δv values above 1000 m/s compared to lower values.

In contrast, the proposed model performs well in this regime, which underscores the importance of the mission design-oriented data generation approach described in the previous section, and demonstrates the effectiveness of the homotopy ray-based data generation method. Moreover, the proposed model achieves better overall performance than that of,²³ even when evaluated on their dataset. This demonstrates the strong generalization ability of the proposed model and further suggests the potential of the scaling law.

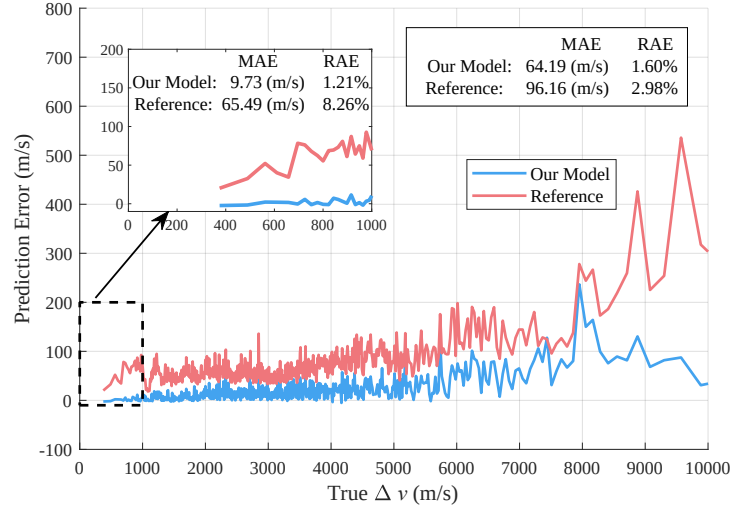


Figure 8. Validation of the third-party dataset.

Multi-Flyby Asteroid Mission Design

This section evaluates the proposed model on a trajectory optimization task. Specifically, the GTOC4 problem is used as a benchmark. This problem involves multiple asteroid flybys, and detailed descriptions can be found in.⁴¹ For comparative analysis, two sets of results are considered. One is derived from the state-of-the-art solution to the GTOC4 problem, proposed by the University of Jena. This trajectory includes flybys of 49 asteroids over a ten-year mission, culminating in a final rendezvous. The proposed model predicts the required velocity increments using segment-wise flyby data extracted from this mission plan. This scenario is referred to as Result 1. Result 2 is generated through neural network-based approximation, which optimizes the timing and relative velocity of each flyby. This results in a more fuel-efficient trajectory, demonstrating the effectiveness of the neural approximators. Since the optimization procedure is beyond the scope of this study, its details are omitted here.

As shown in Fig. 9, the model demonstrates strong predictive performance on the GTOC4 task. This highlights the effectiveness and generalization capability of the proposed model. In the optimal GTOC4 trajectory, some flyby segments require less than 100 m/s of velocity increment. This underscores the importance of including such low-thrust trajectories in the training dataset.

Mission Analysis

This subsection introduces how the model can be applied to a practical mission analysis scenario. The benchmark mission involves a cube satellite launched from Earth to rendezvous with an asteroid, equipped with a low-thrust engine. The launch vehicle provides a maximum departure velocity relative to Earth of 4 km/s. In this paper, asteroid 2012 LA is chosen to show performance, as shown in Figure 10.

The plot demonstrates that the neural network can approximate the structure of the porkchop plots with high accuracy (typically within 10%), thereby validating the effectiveness of the proposed model.

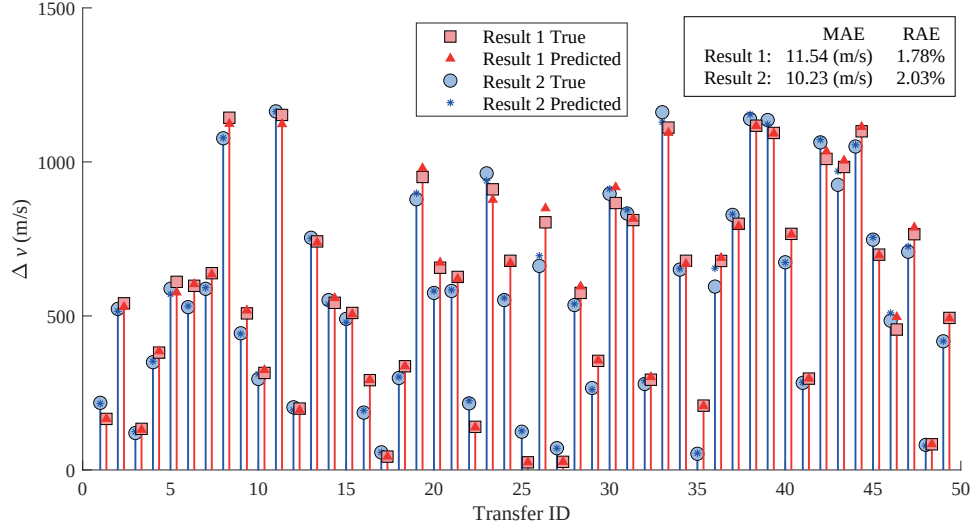


Figure 9. Validate in GTOC4 problem results.

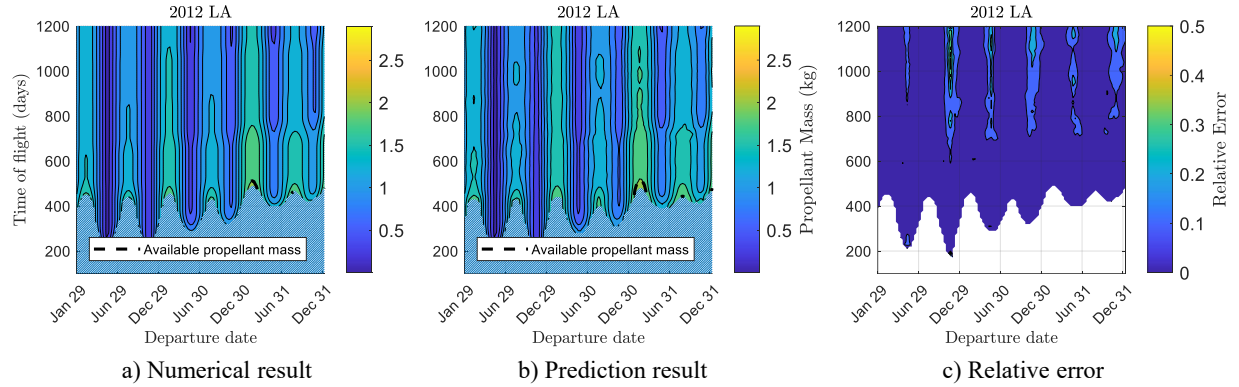


Figure 10. Porkchop plot for the 2012 LA Asteroid.

CONCLUSION

This paper presents a neural network-based approach to efficiently estimate fuel consumption and transfer reachability for low-thrust trajectories. The proposed models extend the existing applicability of low-thrust trajectory approximation, thereby offering adaptability across a variety of mission design scenarios without necessitating retraining for each specific case. This improvement is driven by two insights. First, at the data level, an approximate scaling law for low-thrust trajectory approximation is validated, supporting the generation of an extensive dataset of over 100 million trajectory samples. Second, at the astrodynamics level, the inherent symmetries in the fuel- and time-optimal control problem are exploited to reduce the dimensionality of the corresponding dynamical system. The generalizability, accuracy, and computational efficiency of the proposed neural network models are conclusively demonstrated through validation using a third-party dataset, the GTOC4 mission design problem, and a pork-chop mission analysis scenario.

REFERENCES

- [1] M. D. Rayman, P. A. Chadbourne, J. S. Culwell, and S. N. Williams, “Mission Design for Deep Space 1: A Low-Thrust Technology Validation Mission,” *Acta Astronautica*, Vol. 45, Aug. 1999, pp. 381–388, 10.1016/S0094-5765(99)00157-5.

- [2] J. Kawaguchi, A. Fujiwara, and T. Uesugi, “Hayabusa—Its Technology and Science Accomplishment Summary and Hayabusa-2,” *Acta Astronautica*, Vol. 62, May 2008, pp. 639–647, 10.1016/j.actaastro.2008.01.028.
- [3] J. Benkhoff, J. van Casteren, H. Hayakawa, M. Fujimoto, H. Laakso, M. Novara, P. Ferri, H. R. Middleton, and R. Ziethe, “BepiColombo—Comprehensive Exploration of Mercury: Mission Overview and Science Goals,” *Planetary and Space Science*, Vol. 58, Jan. 2010, pp. 2–20, 10.1016/j.pss.2009.09.020.
- [4] D. Y. Oh, S. Collins, T. Drain, W. Hart, T. Imken, K. Larson, D. Marsh, D. Muthulingam, J. S. Snyder, D. Trofimov, *et al.*, “Development of the Psyche Mission for NASA’s Discovery Program,” 2019.
- [5] Z. Zhang, N. Zhang, Z. Chen, F. Jiang, H. Baoyin, and J. Li, “Global Trajectory Optimization of Multispacecraft Successive Rendezvous Using Multitree Search,” *Journal of Guidance, Control, and Dynamics*, Vol. 47, No. 3, 2024, pp. 503–517, 10.2514/1.G007764.
- [6] D. Izzo, “Revisiting Lambert’s Problem,” *Celestial Mechanics and Dynamical Astronomy*, Vol. 121, Jan. 2015, pp. 1–15, 10.1007/s10569-014-9587-y.
- [7] R. M. Woollands, A. Bani Younes, and J. L. Junkins, “New Solutions for the Perturbed Lambert Problem Using Regularization and Picard Iteration,” *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 9, 2015, pp. 1548–1562, 10.2514/1.G001028.
- [8] R. P. Russell, “Complete Lambert Solver Including Second-Order Sensitivities,” *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 2, 2022, pp. 196–212, 10.2514/1.G006089.
- [9] T. N. Edelbaum, “Propulsion Requirements for Controllable Satellites,” *ARS Journal*, Vol. 31, No. 8, 1961, pp. 1079–1089, 10.2514/8.5723.
- [10] B. J. Wall and B. A. Conway, “Shape-Based Approach to Low-Thrust Rendezvous Trajectory Design,” *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 95–101, 10.2514/1.36848.
- [11] D. Wu, T. Zhang, Y. Zhong, F. Jiang, and J. Li, “Analytical Shaping Method for Low-Thrust Rendezvous Trajectory Using Cubic Spline Functions,” *Acta Astronautica*, Vol. 193, Apr. 2022, pp. 511–520, 10.1016/j.actaastro.2022.01.019.
- [12] D. Hennes, D. Izzo, and D. Landau, “Fast Approximators for Optimal Low-Thrust Hops between Main Belt Asteroids,” *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec. 2016, pp. 1–7, 10.1109/SSCI.2016.7850107.
- [13] L. Casalino, “Approximate Optimization of Low-Thrust Transfers between Low-Eccentricity Close Orbits,” *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 1003–1008, 10.2514/1.62046.
- [14] H. Shen and L. Casalino, “Simple ΔV Approximation for Optimization of Debris-to-Debris Transfers,” *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 575–580, 10.2514/1.A34831.
- [15] P. Gurfil, “Analysis of J2-perturbed Motion Using Mean Non-Osculating Orbital Elements,” *Celestial Mechanics and Dynamical Astronomy*, Vol. 90, No. 3, 2004, pp. 289–306, 10.1007/s10569-004-0890-x.
- [16] A. Petropoulos, D. Grebow, D. Jones, G. Lantoine, A. Nicholas, J. Roa, J. Senent, J. Stuart, N. Arora, T. Pavlak, T. Lam, T. McElrath, R. Roncoli, D. Garza, N. Bradley, D. Landau, Z. Tarzi, F. Laipert, E. Bonfiglio, M. Wallace, and J. Sims, “GTOC9: Results from the Jet Propulsion Laboratory (Team JPL),” *Acta Futura*, Vol. 11, Jan. 2018, pp. 25–35, 10.5281/zenodo.1139152.
- [17] Z. Zhang, N. Zhang, X. Guo, D. Wu, X. Xie, J. Li, J. Yang, S. Chen, F. Jiang, H. Baoyin, H. Li, H. Zheng, and X. Duan, “GTOC 11: Results from Tsinghua University and Shanghai Institute of Satellite Engineering,” *Acta Astronautica*, Vol. 202, Jan. 2023, pp. 819–828, 10.1016/j.actaastro.2022.06.028.
- [18] Y. Zhang, C. Wen, and D. Qiao, “Efficient Low-Thrust Trajectory Data Generation Method Based on the Switch Function,” *Journal of Guidance, Control, and Dynamics*, Vol. 48, Jan. 2025, pp. 156–167, 10.2514/1.G008305.
- [19] J. Park and I. W. Sandberg, “Universal Approximation Using Radial-Basis-Function Networks,” *Neural Computation*, Vol. 3, June 1991, pp. 246–257, 10.1162/neco.1991.3.2.246.
- [20] Y.-h. Zhu and Y.-Z. Luo, “Fast Evaluation of Low-Thrust Transfers via Multilayer Perceptions,” *Journal of Guidance, Control, and Dynamics*, Vol. 42, Dec. 2019, pp. 2627–2637, 10.2514/1.G004080.
- [21] H. Li, S. Chen, D. Izzo, and H. Baoyin, “Deep Networks as Approximators of Optimal Low-Thrust and Multi-Impulse Cost in Multitarget Missions,” *Acta Astronautica*, Vol. 166, Jan. 2020, pp. 469–481, 10.1016/j.actaastro.2019.09.023.
- [22] X. Guo, D. Ren, D. Wu, and F. Jiang, “DNN Estimation of Low-Thrust Transfer Time: Focusing on Fast Transfers in Multi-Asteroid Rendezvous Missions,” *Acta Astronautica*, Vol. 204, Mar. 2023, pp. 518–530, 10.1016/j.actaastro.2022.09.006.
- [23] G. Acciarini, L. Beauregard, and D. Izzo, “Computing Low-Thrust Transfers in the Asteroid Belt, a Comparison between Astrodynamical Manipulations and a Machine Learning Approach,” May 2024, 10.48550/arXiv.2405.18918.
- [24] N. Harl, K. Rajagopal, and S. N. Balakrishnan, “Neural Network Based Modified State Observer for Orbit Uncertainty Estimation,” *Journal of Guidance, Control, and Dynamics*, Vol. 36, July 2013, pp. 1194–1209, 10.2514/1.55711.
- [25] C. Wilson and M. Vasile, “Generation and Classification of Critical Points in Uncertain N-body Problems via Machine Learning,” *75th International Astronautical Congress, ITA*, Oct. 2024.

- [26] L. Federici, A. Scorsoglio, L. Ghilardi, A. D'Ambrosio, B. Benedikter, A. Zavoli, and R. Furfaro, "Image-Based Meta-Reinforcement Learning for Autonomous Guidance of an Asteroid Impactor," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 11, 2022, pp. 2013–2028, 10.2514/1.G006832.
- [27] D. Izzo and E. Öztürk, "Real-Time Guidance for Low-Thrust Transfers Using Deep Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 2, 2021, pp. 315–327, 10.2514/1.G005254.
- [28] D. Izzo and S. Origer, "Neural Representation of a Time Optimal, Constant Acceleration Rendezvous," *Acta Astronautica*, Vol. 204, Mar. 2023, pp. 510–517, 10.1016/j.actaastro.2022.08.045.
- [29] M. Pugliatti, A. Scorsoglio, R. Furfaro, and F. Toppo, "Onboard State Estimation Around Didymos With Recurrent Neural Networks and Segmentation Maps," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 60, June 2024, pp. 2542–2554, 10.1109/TAES.2023.3288506.
- [30] M. J. H. Walker, B. Ireland, and J. Owens, "A Set Modified Equinoctial Orbit Elements," *Celestial mechanics*, Vol. 36, Aug. 1985, pp. 409–419, 10.1007/BF01227493.
- [31] J. L. Junkins and E. Taheri, "Exploration of Alternative State Vector Choices for Low-Thrust Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 42, Jan. 2019, pp. 47–64, 10.2514/1.G003686.
- [32] Y. Gao and C. Kluever, "Low-Thrust Interplanetary Orbit Transfers Using Hybrid Trajectory Optimization Method with Multiple Shooting," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2004, 10.2514/6.2004-5088.
- [33] R. Bertrand and R. Epenoy, "New Smoothing Techniques for Solving Bang–Bang Optimal Control Problems—Numerical Results and Statistical Interpretation," *Optimal Control Applications and Methods*, Vol. 23, No. 4, 2002, pp. 171–197, 10.1002/oca.709.
- [34] F. Jiang, H. Baoyin, and J. Li, "Practical Techniques for Low-Thrust Trajectory Optimization with Homotopic Approach," *Journal of Guidance, Control, and Dynamics*, Vol. 35, Jan. 2012, pp. 245–258, 10.2514/1.52476.
- [35] J. J. Moré, B. S. Garbow, and K. E. Hillstom, "User Guide for MINPACK-1," tech. rep., CM-P00068642, 1980.
- [36] W. S. Levine, *The Control Systems Handbook: Control System Advanced Methods, Second Edition*. CRC Press, Oct. 2018.
- [37] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling Laws for Neural Language Models," Jan. 2020, 10.48550/arXiv.2001.08361.
- [38] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *International Conference on Learning Representations*, Sept. 2018.
- [39] R. Llugsi, S. E. Yacoubi, A. Fontaine, and P. Lupera, "Comparison between Adam, AdaMax and Adam W Optimizers to Implement a Weather Forecast Based on Neural Networks for the Andean City of Quito," *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, Oct. 2021, pp. 1–6, 10.1109/ETCM53643.2021.9590681.
- [40] L. N. Smith and N. Topin, "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates," *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Vol. 11006, SPIE, May 2019, pp. 369–386, 10.1117/12.2520589.
- [41] I. S. Grigoriev and M. P. Zapletin, "Choosing Promising Sequences of Asteroids," *Automation and Remote Control*, Vol. 74, No. 8, 2013, pp. 1284–1296, 10.1134/S0005117913080055.