76th International Astronautical Congress (IAC 2025), Sydney, Australia, 29 Sep-3 Oct 2025.

IAC-25,C1,5,6,x97852

Reference trajectory-aided reinforcement learning for autonomous exterior cislunar transfers

Claudio Toquinho Campana[a*], **Francesco Topputo**[b]

[a,b] *Dept. of Aerospace Science and Technology, Politecnico di Milano, Via G. La Masa 34, 20156 Milano, Italy*
[*] Corresponding Author

Abstract

With a new era of Moon exploration approaching, autonomy is crucial for sustaining a reliable highway of spacecraft flying the cislunar environment. The increasing use of low-cost small platforms necessitates leveraging natural dynamical structures, such as those revealed by the Earth–Moon, Sun-perturbed bi-circular restricted four-body problem, to achieve efficient transfers. However, intricate gravitational dynamics and strict mission constraints make autonomous trajectory planning and closed-loop guidance highly challenging. This work presents a reinforcement learning framework to train an agent that corrects trans-lunar injection maneuver errors at the Earth, thereby enabling autonomous transfers toward the L_2 point region of the Earth–Moon system. The algorithm integrates expert trajectories to guide the learning process for exterior cislunar transfers that exploit the solar attraction. Exploration is enhanced by injecting reference training episodes, thus steering the agent toward desirable motion paths. Simulations demonstrate that the proposed method helps automatizing spacecraft guidance from the Earth to the lunar region, meeting operational constraints, propulsive limits, and successfully compensating thrust errors. Overall, reference trajectory-aided reinforcement learning improves learning efficiency and reliability in complex multi-body dynamics, advancing fully autonomous deep-space mission planning.
Keywords: Cislunar transfers, Reference trajectory-aided reinforcement learning, PPO, Autonomous guidance.

1 Introduction

Humans' innate desire to expand their knowledge and address unresolved scientific questions is propelling the next era of space exploration. As a necessary first step, the colonization of the lunar environment is pivotal to enabling the next generation of human missions to remote regions of the solar system. Lunar settlements and orbiting stations will serve as vital staging point for spacecraft venturing into deep space [1]. To lay the groundwork for this effort, numerous missions have been recently launched in the cislunar space, such as, Chang'e-5 [2], CAPSTONE [3], Danuri [4], Artemis I, Chandrayaan-3 [5], SLIM [6], Chang'e-6, with many more planned for the near future.

As the number of satellites and spacecraft transiting cislunar space inevitably increases, engineers must develop solutions to the resulting challenges to ensure the sustainability of the forthcoming space economy. To mention some, small-sized platforms have recently been introduced as cost-effective solutions that balance risk and performance [7]. Low-energy trajectories, that exploit the intrinsic chaoticity of some dynamic regimes, are more frequently adopted to enable safe and fuel-efficient transfers [8]. Finally, current mission concepts heavily relies on the human-in-the-loop supervision after launch. Autonomous probes are opening the future to self-reliant operations beyond Earth orbit, reducing the need for continuous ground control and enabling more responsive mission architectures [9].

Designing cislunar transfers is a delicate process. The environment is inherently chaotic, resulting in a complex phase space that prevents dynamic generalization. However, specialized astrodynamic models can be utilized in the early stages of mission design [10]. Simplified dynamic models reveal natural structures, such as periodic orbits and invariant manifolds, that provide insight into specific regions of the dynamic landscape [11]. Preliminary trajectories, generated by optimization algorithms, are later refined into more realistic and complex models [12, 13]. Then, after launch, the spacecraft corrects its motion to ensure that the desired path is accurately followed. At this stage, several algorithms and methods available in the literature allow for iterative trajectory updates, ensuring effective guidance and driving the spacecraft toward its target destination [14].

In recent years, the advent of high-performance computers brought to the adoption of machine learning, or, more in general, artificial intelligence techniques to solve astrodynamics problems and facilitate space mission design practices [15]. Campana et al. [16] applied clustering techniques to extract recurrent families of Earth–

[a] Ph.D. Student, claudiotoquinho.campana@polimi.it
[b] Full Professor, francesco.topputo@polimi.it

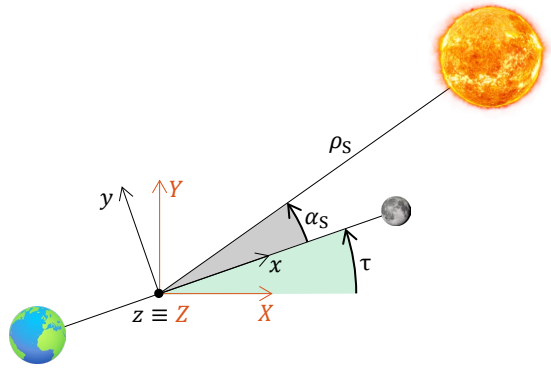

Fig. 1: BCR4BP model (not to scale). Non-rotating, quasi-inertial coordinates in orange.

Moon (EM) transfers. Smith et al. [17] extracted motion primitives to summarize dynamical structures in the circular restricted three-body problem (CR3BP). A survey on the application of supervised learning and reinforcement learning (RL) in the space sector is provided by Shirobokov et al. [18]. Guidance and control problems are addressed in [19–22], where RL is adopted to design robust and efficient trajectories.

This research builds on the concepts introduced above. After a brief background on relevant notions in Sec. 2, a reference cislunar trajectory is designed in the EM, Sun-perturbed bi-circular restricted four body problem (BCR4BP), a dynamically rich astrodynamics model that includes the solar perturbation. A transfer to the L_2 point of the EM system is formulated, given the strategic value this location holds in space applications. Next, in Sec. 3, a RL agent is trained to correct potential injection errors at the Earth, instantaneously and in an autonomous fashion, ensuring the spacecraft reaches its target despite initial deviations. During training, the reference trajectory is provided to the agent to facilitate learning. The advantages of adopting a trajectory-aided RL approach are highlighted, and results are presented and discussed in Sec. 4. Finally, Sec. 5 summarizes the outcomes of the study and outlines future work.

2 Background

2.1 Earth–Moon, Sun-perturbed bi-circular restricted four-body problem

In the BCR4BP, the Earth and the Moon revolve circularly around their common barycenter which, in turn, orbits the Sun as depicted in Fig. 1. It is convenient to formulate the equations of motion in a rotating frame (rf) centered at the Earth–Moon barycenter, with the x axis directed toward the Moon. In the spatial problem,

Tab. 1: Non-dimensional BCR4BP physical parameters.

Sym.	Value	Description
μ	$1.215\,066\,830\times10^{-2}$	mass parameter
m_S	$3.289\,005\,410\times10^{5}$	Sun mass
ρ_S	$3.888\,111\,430\times10^{2}$	EM–S distance
ω_S	$-9.251\,959\,850\times10^{-1}$	Sun angular velocity

the non-dimensional dynamic is expressed as [23]

$$\begin{aligned}\ddot{x} &= 2\dot{y} + \frac{\partial \Omega_4}{\partial x}\\ \ddot{y} &= -2\dot{x} + \frac{\partial \Omega_4}{\partial y}\\ \ddot{z} &= \frac{\partial \Omega_4}{\partial z},\end{aligned} \tag{1}$$

where

$$\begin{aligned}\Omega_4 \;=\; &\frac{1}{2}(x^2+y^2) + \frac{1-\mu}{r_E} + \frac{\mu}{r_M} + \frac{1}{2}\mu(1-\mu) + \\ &+ \frac{m_S}{r_S} - \frac{m_S}{\rho_S^2}\big(x\cos(\alpha_S) + y\sin(\alpha_S)\big)\end{aligned} \tag{2}$$

is the effective potential and

$$\alpha_S = \omega_S(\tau - \tau_0) + \alpha_S\big|_{\tau_0} \tag{3}$$

is the instantaneous Sun phase angle in the rotating frame. The quantity r_i represents the scaled distance of the spacecraft from the ith celestial body. In this representation, the Earth is fixed at $[-\mu, 0, 0]$, whereas the Moon rests at $[1-\mu, 0, 0]$. Table 1 reports the normalized values characterizing the model.

2.2 Reinforcement learning via Proximal Policy Optimization

RL is a goal-oriented approach to learn policies by interaction with an environment and trial-and-error search [24]. An agent, that is, the learning actor, interacts with the environment to understand how to behave in order to maximize the long-term reward. Specifically, at a training step t, the agent perceives, possibly partially, the current state of the environment through the state vector $\boldsymbol{s}_t$, takes an action $\boldsymbol{a}_t$ drawn from a policy function $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, and receives a reward signal $r_t(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{s}_{t+1})$ as the result of the forward propagation of the environment from t to $t+1$. A sequence of steps, i.e., state-action pairs, define a training episode that lasts until reaching either a terminal state or a maximum number of steps. Directly connected to the concept of reward, the state-value function of a state under a policy π is the total amount of reward an agent can expect to accumulate over the future starting from the state itself. It is formally defined as [24]

$$V^{\pi}(\boldsymbol{s}_t) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\right], \tag{4}$$

where the discount factor $\gamma = [0, 1)$ regulates the importance of future rewards as compared to immediate ones. A policy whose value function is optimal, that means, each state is assigned the largest expected return, is an optimal policy [24].

Problems with very large state spaces that cannot be efficiently represented by tabular descriptions require some form of generalization. In these cases, optimal policies cannot be found, and the goal shifts to identifying good approximate solutions that maximize value functions. RL algorithms often leverage neural networks (NN), which serve as universal function approximators [25] to achieve this level of generalization and to represent the learned policies. In its basic feedforward form, a NN is a sequence of layers, each consisting of multiple nodes, called neurons or perceptrons. Information flows from the input layer to the output layer, potentially passing though one or more hidden layers to enhance complexity. Along this computational path, a neuron k processes the signals x_j coming from the previous layer of N neurons as

$$y_k = \varphi\left(\sum_{j=0}^{N} w_{kj} x_j + b_k\right), \tag{5}$$

where φ is a nonlinear activation function, and w_{kj} and b_k are weights and bias, respectively, to be learned during training. These parameters are adjusted at each iteration using a stochastic gradient descent optimization algorithm based on the back-propagation of a loss function modeling the training error [26].

RL algorithms designed to solve training problems can be broadly categorized into two principal families: action-value and policy gradient methods [24]. In the former, the value of taking a specific action in a given state state is learned, enabling selection of the most promising action to maximize the cumulative return. Contrarily, policy gradient methods learn a parametrized policy directly, allowing actions to be selected without referencing a value function. Several algorithms exist for both categories. To solve the problem, this work adopts the Proximal Policy Optimization (PPO) algorithm, a state-of-the-art policy gradient method known for its effectiveness in continuous control tasks and robustness in dynamically complex environments [24, 27]. Specifically, PPO is an actor-critic method that simultaneously learns NN approximations of both the policy (actor) and the value function (critic). The actor $\pi_{\boldsymbol{\nu}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ is parametrized by trainable weights and biases $\boldsymbol{\nu}$, and the critic $V^{\pi}_{\boldsymbol{\xi}}(\boldsymbol{s}_t)$ by $\boldsymbol{\xi}$. Ultimately, the learning process terminates when the parameters $\boldsymbol{\theta} = [\boldsymbol{\nu}, \boldsymbol{\xi}]$ have been estimated. At each learning iteration, a sequence of state-action-reward tuples is collected from the environment using the current policy. Then, the Generalized Advantage Function (GAE) at an episode step t is estimated as [28]

$$\hat{A}_t(\boldsymbol{s}_t, \boldsymbol{a}_t) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \tag{6}$$

with the temporal difference defined as

$$\begin{aligned}\delta_{t+l} = r_{t+l}(\boldsymbol{s}_{t+l}, \boldsymbol{a}_{t+l}; \boldsymbol{s}_{t+l+1}) + \gamma V^{\pi}(\boldsymbol{s}_{t+l+1}) + \\ - V^{\pi}(\boldsymbol{s}_{t+l}).\end{aligned} \tag{7}$$

The GAE quantifies whether the taken action outperformed or underperformed the expected behavior of the current policy, with the parameter λ trading off bias and variance. Defining the probability ratio between the new and old policies as

$$z_t(\boldsymbol{\nu}) = \frac{\pi_{\boldsymbol{\nu}}\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)}{\pi_{\boldsymbol{\nu}_{\text{old}}}\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)}, \tag{8}$$

the clipped objective function is expressed as [27]

$$L^{\text{CLIP}}(\nu) = \hat{\mathbb{E}}_t\left[\min\left(z_t \hat{A}_t, \text{clip}\left(z_t, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], \tag{9}$$

where the clipping parameter ϵ limits how much the new policy can diverge from the old one in a single update. Finally, the total PPO loss function is modeled as [27]

$$L^{\text{PPO}}_t(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t\left[L^{\text{CLIP}}_t(\boldsymbol{\nu}) - c_1 L^{\text{VF}}_t(\boldsymbol{\xi}) + c_2 S[\pi_\theta](s_t)\right], \tag{10}$$

where $L^{\text{VF}}_t(\boldsymbol{\xi}) = (V^{\pi}_{\boldsymbol{\xi}}(\boldsymbol{s}_t) - V^{\text{targ}}_t)^2$ is a squared-error loss that encourages learning of the true value function, S denotes an entropy bonus to promote exploration, and c_i are coefficients tuning the relative importance of each term. At each learning iteration, the experience collected from the environment is used to update the NNs parameters using a stochastic gradient descent algorithm. This work employs the Adam optimizer [29].

3 Reference trajectory-aided reinforcement learning

This section outlines the process for automatizing spacecraft guidance using RL. First, the procedure to generate a reference low-energy trajectory to reach the vicinity of the Moon is presented. Then, the trajectory-aided RL approach is introduced.

3.1 Reference trajectory design

3.1.1 Seeding trajectory generation

To simulate a realistic mission, the spacecraft is assumed to target a Halo orbit about the L_2 point of the EM system [30]. As an example, a target orbit with

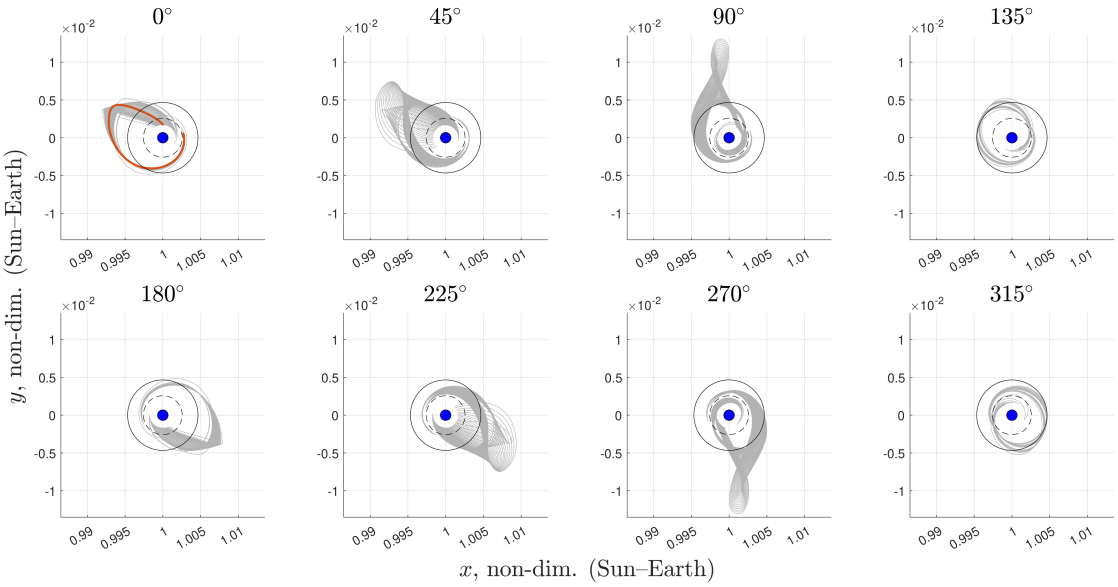

Fig. 2: Seeding trajectories projected onto the Sun–Earth rotating frame. Tiles correspond to different Sun–Earth–Moon configurations at the crossing of the EM region of prevalence (black circle). Dashed line marks lunar orbit. Orange solution is reference for trajectory-aided RL processing.

Jacobi energy of 3.09 is chosen. Initial conditions for this orbit in the CR3BP are estimated via a third-order Lindstedt–Poincaré approximation. Convergence to the desired Halo orbit is then achieved through a continuation scheme based on single shooting.

Dynamical systems theory suggests that stable and unstable invariant manifolds arise from periodic orbits around Lagrange points [31]. These manifolds are generated by applying small perturbations along the orbits themselves. The spacecraft shall approach the Halo orbit from afar of the EM system. Therefore, the right stable manifold is computed by perturbing the orbital states along the local stable directions and back-propagating until crossing the EM region of prevalence. This imaginary surface bounds the region where the dynamics of the BCR4BP can be effectively approximated by the CR3BP solely, thereby ignoring the solar effect.

Terminal states of each manifold trajectory are further propagated backward in the BCR4BP. Since the model is non-autonomous, the Sun–Earth–Moon relative configuration along the trajectory is relevant. Eight different geometries are considered at the time the spacecraft departs backward the section by varying the solar angle α_S from 0° to 360°, with an angular discretization of 45°. Propagation terminates either when the spacecraft reaches the vicinity of the Earth at a threshold distance from its center of 2/3 non-dimensional units or after a maximum propagation time of 100 days. Figure 2 reports all trajectories obtained with the mentioned procedure. Solutions are projected onto the xy plane of the Sun–Earth rotating frame. The trajectory highlighted in orange, selected randomly from the first family, serve as seed to generate the reference for the trajectory-aided RL training phase.

3.1.2 Nonlinear programming optimization

In [30], hundreds of transfers to the target Halo were generated by optimizing the seeding trajectories shown in Fig. 2. Given the highly sensitive nature of the dynamics, automating guidance over a full trajectory via RL would be extremely challenging. For this reason, focus is placed on the cislunar transfer phase only, disregarding the terminal orbital insertion phase. To determine the target state, all trajectories from [30] are sectioned near their end at the xz plane, positions and velocities are recorded and then averaged out. This ensures a spacecraft reaches the vicinity of the destination Halo orbit, so that other techniques can handle the final insertion phase. The resulting target state in the EM system is $\boldsymbol{x}^* = [1.2809, 0.0, -0.1231, -0.2016, -0.3163, 0.1218]$.

A nonlinear programming problem is formulated to optimize the trajectory highlighted in Fig. 2. Objective functions and constraints are introduced to simulate a real space mission, thereby incorporating technological limits and operational constraints. The optimization is carried out via direct transcription and multiple shooting [14]. After trans-lunar injection (TLI), the spacecraft is assumed to perform $n_m = 6$ trajectory correction maneuvers (TCMs) to reach the target state. To transcript the problem, the trajectory is sectioned into

segments at each TCM execution point. Therefore, including the departure point, a total of 7 nodes model the optimization problem. Each maneuver is scheduled 14 days after the previous one to accommodate navigation and operation routines. The objective is to minimize the fuel expenditure for the transfer, and is modeled as

$$\mathcal{J} = \sum_{i=2}^{n_m+1} \left\| \boldsymbol{\Delta V}_i \right\|. \tag{11}$$

As the TLI maneuver is assumed to be provided by the last stage of the launcher, its cost is not included in the loss function. Nonlinear equality and inequality constraints define mission requirements. Prior to TLI maneuver, the spacecraft orbits a 167 km altitude circular low Earth orbit (LEO). To prescribe a tangential departure, equality constraints

$$\begin{aligned} &(x_1+\mu)^2 + (y_1)^2 + (z_1)^2 - r_{\text{dep}}^2 = 0 \\ &(x_1+\mu)\,\dot{x}_1 + y_1\,\dot{y}_1 + z_1\,\dot{z}_1 = 0 \end{aligned} \tag{12}$$

are introduced, with r_{dep} representing the departing non-dimensional distance from the center of the Earth. Arrival at the target point is guaranteed by imposing $\boldsymbol{x}_7 = \boldsymbol{x}^*$. Ballistic continuity between adjacent nodes is enforced on positional components by

$$\boldsymbol{\xi}_{i_{pos}} = \boldsymbol{\varphi}(\boldsymbol{x}_i, \tau_i; \tau_{i+1})_{\text{pos}} - \boldsymbol{x}_{i+1_{\text{pos}}} = \boldsymbol{0}, \tag{13}$$

where $\boldsymbol{\varphi}(\boldsymbol{x}_i, \tau_i; \tau_{i+1})_{\text{pos}}$ denotes the flow of the ith segment of the trajectory propagated in the BCR4BP dynamics. As TCMs are permitted, velocity within two arcs may change instantaneously. Inequality constraints bound each TCM magnitude to 50 m/s, with a maximum total ΔV of 200 m/s. Additional constraints prevent impacting with either primary along the transfer. After optimization, the resulting state–action sequence becomes available for the subsequent trajectory-aided RL training phase.

3.2 Reinforcement learning framework

In recent years, numerous studies have proposed methodologies to facilitate optimization of action policies via behavior cloning algorithms. Learning from demonstrations, or imitation learning, supplies expert examples to learn suitable action policies that replicate the demonstrated behavior. RL is subsequently employed to generalize over unseen, but related, conditions and scenarios [32, 33]. Expert state-action pairs may be used for pre-training of NNs, as in supervised learning. A behavior cloning loss term is then incorporated into Eq. 10 to reward similarity between agent and expert actions [34–38]. After an introduction on the RL environment modeling the problem of the study, the approach to inject expert actions into the learning process, when necessary, is described.

3.2.1 Environment modeling

In RL, an agent interacts with an environment that represents the physical scenario. At each state, an action is drawn from a policy function, the environment is updated, and a reward is collected. In this study, the state is defined as a 6×1 vector of spacecraft position and velocity components. The action is a 4×1 vector: the first three elements specify the thrust direction, and the last defines the magnitude of the impulse. Each maneuver can generate an instantaneous velocity change up to 70 m/s, to introduce some additional controllability with respect to the reference trajectory. State and action components are linearly scaled to $[-1, 1]$, with bounds corresponding to physical limits large enough to properly model the problem. The environment is implemented using the Gymnasium library in Python. Both the *reset* and *step* episode phases must be defined for a complete description of the environment.

The *reset* of the environment is invoked at the beginning of a new training episode. The state on the reference trajectory at LEO prior to TLI is extracted. Then, a perturbed TLI maneuver is performed and the resulting state is propagated for 14 days in the BCR4BP. If, instead, an expert demonstration is requested by the learning routine, the reference TLI impulse is applied to the initial state, and the dynamics is propagated without further manipulation. Position and velocity at the end of the propagation define the state for the next phase.

In the *step* phase, the agent draws an action by querying the actor network over the state information. After rescaling to physical quantities, the maneuver is applied, and the resulting state is propagates for 14 days. In case the spacecraft makes a fly-by excessively close to either primary or crosses prescribed distance boundaries, the propagation is halted and the episode terminates. This prevents inclusion of excessively sensitive or overly broad regions of the phase space that would hinder learning. If no termination occurs, the step reward is obtained by summing individual contributions

$$r_t = r_{\Delta V} + r_{\text{targ}} + r_{\text{dem}}, \tag{14}$$

with

- $r_{\Delta V} = -0.25\ \Delta V / \Delta V_{\max}$: fuel penalty;
- $r_{\text{targ}} = (1/3\, e^{-10\,\delta_{\text{pos}}^{\text{targ}}+3} - 2) + (1/3\, e^{-10\,\delta_{\text{vel}}^{\text{targ}}+3} - 2)$: reward closeness to the target, with δ denoting Euclidean distance;

- $r_{\text{dem}} = (1/8\, e^{-8\,\delta_{\text{pos}}^{\text{dem}}+3} - 1) + (1/8\, e^{-8\,\delta_{\text{vel}}^{\text{dem}}+3} - 1)$: reward proximity to the reference trajectory.

An episode is truncated if its length exceeds the number of steps required to fly the reference trajectory. Furthermore, an episode may terminate with:

- Success: position and velocity fall within threshold Euclidean distances from the target state, yielding $r_{\text{success}} = 10$;
- Fly-by: excessively close fly-by to either primary triggers termination with penalty $r_{\text{fly-by}} = -10\,p$, where p decreases with the number of successful steps completed;
- Boundary crossing: the spacecraft state exceeds environment limits, incurring in $r_{\text{bound}} = -10\ -10\,p$.

Rewards and penalties are designed to encode mission objective and constraints, guiding the algorithm toward a suitable policy. A local exponential moving average is used to normalize rewards, thus stabilizing the training.

3.2.2 Reference trajectory injection

This work adopts imitation learning to facilitate the training of an agent. Although conceptually similar to the studies presented at the beginning of the section, the method introduced here differs substantially. The primary objective is autonomous guidance to compensate for injection errors at the Earth, thereby ensuring minimum deviation from the planned path. Since a single reference trajectory is considered, only few expert state-action pairs exist, preventing pre-training of networks in a supervised manner. The implemented approach is based on the injection of a reference trajectory every fixed number of training episodes. At those instances, the agent interacts with an expert teacher by querying the appropriate action for the observed state: the teacher takes control over the agent, applies reference state-action pairs until episode completion, and then releases control back to the agent to continue sampling from the environment. This on-demand knowledge injection fosters learning in chaotic and sensitive environments, effectively restricting exploration to regions of the search space relevant to accomplish the task.

The open-source Python library RLlib from Ray enables experimenting with RL algorithms for both academic and industrial applications. Its architecture supports experimenting, rapid prototyping, and customization. For these reasons, the PPO algorithm implemented in RLlib is employed in this research for agent training.

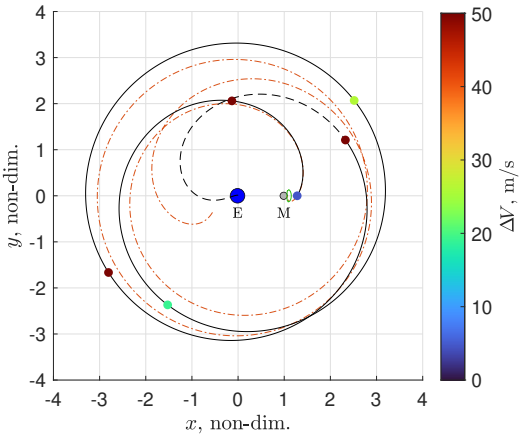

Fig. 3: Seeding trajectory in orange, reference transfer in black, target Halo in green. Colorful dots are TCMs.

Whenever an expert demonstration is initiated at *reset* time, the teacher assumes control: actions are extracted from the expert dataset rather than being drawn from statistical properties generated by the actor network. As expected, such expert-driven episodes conclude successfully. During training, maintaining a balance between expert demonstrations and random episodes is critical. Excessive reliance on reference trajectories risks bias and overfitting, hindering generalization to unseen initial conditions. Conversely, without guidance, the chaotic and highly sensitive dynamics would render learning impractical, since exploration would cover vast, unproductive regions without assistance.

4 Agent training and results

The method outlined in Sec. 3.1.2 is now applied to design a baseline transfer to the L_2 point of the EM system. Subsequently, the reference trajectory-aided RL technique developed Sec. 3.2 is adopted to train the control agent. A robustness analysis concludes the section.

4.1 Demonstration trajectory generation

Trajectory highlighted in Fig. 2 is optimized through direct transcription and multiple shooting. Figure 3 shows in black the resulting demonstration trajectory that is used to guide the training process. After TLI, the spacecraft follows a first ballistic, uncontrolled arc (dashed curve). It then executes a sequence of impulsive maneuvers until reaching the target state in proximity of the Halo orbit. In total, the spacecraft requires a cumulative ΔV of 200 m/s.

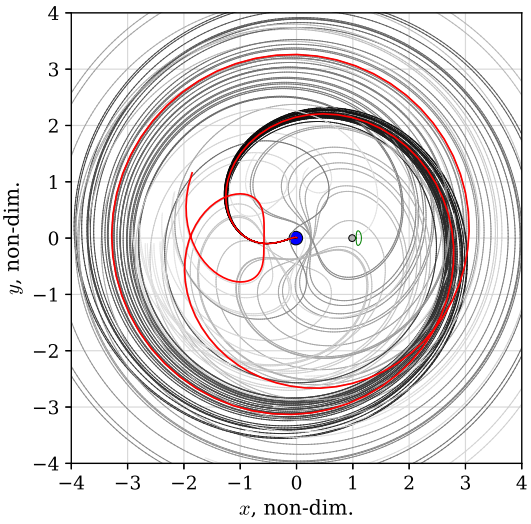

Fig. 4: Propagation of trajectories. Trajectories with perturbed TLI are shown from black to gray as propagation time progresses. Uncontrolled reference trajectory with nominal TLI in red.

Tab. 2: PPO hyperparameters and NNs structure.

Parameter	**Value**
N° training iterations	1e4
Batch size	4096
Minibatch size	64
N° epochs per batch	3
Initial learning rate	3e-5
Final learning rate	3e-6
Discount factor (γ)	0.9
GAE factor (λ)	0.8
Value function loss coeff. (c_1)	1
Initial entropy coeff. ($c_{2,i}$)	1e-3
Final entropy coeff. ($c_{2,f}$)	1e-4
Clip parameter (ϵ)	1e-1
N° actor NN hidden layers	2
N° actor NN neurons per layer	128
Actor hidden layers activation function	ReLU
Actor output layer activation function	tanh
N° critic NN hidden layers	2
N° critic NN neurons per layer	256
Critic hidden layers activation function	ReLU
Critic output layer activation function	linear

4.2 Training and testing

4.2.1 Trans-lunar injection error

TLI errors are injected to mimic a realistic space mission. In particular, three sources of error are introduced when initializing each training episode.

Execution time error: since the model is non-autonomous, the departure epoch is relevant. The reference departure time, defining the Sun–Earth–Moon relative geometry, can vary by up to 15 minutes (3σ);

Direction error: TLI maneuvers are subject to pointing uncertainty. A three-dimensional cone of semi-aperture 0.1° (3σ) around the nominal thrust direction is defined to model TLI alignment errors. Passage through the quasi-inertial frame centered at the Earth is necessary to introduce the error.

Magnitude error: TLI burns also exhibit magnitude uncertainty. The reference TLI ΔV is perturbed with a standard deviation of 10 m/s (3σ). Again, passage through the quasi-inertial frame centered at the Earth is necessary to introduce the error.

Figure 4 reproduces several examples of ballistic trajectories obtained by perturbing the reference TLI maneuver and propagating for the nominal transfer duration. As the figure illustrates, sensitive dependence on initial conditions causes the perturbed trajectories to diverge rapidly, retaining similar geometries only during the early phase (dark sections). The red trajectory, representing the ballistic propagation of the reference path after nominal TLI, confirms that active control is required to ensure the spacecraft reaches the target region.

4.2.2 Training

Once the demonstration trajectory, that is, the expert state-action pairs, is available, the reference trajectory-aided RL training begins. A successful training hinges on several factors: the design of a reward function that steers the learning process toward the most relevant regions of the search space; the careful tuning of PPO hyperparameters; and a well-structured NN architecture. In practice, an iterative approach is required to identify and adjust all variables affecting training. To assist with hyperparameter selection, one can employ libraries that use autonomous search strategies, such as Bayesian optimization, to propose candidate values. After exhaustive tuning guided by the outcomes of multiple training campaigns, Table 2 presents the configuration yielding to the best outcome. Difficulties emerged during training owing to very short episode length (the maximum number of steps before termination is $n_m - 1$, since the final maneuver necessary to achieve the terminal state does not generate a training step) and extreme sensitivity of the underlying dynamics. As a result, careful tuning of hyperparameters is required to mitigate high variance and poor generalization. Moreover, as noted in Sec. 3.2.2, a balance between random episodes and episodes fed with demonstrations must be attained. In this scenario, it results that injecting the reference trajectory once every 50 episodes significantly improves training quality. In contrast, the chaoticity of the dynamics and the large state-action space preclude learning if excluding demonstrations entirely, as observed in other training campaigns.

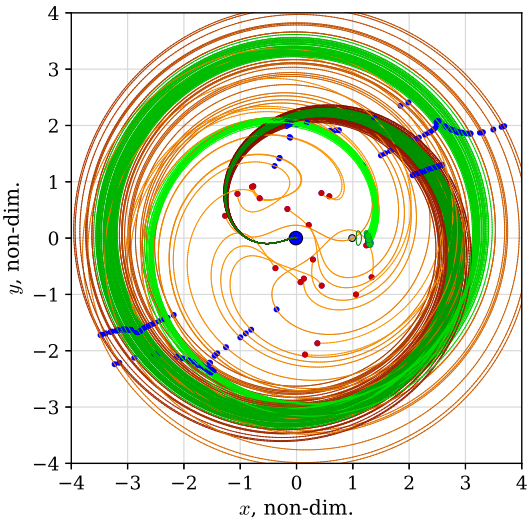

Fig. 5: Examples of controlled motion after policy training. Successful and faulted trajectories in green and red (darker to lighter), respectively. Intermediate TCMs in blue (last maneuvers colored following episode outcome).

4.3 Testing

After training, the autonomous control agent is tested over 1000 random initializations of the environment. In 83% of the cases, episodes terminate at the designated target region. None of the trajectories crosses environment boundaries, which would correspond to leaving the EM system. 12% of the training episodes end because the spacecraft approaches too close to either the Earth or the Moon, and the remaining 5% reach the maximum number of steps.

Figure 5 reproduces 100 trials using the trained agent. Most trajectories reach the target state close the L_2 point of the EM system within the prescribed tolerances. Others fail to control and wander through the cislunar space until episodes terminate either due to close passages to a primary, or by reaching the maximum number of steps. Examining the early segments of the trajectories in Fig. 5, the TLI direction accuracy results to strongly affect controllability: a wide pointing cone error can place the spacecraft in conditions from which the policy cannot recover, whether because of control authority limits or suboptimal hyperparameter tuning.

Figure 6 shows the distribution of TCMs costs across 100 successful random trials. The first two maneuvers and the fifth one incur the highest ΔV, in agreement with the reference trajectory in Fig. 3. This evidences the influence the injection of a demonstration has in steering the training process and in impacting its final quality.

4.3.1 Robustness of guidance policy

To test the robustness of the guidance policy, control errors are injected into the environment on top of the actions determined by the agent. Five test campaigns, each comprising 1000 episodes, are performed, perturbing a different TCM in each case while leaving the others unaltered. Perturbed actions are subject to magnitude uncertainty with a dispersion of 10% (3σ) relative to their policy-generated values. Results show that performance remains virtually unchanged when errors are injected at any TCM except the fifth. In that case, the success rate slightly decreases since errors in the final impulse cannot be compensated by subsequent maneuvers, as they can instead for the other burns.

5 Conclusions

The lunar environment is foreseen to play a strategic role in the next generation of space exploration. Engineers and scientists are developing technologies to reduce overall mission design costs, making access to space more affordable and efficient. Although innovative production processes, standardized components, reduced scale spacecraft, and efficient trajectories have yielded significant gains, post-launch operational expenses remain largely unchanged. Consequently, autonomous systems are essential to alleviate ground-control workloads, increase launch cadence, and maximize both scientific and economic returns.

This study presents a method to correct potential injection errors at the Earth for transfers heading to the vicinity of the Moon. Dynamical systems theory notions are exploited to generate a seeding trajectory. Then, a nonlinear programming problem is formulated to obtain a multi-impulse transfer that meets mission constraints. A reference trajectory-aided reinforcement learning framework is conceptualized to train an agent that, following a perturbed trans-lunar injection burn, instantaneously re-plans the path and safely reaches the target state in an autonomous fashion.

Results demonstrate that the proposed approach yields a control policy capable of autonomous guidance in highly sensitive dynamical environments. Injecting a reference demonstration during training substantially improves learning; without it, the chaotic dynamics and vast search space inhibit policy convergence. Furthermore, robustness of the learned guidance policy has been validated by introducing additional control errors to the actions determined by the agent at evaluation time. Spacecraft operators could leverage this technique to streamline guidance operations, thereby reducing both cost and complexity.

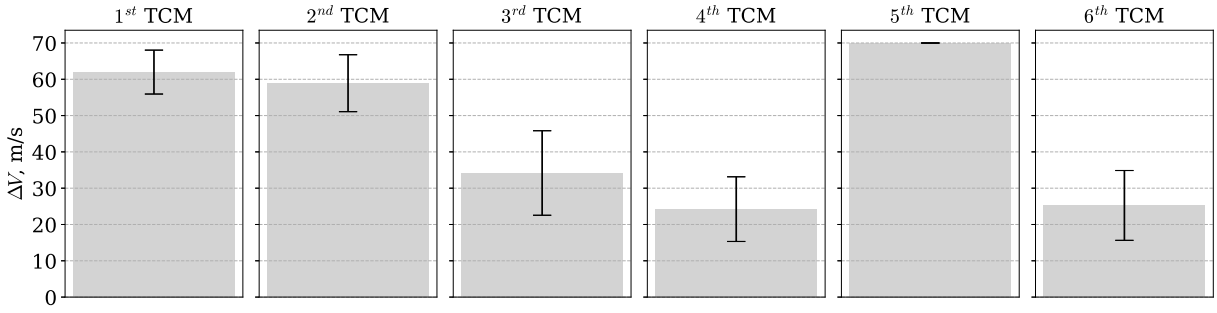

Fig. 6: Cost distribution among TLIs. 1σ values as per bars.

Bibliography

[1] M. Smith et al. The Artemis program: an overview of NASA's activities to return humans to the Moon. In *2020 IEEE Aerospace Conference*. IEEE, mar 2020.

[2] Z. Wang, Z. Meng, S. Gao, and J. Peng. Orbit design elements of Chang'e 5 mission. *Space: Science & Technology*, 2021, January 2021.

[3] B. Cheetham. Cislunar Autonomous Positioning System Technology Operations and Navigation Experiment (CAPSTONE). In *ASCEND 2021*. American Institute of Aeronautics and Astronautics, November 2021.

[4] Y. Song, J. Bang, J. Bae, and S. Hong. Lunar orbit acquisition of the Korea Pathfinder Lunar Orbiter: design reference vs actual flight results. *Acta Astronautica*, 213:336–343, December 2023.

[5] S. Mathavaraj and K. Negi. Chandrayaan-3 trajectory design: injection to successful landing. *Journal of Spacecraft and Rockets*, 62(1):159–166, January 2025.

[6] S. Sakai et al. Moon landing results of SLIM: a smart lander for investigating the Moon. *Acta Astronautica*, 235:47–54, October 2025.

[7] M. N. Sweeting. Modern small satellites-changing the economics of space. *Proceedings of the IEEE*, 106(3):343–361, March 2018.

[8] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross. Low energy transfer to the Moon. *Celestial Mechanics and Dynamical Astronomy*, 81(1/2):63–73, 2001.

[9] J. A. Starek, B. Açıkmeşe, I. A. Nesnas, and M. Pavone. *Spacecraft autonomy challenges for next-generation space missions*, pages 1–48. Springer Berlin Heidelberg, September 2015.

[10] D. A. Dei Tos and F. Topputo. On the advantages of exploiting the hierarchical structure of astrodynamical models. *Acta Astronautica*, 136:236–247, July 2017.

[11] F. Topputo, M. Vasile, and F. Bernelli-Zazzera. Low energy interplanetary transfers exploiting invariant manifolds of the restricted three-body problem. *The Journal of the Astronautical Sciences*, 53(4):353–372, December 2005.

[12] C. T. Campana, G. Merisio, and F. Topputo. Low-energy Earth–Moon transfers via Theory of Functional Connections and homotopy. *Celestial Mechanics and Dynamical Astronomy*, 136(3), May 2024.

[13] C. T. Campana and F. Topputo. Ephemeris refinement of low-energy Earth–Moon transfers via an astrodynamic model chain. *Acta Astronautica*, March 2025.

[14] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, mar 1998.

[15] D. Izzo, M. Märtens, and B. Pan. A survey on artificial intelligence trends in spacecraft guidance dynamics and control. *Astrodynamics*, 3(4):287–299, July 2019.

[16] C. T. Campana and F. Topputo. Clustering of Earth–Moon transfers in Sun-perturbed environment. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, volume 96, pages 1–8. IEEE, June 2024.

[17] T. R. Smith and N. Bosanac. Constructing motion primitive sets to summarize periodic orbit families and hyperbolic invariant manifolds in a multi-body system. *Celestial Mechanics and Dynamical Astronomy*, 134(1), February 2022.

[18] M. Shirobokov, S. Trofimov, and M. Ovchinnikov. Survey of machine learning techniques in spacecraft control design. *Acta Astronautica*, 186:87–97, sep 2021.

[19] A. Das-Stuart, K. C. Howell, and D. C. Folta. Rapid trajectory design in complex environments enabled by reinforcement learning and graph search strategies. *Acta Astronautica*, 171:172–195, jun 2020.

[20] A. Zavoli and L. Federici. Reinforcement learning for robust trajectory design of interplanetary missions. *Journal of Guidance, Control, and Dynamics*, 44(8):1440–1453, aug 2021.

[21] N. B. LaFarge, D. Miller, K. C. Howell, and R. Linares. Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment. *Acta Astronautica*, 186:1–23, sep 2021.

[22] C. J. Sullivan, N. Bosanac, and R. L. Anderson. Designing low-thrust transfers near Earth–Moon L2 via multi-objective reinforcement learning. *Journal of Spacecraft and Rockets*, 60(2):634–647, mar 2023.

[23] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross. *Dynamical systems, the three-body problem and space mission design*. Marsden Books, 2011.

[24] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. A Bradford Book, 2018.

[25] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization algorithms, 2017.

[28] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using Generalized Advantage Estimation, 2015.

[29] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization, 2014.

[30] C. T. Campana and F. Topputo. Low-energy transfers to Quasi-Halo orbit in a Sun-perturbed environment. In *35th AAS/AIAA Space Flight Mechanics Meeting*, 2025.

[31] E. Belbruno, M. Gidea, and F. Topputo. Weak stability boundary and invariant manifolds. *SIAM Journal on Applied Dynamical Systems*, 9(3):1061–1089, jan 2010.

[32] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: a survey of learning methods. *ACM Computing Surveys*, 50(2):1–35, April 2017.

[33] J. Ramírez, W. Yu, and A. Perrusquía. Model-free reinforcement learning from expert demonstrations: a survey. *Artificial Intelligence Review*, 55(4):3213–3241, October 2021.

[34] Todd Hester et al. Deep Q-learning from demonstrations, 2017.

[35] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations, 2017.

[36] A. Rajeswaran et al. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2017.

[37] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich. Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments, 2019.

[38] T. Huang, K. Chen, B. Li, Y. Liu, and Q. Dou. Demonstration-guided reinforcement learning with efficient exploration for task automation of surgical robot, 2023.