Autonomous Guidance between Quasiperiodic Orbits in Cislunar Space via Deep Reinforcement Learning

Lorenzo Federici[*] and Andrea Scorsoglio[†]
The University of Arizona, 1127 E. James E. Rogers Way, 85721, Tucson, AZ (USA)

Alessandro Zavoli [‡]
Sapienza University of Rome, Via Eudossiana 18, 00184, Rome, Italy

Roberto Furfaro[§]
The University of Arizona, 1127 E. James E. Rogers Way, 85721, Tucson, AZ (USA)

This paper investigates the use of reinforcement learning for the fuel-optimal guidance of a spacecraft during a time-free low-thrust transfer between two libration point orbits in the cislunar environment. To this aim, a deep neural network is trained via proximal policy optimization to map any spacecraft state to the optimal control action. A general-purpose reward is used to guide the network toward a fuel-optimal control law regardless of the specific pair of libration orbits considered and without the use of any ad-hoc reward shaping technique. Eventually, the learned control policies are compared with the optimal solutions provided by a direct method in two different mission scenarios, and Monte Carlo simulations are used to assess the policies' robustness to navigation uncertainties.

Nomenclature

A^{π}	=	advantage function
C	=	Jacobi integral
c	=	effective exhaust velocity, km/s
$d_{\min}$	=	minimum distance of the spacecraft trajectory from the target orbit, km
$\underset{\tau}{\mathbb{E}}[\boldsymbol{v}]$	=	expected value of a vector $\boldsymbol{v}$ with respect to τ
f	=	spacecraft dynamical model
J	=	merit index
m	=	spacecraft mass, kg
m_p	=	consumed propellant mass, kg

[*]Postdoctoral Research Associate, Department of Systems & Industrial Engineering, lorenzof@arizona.edu
[†]PhD Candidate, Department of Systems & Industrial Engineering, andreascorsoglio@arizona.edu
[‡]Assistant Professor, Department of Mechanical and Aerospace Engineering, alessandro.zavoli@uniroma1.it
[§]Professor, Department of Systems & Industrial Engineering, robertof@arizona.edu

Presented at the 2021 AAS/AIAA Astrodynamics Specialist Conference, Big Sky, Virtual, August 9 - 11, 2021, paper number AAS 21-610.

H	=	number of time-steps per episode
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	=	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
Q^{π}	=	Q-function
R	=	reward
$\boldsymbol{r}$	=	position vector, km
$\boldsymbol{x}$	=	state vector
$\tilde{\boldsymbol{x}}$	=	augmented state vector
K	=	total number of training steps
t	=	time, s
$\boldsymbol{T}$	=	thrust, kN
$\boldsymbol{u}$	=	control vector
V^{π}	=	value function
$\boldsymbol{v}$	=	velocity vector, km/s
$\boldsymbol{z}$	=	vector containing position and velocity
γ	=	discount factor
θ	=	neural network's parameters
μ	=	gravitational parameter, km^3/s^2
π	=	control policy
σ_x	=	standard deviation of Gaussian random variable x
τ	=	trajectory
$\boldsymbol{\Phi}$	=	discrete-time dynamical model
ω	=	angular velocity of the Earth-Moon system, rad/s

Subscripts

f	=	final value
h	=	value at h-th time-step
i	=	initial value
p_i	=	value referred to body p_i, $i = 1, \ldots, 3$
max	=	maximum value

Superscripts

$*$	=	reference value
$\star$	=	optimal value
T	=	transpose

I. Introduction

In the last decades, an increasing interest has been expressed in the exploitation of libration point orbits (LPOs) for innovative space missions, due to their peculiar characteristics. Indeed, their fixed relative configuration with respect to both primary celestial bodies (usually Sun-Earth or Earth-Moon) and their periodicity make those orbits especially suitable for a wide range of applications, such as deep-space observation, as for the NASA-ESA-CSA James Webb Space Telescope [1], or support to future manned missions to the moon, as for the proposed Lunar Orbital Platform-Gateway [2].

The design of optimal spacecraft trajectories in multi-body systems is a high-dimensional and highly-nonlinear problem, further complicated when low-thrust electric motors are considered for the orbital maneuvers. Dynamical system methods are typically used to construct zero-fuel transfer trajectories between LPOs, also known as heteroclinic connections, by looking for intersecting stable and unstable invariant manifolds [3, 4]. In the presence of low-thrust propulsion, special attainable sets are generally used in conjunction with invariant manifolds to define a first-guess trajectory [5], which is then optimized using indirect methods based on the Pontryagin maximum principle. However, indirect methods are highly sensitive to the first-guess solution, limiting their range of applicability to multi-body systems because of the chaotic dynamics [6]. In most cases, continuation methods are used in conjunction with indirect methods to provide robustness during the initialization process [7]. Designing closed-loop control laws for optimal transfers between distant orbits/states in the circular restricted three-body problem (CR3BP), able to run in real-time onboard, poses even additional challenges. Traditional Keplerian-based guidance approaches have been extensively used to provide spacecraft with closed-loop control capabilities [8], but they usually fail to deal with complex nonlinear dynamical models or they require abundant computational resources on the flight hardware. A few guidance techniques, mainly based on Floquet theory, have been specifically devised for spacecraft control in multi-body environments [9, 10], but they are limited to station-keeping applications around unstable periodic orbits.

In an effort to increase the autonomy and robustness to failures and model uncertainties of next-generation spacecraft, guidance systems based on artificial deep neural networks (DNNs) have become increasingly popular within the aerospace community. Such systems exploit the low computational times and the high accuracy in function approximation of DNNs to compute an optimal state-feedback control policy in real-time on the onboard hardware [11]. The DNN is trained offline (i.e., on the ground) to solve an optimal control problem (OCP) by leveraging on training data either provided by an "expert", typically a numerical solver, or collected by the network itself during repeated simulations of the considered mission scenario. In the former approach, known as behavioral cloning (BC), the learning process reduces to a supervised learning task: the goal is to generalize the expert behavior, provided in the form of a set of optimal trajectories, in order to correctly cope also with unseen scenarios. Some recent applications of BC to space guidance and control problems include, but are not limited to, the computation of minimum-time [12] and mass-optimal [13] interplanetary trajectories, the optimal control of a spacecraft during powered descent maneuvers [14], also starting just from raw camera images [15, 16], and the real-time generation of optimal trajectories for the atmospheric reentry of hypersonic vehicles [17, 18]. Conversely, in the latter approach, known as deep reinforcement learning (RL), learning occurs by trial and error rather than by looking at expert demonstrations. The policy network, or agent, repeatedly interacts with the considered environment and receives, at each training step, a scalar reward, which depends on the current spacecraft state and control. The control policy is progressively refined so as to maximize the cumulative reward received along a trajectory, which accounts for both the merit index and the violation of the constraints of the original OCP. In recent years, reinforcement learning has been extensively exploited for Lunar [19], planetary [20, 21] and asteroid [22] landing guidance, robust interplanetary trajectory design [23–25], and terminal guidance during interception [26, 27] or rendezvous [28] maneuvers. A systematic comparison between a BC and RL approach has been realized for the case of in-orbit proximity operations [29].

A third AI-based solution approach to optimal control problems that is worth mentioning is the use of physics-informed neural networks to solve the two-point boundary value problem coming from the application of the Pontryagin minimum principle. This approach has been recently applied to retrieve optimal planar orbit transfers [30].

In particular, several research papers dealt with the use of deep learning methods for the solution of optimal control problems in the cislunar environment. More precisely, D'Ambrosio et al. [31] computed a minimum-time energy-optimal transfer trajectory with control constraints between a L_1 Halo and a L_2 Halo orbit by using a combination of indirect methods and physics-informed neural networks. LaFarge et al. [32] used proximal policy optimization, a policy gradient RL algorithm, to devise a robust fuel-optimal guidance law between two Lyapunov orbits with the same energy in the presence of perturbed initial conditions and navigation errors, by leveraging on the knowledge of the zero-fuel heteroclinic connections between them. Sullivan et al. introduced a multi-reward proximal policy optimization algorithm to design both minimum-propellant and minimum-time low-thrust transfers between two southern L_2 Halo orbits [33, 34] and from an L_1 Lyapunov/Halo orbit to an L_2 Lyapunov/Halo orbit [35]. An algorithmic modification defined as the "moving reference" has been successively introduced by the same authors [36] to autonomously construct the reference optimal trajectories to recover with reinforcement learning during the network training. Guzzetti compared a Q-learning-based controller with more traditional station-keeping techniques for the maintenance of unstable symmetric periodic orbits within CR3BP dynamics [37]. A fuel-optimal impulsive station-keeping maneuvering strategy has been also found by Bonasera et al. via proximal policy optimization to stay on a L_2 quasi-Halo trajectory using a Sun–Earth–Moon point mass ephemeris model with solar radiation pressure [38]. Furthermore, an actor-critic reinforcement learning approach was successfully exploited by Scorsoglio et al. [39] and by Sullivan et al. [40] for close-proximity guidance applications in the cislunar environment. However, the great part of the works on RL for spacecraft guidance in the cislunar environment exploited a reward function that relies on environment-specific knowledge, provided externally to the controller in a BC-fashion, thus hindering the cross-applicability and robustness typical of model-free RL methods.

This paper, instead, aims at developing a simple and efficient learning procedure of general validity, which does not make use of any prior knowledge about the problem solution or of any pre-computed reference trajectory, as carried out in previous works on the topic [32], and which could be readily applied to devise an autonomous guidance system for cislunar orbit transfers. To this end, a novel reward function, which only relies on the minimum distance from the target orbit and the fuel consumption, is proposed. A state-of-the-art algorithm, named proximal policy optimization (PPO),[41] is used to optimize the guidance law within the RL framework. Numerical results are presented for two cases study, which refer to low-thrust transfers between planar LPOs in the Earth-Moon CR3BP. The solutions found with the RL approach are first compared with the optimal trajectories provided by a traditional direct optimization technique.[42]

The robustness of the obtained closed-loop guidance laws is then assessed through Monte Carlo campaigns performed in perturbed scenarios featuring increasing levels of navigation errors.

II. Problem Statement

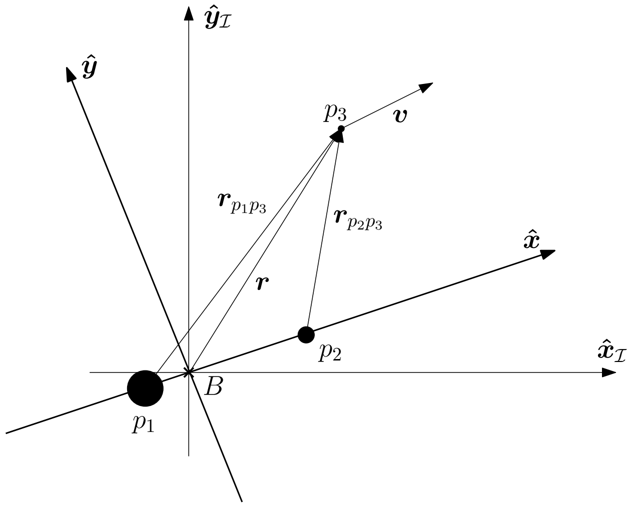
In this section, the spacecraft dynamics are described in detail, and the cislunar transfer problem is mathematically formulated as an optimal control problem.

A. Dynamical Model

The circular restricted three-body problem (CR3BP) is used to describe the system dynamics. In this model, the motion of the spacecraft, or third body (p_3), is determined by the gravitational attraction of two spherically symmetric celestial bodies, namely the primary body (p_1), with gravitational constant μ_{p_1}, and the secondary body (p_2), with gravitational constant μ_{p_2}. The two celestial bodies are supposed to move about a common center of mass (B) along circular orbits, with an angular velocity equal to $\omega = \sqrt{\frac{\mu_{p_1}+\mu_{p_2}}{d_{p_1 p_2}}}$, where $d_{p_1 p_2}$ is the (constant) distance between the two bodies. The mass of p_3 is considered negligible if compared to the masses of the other two bodies: as a consequence, the third body has no gravitational influence on the other two. In the Earth-Moon CR3BP, the primary body coincides with the Earth, while the secondary body with the Moon.

Let us introduce a non-inertial reference frame $\mathcal{N} = (B; \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ with origin in the center of mass B, the x-axis directed from p_1 to p_2, the z-axis directed as the angular momentum of the two primary bodies, and the y-axis forming a right-handed triad. So, frame $\mathcal{N}$ rotates counterclockwise about axis $\hat{\boldsymbol{z}}$ with angular velocity ω with respect to a correspondingly defined inertial reference frame $\mathcal{I} = (B; \hat{\boldsymbol{x}}_{\mathcal{I}}, \hat{\boldsymbol{y}}_{\mathcal{I}}, \hat{\boldsymbol{z}}_{\mathcal{I}})$. A view of the inertial and rotating reference frames from the positive z-axis is reported in Fig. 1.

A planar problem is considered in this manuscript for the sake of simplicity: the spacecraft is supposed to move in the same orbital plane as the two primary bodies. Nevertheless, as already shown in a number of research works by the authors on model-free reinforcement learning for spacecraft guidance [23, 29], the extension to a three-dimensional problem is straightforward and does not represent a challenge for the presented solution method. An increase in the



$\hat{\boldsymbol{y}}_{\mathcal{I}}$
$\hat{\boldsymbol{y}}$
p_3
$\boldsymbol{v}$
$\boldsymbol{r}_{p_1p_3}$
$\boldsymbol{r}_{p_2p_3}$
$\hat{\boldsymbol{x}}$
$\boldsymbol{r}$
p_2
$\hat{\boldsymbol{x}}_{\mathcal{I}}$
B
p_1

Fig. 1 Inertial $\mathcal{I}$ and rotating $\mathcal{N}$ reference frames.

average neural network training time should be anyway expected when facing the three-dimensional problem as a direct consequence of the increase in the number of inputs and outputs (and, thus, network dimensions).

The spacecraft, with initial mass m_0, is equipped with a low-thrust engine with maximum thrust $T_{\max}$ and effective exhaust velocity c. The control thrust at any time can be expressed as

$$\boldsymbol{T} = T_x \hat{\boldsymbol{x}} + T_y \hat{\boldsymbol{y}} \tag{1}$$

So, the following condition must hold along the whole spacecraft trajectory

$$\|\boldsymbol{T}\| \leqslant T_{\max} \tag{2}$$

The spacecraft state $\boldsymbol{x}$ at any time t in frame $\mathcal{N}$ is completely described by its position $\boldsymbol{r} = [x \; y]^T$, velocity $\boldsymbol{v} = [v_x \; v_y]^T$ and mass m, that is, $\boldsymbol{x} = [\boldsymbol{r}^T \; \boldsymbol{v}^T \; m]^T$. The time evolution of the spacecraft state is governed by the equations of motion $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{T})$, written as

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{v}_x = 2v_y + x - \frac{(1-\mu)(x+\mu)}{r_{p_1p_3}^3} - \frac{\mu(x-1+\mu)}{r_{p_2p_3}^3} + \frac{T_x}{m} \\ \dot{v}_y = -2v_x + y - \frac{(1-\mu)y}{r_{p_1p_3}^3} - \frac{\mu y}{r_{p_2p_3}^3} + \frac{T_y}{m} \\ \dot{m} = -\frac{\|\boldsymbol{T}\|}{c} \end{cases} \tag{3}$$

where the distances of the spacecraft from the primary and secondary body are evaluated as

$$r_{p_1p_3} = \sqrt{(x+\mu)^2 + y^2} \tag{4}$$

$$r_{p_2p_3} = \sqrt{(x-(1-\mu))^2 + y^2} \tag{5}$$

and where $\mu = \frac{\mu_{p_2}}{\mu_{p_1}+\mu_{p_2}}$ indicates the mass ratio of the primary and secondary body.

All the physical quantities in Eq. (3) are made nondimensional by using as the unit of length the distance between the primary and secondary body $l^* = d_{p_1p_2}$, as the unit of time the orbital period of the secondary body divided by 2π, $t^* = 1/\omega$, as the unit of velocities $v^* = l^*/t^*$ and as the unit of masses the initial mass of the spacecraft $m^* = m_0$.

The "free" CR3BP equations of motion, that is, without the control term, admit a well-known integral, the Jacobi integral C, defined as

$$C(\boldsymbol{x}) = 2\left[\frac{1-\mu}{r_{p_1p_3}} + \frac{\mu}{r_{p_2p_3}} + 0.5(x^2+y^2)\right] - \left(v_x^2 + v_y^2\right) \tag{6}$$

The Jacobi constant represents the specific mechanical energy of the spacecraft in the rotating frame, and it maintains constant along any ballistic trajectory. The free CR3BP equations of motion also admit five different equilibrium points in the orbital plane of the primary and secondary body, called Lagrangian points: three collinear points along the axis joining the primary with the secondary, denoted by L_1, L_2, and L_3, and two equilateral points, specular with respect to the x-axis, denoted by L_4 and L_5, at the vertex of two equilateral triangles with the two bodies.

The values of all the Earth-Moon CR3PB constants and of the spacecraft's engine characteristics in nondimensional units are summarized in Table 1. The same data used in Ref. [43] have been considered in this manuscript for the sake of a fair comparison.

μ	l^*, km	t^*, s	v^*, km/s	m^*, kg	$T_{\max}$	c
1.2151×10^{-2}	3.8440×10^5	3.7519×10^5	1.0245	1000	4×10^{-2}	28.7306

Table 1 Values of the Earth-Moon system's constants and of the spacecraft's characteristics.

B. Optimal Control Problem

The spacecraft is supposed to be initially in a given state $\boldsymbol{x}_i = [\boldsymbol{r}_i^T\ \boldsymbol{v}_i^T\ 1]^T$ in a Lyapunov orbit around L_1, denoted as Ly_1. The objective is to find the optimal control thrust $\boldsymbol{T}^\star$ that drives the spacecraft, within a maximum transfer time $t_{\max}$, towards a given Lyapunov orbit around L_2, named Ly_2, while minimizing the propellant consumption. The corresponding optimal control problem can be mathematically formulated as follows

$$\mathcal{P}: \begin{cases} \max\limits_{\boldsymbol{T}} m(t_f) \\ \text{s.t.}: \ \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{T}), & t \in [0, t_f] \\ \qquad \|\boldsymbol{T}\| \leqslant T_{\max}, & t \in [0, t_f] \\ \qquad \boldsymbol{x}(0) = \boldsymbol{x}_i \in \mathrm{Ly}_1 \\ \qquad \boldsymbol{x}(t_f) \in \mathrm{Ly}_2 \\ \qquad t_f \leqslant t_{\max} \end{cases} \tag{7}$$

Figure 2 shows an example of these two Lyapunov orbits.

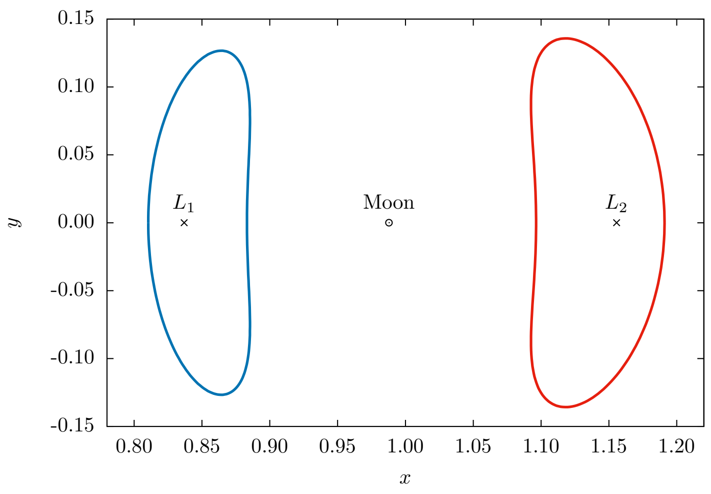


Fig. 2 Initial (in blue) and target (in red) Lyapunov orbits.

III. Reinforcement Learning

The fundamental concepts behind reinforcement learning are outlined in this section. First, a mathematical formulation of the optimal control problem as a Markov decision process (MDP) is introduced. Then, the general-purpose delayed reward function devised for the cislunar transfer problem at hand is described. Eventually, details on the architecture used for the control policy network and how learning is achieved through proximal policy optimization are presented.

A. Markov Decision Process

To properly match the mathematical framework of reinforcement learning, the OCP in Eq. (7) is recast as an equivalent discrete-time MDP. To this aim, time is discretized over a grid with $H + 1$ uniformly spaced points between the initial time $t_i = 0$ and the maximum time $t_{\max}$: $0 = t_0 < t_1 < \ldots < t_H = t_{\max}$. Subscript "$h$" will be used to identify quantities at time t_h, with $h = 0, \ldots, H$. The time-step of the grid is $\Delta t = t_{\max}/H$. A sequence of H time steps is indicated as an episode.

An "augmented" spacecraft state $\tilde{\boldsymbol{x}}$ is introduced, which includes also time t and the Jacobi integral C. The augmented state at time t_{h+1} is obtained by numerically propagating the state $\boldsymbol{x}_h$ between time t_h and time t_{h+1} through Eqs. (3), assuming that the thrust $\boldsymbol{T} = \boldsymbol{T}_h$ is constant in magnitude and direction during the whole time-step, and then by evaluating the Jacobi integral at the final time of the trajectory segment. Hence

$$\tilde{\boldsymbol{x}}_{h+1} = \Phi(\tilde{\boldsymbol{x}}_h, \boldsymbol{T}_h) = [\boldsymbol{x}_{h+1}^T \; C(\boldsymbol{x}_{h+1}) \; t_h + \Delta t]^T \tag{8}$$

A closed-loop control law is used to drive the spacecraft toward the desired final orbit. Specifically, the controller returns at every time step t_h a control action $\boldsymbol{u}_h$ as the output of a state-feedback control policy π: $\boldsymbol{u}_h = \pi(\tilde{\boldsymbol{x}}_h)$. In order to ensure that the thrust $\boldsymbol{T}_h$ meets the imposed constraint on the maximum modulus (see Eq. (2)), a peculiar definition of the action has been devised. Specifically, the action $\boldsymbol{u}_h$ returned by π at time t_h is

$$\boldsymbol{u}_h = [\tilde{u} \;\; \sin\phi \;\; \mathrm{sign}(\cos\phi)] \in [-1, 1] \times [-1, 1] \times \{-1, 1\} \tag{9}$$

where ϕ is the angle between $\boldsymbol{T}_h$ and the x-axis of frame $\mathcal{N}$. The actual thrust $\boldsymbol{T}_h$ can be easily computed from $\boldsymbol{u}_h$ using Eq. (1), with

$$\|\boldsymbol{T}\| = \frac{\tilde{u} + 1}{2} T_{\max} \tag{10}$$

$$T_x = \|\boldsymbol{T}\| \; \mathrm{sign}(\cos\phi)\sqrt{1 - \sin^2\phi} \tag{11}$$

$$T_y = \|\boldsymbol{T}\| \sin\phi \tag{12}$$

The parameter $\tilde{u} \in [-1,\ 1]$ has been preferred over the thrust modulus as RL algorithms typically work better with parameters whose value is centered about zero. This mixed-integer definition of the action has the advantage, over a more traditional Cartesian definition $\boldsymbol{u} = [T_x \;\, T_y]$, to inherently meet the constraint on the maximum thrust provided by the spacecraft engine, without the need to introduce additional penalty terms inside the reward function. Moreover, as opposed to a definition in spherical coordinates $\boldsymbol{u} = [\tilde{u} \;\, \phi]$, the discontinuity of the control around $\phi = 0$ is eliminated.

At the end of every time step t_h, the controller receives also a reward $R_h = R(\tilde{\boldsymbol{x}}_h, \tilde{\boldsymbol{x}}_{h+1}, \boldsymbol{u}_h)$, which measures the "desirability" of the transition from state $\tilde{\boldsymbol{x}}_h$ to state $\tilde{\boldsymbol{x}}_{h+1}$ due to control $\boldsymbol{u}_h$. The reward function devised for this application will be described in detail in Sec. III.B.

The goal in an MPD is to find the control policy $\pi^\star$ that maximizes a performance index J, which is the expected discounted return $G(\tau)$ obtained along a trajectory $\tau = \{(\tilde{\boldsymbol{x}}_0, \boldsymbol{u}_0), \ldots, (\tilde{\boldsymbol{x}}_{H-1}, \boldsymbol{u}_{H-1}), \tilde{\boldsymbol{x}}_H\}$

$$J(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi}[G(\tau)] = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{h=0}^{H-1} \gamma^h R_h\right] \tag{13}$$

where $\gamma \in (0, 1]$ is a discount factor, usually close to 1, which is used to give more importance to short-term rewards.

Overall, the discrete-time MDP can be mathematically formulated as follows

$$\mathcal{M}: \begin{cases} \max\limits_{\pi} J(\pi) \\ \text{s.t.: } \tilde{\boldsymbol{x}}_{h+1} = \Phi(\tilde{\boldsymbol{x}}_h, \boldsymbol{u}_h), \qquad h = 0, \ldots, H-1 \\ \qquad \boldsymbol{u}_h = \pi(\tilde{\boldsymbol{x}}_h), \qquad\qquad h = 0, \ldots, H-1 \\ \qquad \tilde{\boldsymbol{x}}_0 = [\boldsymbol{x}_i^T \; C(\boldsymbol{x}_i) \; 0]^T \\ \qquad t_H = t_{\max} \end{cases} \tag{14}$$

B. Reward Function

A suitable definition of the reward function is the key to the success of an RL procedure in a given environment. For this reason, devising a satisfactory-good reward function is also the most challenging part of using RL as an optimization method.

When tackling an OCP with RL, one of the major limitations of RL is its poor constraint-handling capability. Indeed, as stated by Eq. (14), MDPs allow only scalar rewards. Thus, terminal and path constraints cannot be separately accounted for, and all constraint violations must be included in the reward function as weighted penalty terms.

In this respect, control/state path constraints are often well handled by RL. Indeed, whenever one of these constraints is not met either the agent receives an immediate negative reward, thus allowing for frequent rewards, or the episode is forced to terminate prematurely, thus informing the agent that it should not visit that specific state again. Instead, teaching the agent to effectively cope with terminal constraints, as in this study, is a much more challenging task. In this case, the cumulative effect of all actions can be generally judged only at the end of the episode, when the agent receives a delayed negative reward as a response to the terminal constraint violation.

In both scenarios, the weighting factors play a pivotal role, as they control the relative importance of the various terms (merit index and constraint violations). As a consequence, a proper (and typically very time-consuming) tuning of these weights is generally required to avoid the creation of a number of spurious sub-optimal solutions the search could get trapped into.

An environment-dependent "shaping" of the reward function [44] is often adopted in RL applications to space guidance problems [20] to simplify the agent's task during training and speed up the learning procedure. In this case, an *a priori* knowledge about the, more or less approximate, shape that the optimal, or a good suboptimal, guidance profile should have is exploited to devise a frequent definition of the reward, which keeps the agent close to this reference path, thus ensuring the satisfaction of the problem constraints. While performing at first glance, reward shaping has the drawback of requiring some information about the structure of the optimal solution, or even the exact solution of relaxed versions (e.g., without uncertainties) of the problem at hand [35, 43]. Furthermore, being the shaping problem-specific, even modest variation in the mission parameters may turn a previously well-devised reward function completely ineffective, typically leading to the need of devising another reward function from scratch, which makes the overall procedure very time-consuming.

In cislunar transfer problems, proper shaping of the reward function is a non-trivial task, as, in principle, neither the problem solution nor an approximation of the optimal trajectory is known in advance. Furthermore, slight changes to the initial and/or target spacecraft conditions usually bring large changes to the optimal solution, as a result of the chaotic three-body problem dynamics. Thus, a non-sparse reward function that both guarantees the control policy optimality and it is applicable to a wide class of transfer problems in cislunar space, without the use of any *a priori* information, is hard, if not completely impossible, to devise. Instead, this manuscript proposes a simple and effective sparse reward function. This reward can be easily applied to other problems in the cislunar environment (e.g., different pairs of Lyapunov orbits) without significant effort. Indeed, without any information about the shape of the optimal trajectory, the performance of the agent (i.e., the final distance of the spacecraft from the target Lyapunov orbit) can be just determined at the end of the episode.

Specifically, a minimum-propellant low-thrust transfer from a specified initial state $\boldsymbol{x}_i \in \mathrm{Ly}_1$ to any state in Ly_2 in a maximum mission time $t_{\max}$ is searched for. So, the reward function must be devised in such a way as to both minimize the total propellant consumption and effectively handle periodic terminal conditions. To this aim, a delayed reward is used: the agent receives a non-zero reward only at the end of each episode, that is, at step H. The terminal reward R_H is made up of two terms

$$R_H = -0.1 \max\{0, d_{\min} - \varepsilon\} - m_p \tag{15}$$

In particular, $d_{\min}$ is the minimum distance, in space $\boldsymbol{r}$, $\boldsymbol{v}$, between the whole spacecraft trajectory and the target Lyapunov orbit. Let us denote with $\boldsymbol{z} = [\boldsymbol{r}^T \ \boldsymbol{v}^T]^T$ the vector made up of position and velocity at a given time (that is, the state vector without the mass). By indicating with $\boldsymbol{z}_h^{nn} \in \mathrm{Ly}_2$ the nearest neighbor of $\boldsymbol{z}_h$ along the target Lyapunov orbit, $d_{\min}$ can be evaluated as

$$d_{\min} = \min_{h \in \{0,\dots,H\}} d_h \tag{16}$$

$$d_h = \frac{\left\| z_h - z_h^{nn} \right\|}{\left\| z_h^{nn} \right\|} \tag{17}$$

$$z_h^{nn} = \arg\min_{z \in \mathrm{Ly}_2} \| z_h - z \| \tag{18}$$

The nearest neighbors z_h^{nn}, $h = 0, \dots, H$, are computed by exploiting a k-d tree partitioning of the space Ly_2, which allows speeding up the nearest neighbor search procedure by quickly eliminating large portions of the search space, thanks to the tree properties. An example of computation of $d_{\min}$ for a sample trajectory is shown in Fig. 3, by considering position only for visualization purposes.

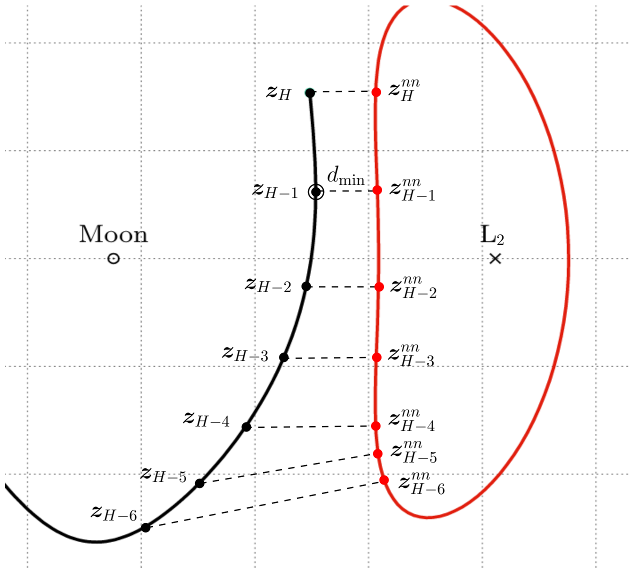


Fig. 3 Schematic of the evaluation process for the minimum distance $d_{\min}$.

The final time is set to be the time at which the closest approach between the spacecraft and the target orbit occurs

$$t_f = t_{\hat{h}} \qquad \text{with} \qquad \hat{h} = \arg\min_h d_h \tag{19}$$

The second term in the reward, m_p, instead, indicates the total propellant mass consumed up to time t_f, that is

$$m_p = m_{\hat{h}} - m_0 \tag{20}$$

A tolerance on the violation of the terminal constraint $\varepsilon = 1 \times 10^{-3}$ is allowed, as done in similar works. [23] The weighting factor 0.1 between the mass and constraint violation term in the reward function has been selected empirically after a tuning procedure with different values (e.g., 0.01, 0.1, 1, and 10), as it showed the best performance on average.

It is worthwhile to note that the reward function presented above does not exactly match an MDP formulation, as the reward at the last time step depends on the spacecraft states at all the previous times. However, a "pure" MDP formulation can be easily restored if the minimum distance between the spacecraft trajectory and the target orbit up to time t_h, $d_{\min,h}$, is added to the augmented state vector $\tilde{\boldsymbol{x}}_h$. In this way, the minimum distance at time t_{h+1} can simply be evaluated as $d_{\min,h+1} = \min\left\{d_{\min,h}, d_{h+1}\right\}$, so that the reward at any time only depends on the state-action-next-state transition, and the Markov property is retained.

C. Policy Network

In policy gradient deep reinforcement learning a deep neural network with parameters (i.e., weights and biases) θ is used to parameterize the control policy π, and defined as policy network π_θ. So, the optimal policy $\pi^\star$ is found by directly looking for the optimal network's parameters $\theta^\star$, thus turning the problem into a parametric optimization one.

A stochastic policy is commonly used to favor a wider exploration of the search space in the first phases of the training procedure. In this case, the expression $\pi(\boldsymbol{u}|\tilde{\boldsymbol{x}})$ indicates the probability of taking action $\boldsymbol{u}$ in state $\tilde{\boldsymbol{x}}$. A diagonal Gaussian action distribution is used in this work: given the current state $\tilde{\boldsymbol{x}}_h$ as input, the policy network returns the mean value of the action components $\boldsymbol{\mu}_{\theta,h}$, while the standard deviations $\boldsymbol{\sigma}_{\theta,h}$ are treated as additional network biases, thus not dependent on the current input. In the training phase, the actual action is sampled from the Gaussian distribution. So

$$\boldsymbol{u}_h \sim \pi_\theta(\cdot|\tilde{\boldsymbol{x}}_h) = \mathcal{N}(\boldsymbol{\mu}_{\theta,h}, \boldsymbol{\Sigma}_{\theta,h}) \tag{21}$$

with $\boldsymbol{\Sigma}_{\theta,h} = \text{diag}(\boldsymbol{\sigma}_{\theta,h}\boldsymbol{\sigma}_{\theta,h}^T)$ the covariance matrix. After training or during independent evaluations, instead, the policy is deployed as a deterministic law by switching off exploration: $\boldsymbol{u}_h = \boldsymbol{\mu}_{\theta,h}$.

A fully-connected network architecture with 3 hidden layers has been used, where the width (i.e., the number of neurons) of the hidden layers is an increasing function of the number of network inputs and outputs. This architecture comes from the idea that the network complexity should increase with the dimension of the function domain to correctly approximate the relations between the different input variables and the output values. This network configuration has been already adopted in similar research works on deep reinforcement learning for spacecraft guidance, achieving high performance [20, 23, 25]. Specifically, the input layer consists of $n_i = 7$ neurons, the output layer of $n_o = 3$ neurons, and the three hidden layers have width $l_1 = k_f n_i$, $l_2 = \sqrt{l_1 l_3}$, and $l_3 = k_f n_o$, respectively. The proportionality factor $k_f = 5$ has been preferred over the one originally proposed in the literature (i.e., $k_f = 10$) as it showed faster convergence properties during a preliminary tuning procedure. All the neurons in the hidden layers feature the hyperbolic tangent (tanh) as the activation function, while the neurons in the input and output layers use a linear (lin) activation function. A schematic of such a policy network is shown in Fig. 4.

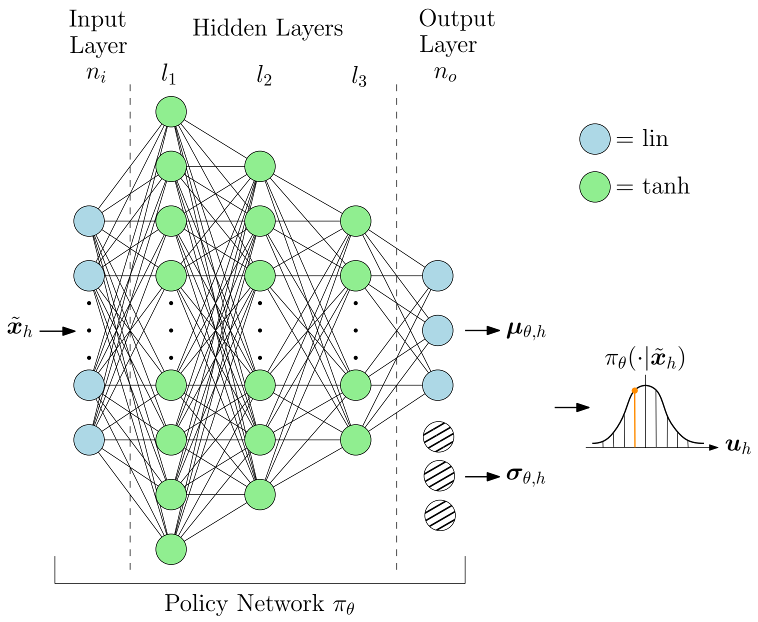

Input Layer
n_i
Hidden Layers
l_1
l_2
l_3
Output Layer
n_o
= lin
= tanh
$\tilde{\boldsymbol{x}}_h$
$\boldsymbol{\mu}_{\theta,h}$
$\boldsymbol{\sigma}_{\theta,h}$
$\pi_\theta(\cdot|\tilde{\boldsymbol{x}}_h)$
$\boldsymbol{u}_h$
Policy Network π_θ

Fig. 4 Policy network.

D. Proximal Policy Optimization

In this manuscript, the policy network training is performed by proximal policy optimization (PPO), a model-free, policy gradient algorithm developed in 2017 by OpenAI and almost unanimously considered as the state-of-the-art for control problems with both discrete and continuous action spaces [41].

In PPO, a so-called "clipped policy objective function" J^{clip} is used in place of J to constrain the policy π_θ to stay in a small range ϵ, named clip range, around its previous value $\pi_{\theta_{\text{old}}}$, and avoid too large policy updates between consecutive training iterations, which usually cause a performance collapse. J^{clip} is defined as

$$J^{\text{clip}}(\theta, \theta_{\text{old}}) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\frac{1}{H}\sum_{h=0}^{H-1}\min\left\{r_h(\theta)A^{\pi_\theta}, \text{clip}(\tilde{r}_h(\theta,\theta_{\text{old}}), 1-\epsilon, 1+\epsilon)A^{\pi_\theta}\right\}\right] \tag{22}$$

In particular, $r_h(\theta)$ represents the probability ratio between the updated and the previous policy

$$\tilde{r}_h(\theta,\theta_{\text{old}}) = \frac{\pi_\theta(\tilde{\boldsymbol{x}}_h)}{\pi_{\theta_{\text{old}}}(\tilde{\boldsymbol{x}}_h)} \tag{23}$$

while A^{π_θ}, denoted as advantage function, quantifies by how much the total reward improves by taking a specific action

$\boldsymbol{u}$ in the state $\tilde{\boldsymbol{x}}$, instead of randomly selecting the action according to π_θ. Specifically

$$A^{\pi_\theta}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) = Q^{\pi_\theta}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) - V^{\pi_\theta}(\tilde{\boldsymbol{x}}) \tag{24}$$

$$Q^{\pi_\theta}(\boldsymbol{x}, \boldsymbol{u}) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{h'=h}^{H-1} \gamma^{h'-h} R_{h'} \middle| \tilde{\boldsymbol{x}}_h = \tilde{\boldsymbol{x}}, \boldsymbol{u}_h = \boldsymbol{u}\right] \tag{25}$$

$$V^{\pi_\theta}(\tilde{\boldsymbol{x}}) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{h'=h}^{H-1} \gamma^{h'-k} R_{h'} \middle| \tilde{\boldsymbol{x}}_h = \tilde{\boldsymbol{x}}\right] \tag{26}$$

with $Q^{\pi_\theta}(\tilde{\boldsymbol{x}}, \boldsymbol{u})$ and $V^{\pi_\theta}(\tilde{\boldsymbol{x}})$ the Q-function and value function, respectively.

In order to compute the advantage function during policy optimization, a second DNN, named Critic, is trained concurrently to the policy network, named Actor. Just like the Actor, the Critic receives as input the current state $\tilde{\boldsymbol{x}}_h$, but returns as output an estimate $\hat{V}_h$ of the value function $V^{\pi_\theta}(\tilde{\boldsymbol{x}}_h)$. In this work, the same DNN is shared for both the control policy and value function estimation (i.e., a fourth output $\hat{V}$ is added to the three policy-related outputs in the network shown in Fig 4). Once an approximation of the value function is known, the advantage function can be computed by using the so-called generalized advantage estimator (GAE) [45]

$$\hat{A}_h = \sum_{h'=h}^{H-1} (\gamma\lambda)^{h'-h} \left(R_{h'} + \gamma \hat{V}_{h'+1} - \hat{V}_{h'}\right) \tag{27}$$

where λ is a hyperparameter named the GAE factor.

Since the DNN returns also an estimate of the value function, an additional term e related to the quality of the value function estimate $\hat{V}$ must be added to the clipped objective function J^{clip} in order to correctly update the set of network parameters θ. So, a so-called "surrogate" objective function J^{ppo} is used as merit index

$$J^{\text{ppo}}(\theta, \theta_{\text{old}}) = J^{\text{clip}}(\theta, \theta_{\text{old}}) - c_v e(\theta) \tag{28}$$

where

$$e(\theta) = \frac{1}{2} \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\frac{1}{N}\sum_{h=0}^{H-1}\left(V_\theta(\tilde{\boldsymbol{x}}_h) - \sum_{h'=h}^{H-1} \gamma^{h'-h} R_{h'}\right)^2\right] \tag{29}$$

is the mean-squared error on the value function and c_v is a hyperparameter named value function coefficient.

As in all policy-gradient RL algorithms, in PPO two different phases alternate at every training iteration. In the policy rollout phase, n_w worker agents run the most up-to-date policy network in parallel in as many independent realizations of the environment for n_e complete episodes each. Concurrently, a deterministic version of the policy is run in an evaluation environment (equal to the training one, in this case), and the corresponding value of the surrogate objective is saved. The $N = n_w n_e$ training trajectories just collected are then randomly split into n_b mini-batches and used in the policy update phase to update the network parameters θ by performing, sequentially, n_{sga} stochastic gradient

Algorithm 1 Proximal policy optimization (PPO)

```
1:  function PPO($\mathcal{M}$, net)
      ▷ Input: environment $\mathcal{M}$, policy network architecture net
      ▷ Hyperparams: clip range $\epsilon$, episodes per rollout $N$, learning rate $\alpha$, training iterations $K$
2:      Randomly initialize $\theta_{(0)}$ with dimensions defined by net
3:      Initialize the batch $\mathcal{J} \leftarrow \varnothing$
4:      for $k \leftarrow 0, K-1$ do
5:          Initialize the batches $\mathcal{D}_{(k)} \leftarrow \varnothing$, $\mathcal{R}_{(k)} \leftarrow \varnothing$, $\mathcal{A}_{(k)} \leftarrow \varnothing$
6:          for $i \leftarrow 1, N$ do
7:              Run $\pi_{\theta_{(k)}}$ in the environment for a episode and store the new
                    trajectory in $\mathcal{D}_{(k)}$ and rewards-to-go in $\mathcal{R}_{(k)}$
8:              Compute the approximate advantages $\hat{A}_h$ based on the current
                    value function estimate $\hat{V}$, and store them in $\mathcal{A}_{(k)}$
9:          end for
10:         Run $\boldsymbol{\mu}_{\theta_{(k)}}$ in the environment for a episode and store the value
              of the surrogate objective in $\mathcal{J}$
11:         Randomly split the data in $\mathcal{D}_{(k)}$, $\mathcal{R}_{(k)}$ and $\mathcal{A}_{(k)}$ in $n_b$ mini-batches
12:         $\theta_{\text{new}} \leftarrow \theta_{(k)}$
13:         for $j \leftarrow 1, n_b$ do
14:             for $s \leftarrow 1, n_{\text{sga}}$ do
15:                 Update the network parameters through SGA on an estimate
              of the surrogate objective in Eq. (28) evaluated using the data
              in the $j$-th mini-batch

                        $\theta_{\text{new}} \leftarrow \theta_{\text{new}} + \alpha_{(k)} \nabla_\theta \hat{J}^{\text{ppo}}(\theta, \theta_{\text{new}})\big|_{\theta_{\text{new}}}$

16:             end for
17:         end for
18:         $\theta_{(k+1)} \leftarrow \theta_{\text{new}}$
19:     end for
20:     return $\theta^\star = \arg\min \mathcal{J}$
21: end function
```

Fig. 5 Pseudo-code for proximal policy optimization algorithm.

ascent (SGA) iterations on each mini-batch. A constant, or slowly decreasing, learning rate α is used to define the step size of each SGA iteration. The training process is stopped when a total number of training iterations equal to K is reached. The value of the parameters that yielded the maximum value of the surrogate objective in the evaluation environment is returned as putative optimum. A pseudo-code of the PPO algorithm is reported in Fig. 5.

E. Algorithm settings

The numerical results presented in the next section have been obtained by using Ray [46], an open-source library that contains a wide set of improved and highly-parallel algorithms for reinforcement learning, among which a state-of-the-art implementation of PPO. The main hyperparameters of PPO (namely, the clip range, the learning rate, and the number of mini-batches for SGD) have been tuned by trial-and-error on the problem at hand, and the best-found values are reported in Table 2.

Table 2 PPO hyperparameters.

Hyper-parameter	Symbol	Value
Training iterations	K	1500
Discount factor	γ	0.9999
GAE factor	λ	0.99
Clip range	ϵ	0.05
Value function coefficient	c_v	0.5
Parallel workers	n_w	56
Number of episodes per worker	n_e	10
Number of mini-batches	n_b	7
SGA epochs per update	n_{sga}	50

A decreasing, piecewise linear learning rate α has been used

$$\alpha_{(k)} = \alpha_i + (\alpha_{i+1} - \alpha_i)\frac{k - k_i}{k_{i+1} - k_i}, \qquad i = 0, \ldots, 3 \tag{30}$$

The values of the training step k_i and learning rates α_i at the five switching points are reported in Table 3.

Table 3 Learning rate trend.

i	0	1	2	3	4
k_i	0	375	750	1125	1500
α_i	5×10^{-4}	5×10^{-5}	1×10^{-5}	5×10^{-6}	1×10^{-6}

IV. Numerical Results

This section presents the numerical results for two study cases. The first one refers to a transfer from a L_1 Lyapunov orbit (Ly$_1$) to a L_2 Lyapunov orbit with roughly the same energy level (Ly$_2^{(A)}$). In the second case, the transfer starts from the same orbit but a L_2 Lyapunov orbit with slightly higher energy is targeted (Ly$_2^{(B)}$).

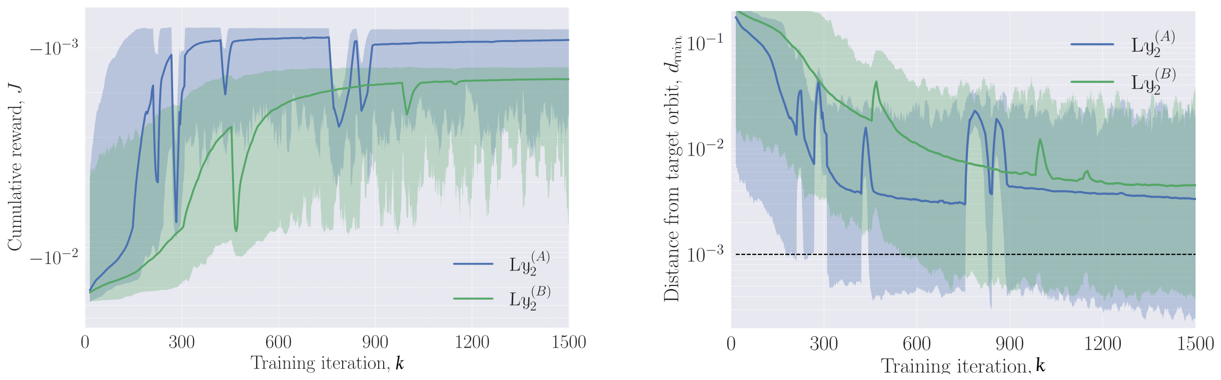
Orbit	$\boldsymbol{r}_i$	$\boldsymbol{v}_i$	T	C
Ly$_1$	$[0.8104000\ 0\ 0]^T$	$[0\ \ +0.2681030\ 0]^T$	2.9771360	3.1237338
Ly$_2^{(A)}$	$[1.1910000\ 0\ 0]^T$	$[0\ \ -0.2373133\ 0]^T$	3.4937505	3.1238893
Ly$_2^{(B)}$	$[1.1880000\ 0\ 0]^T$	$[0\ \ -0.2114158\ 0]^T$	3.4619924	3.1342709

Table 4 Lyapunov orbits used as study cases.

Table 4 presents the position $\boldsymbol{r}_i$ and velocity $\boldsymbol{v}_i$ vectors used to generate the three Lyapunov orbits here considered, as well as their orbital period T and Jacobi constant C. The position and velocity referred to orbit Ly_1 correspond to the initial spacecraft state. The maximum flight time is set to $t_{\max} = 6$, and a time grid with $H = 40$ steps has been used. So, the time step is $\Delta t = 0.15$.

In the following sections, the two trajectories found in a deterministic scenario are compared with the optimal ones provided by a direct optimization technique. Then, the robustness of the closed-loop control policies in the presence of navigation uncertainties is investigated by means of Monte Carlo campaigns.

A. Training



(a) Cumulative reward vs training iteration. **(b) Distance from target orbit vs training iteration.**

Fig. 6 Convergence behavior of the RL policies during training. Values are smoothed with a moving average on a 15-step window.

The evolution of the cumulative reward and of the terminal error (i.e., the minimum distance from the target orbit) during the training of the two RL policies related to the mission towards $\mathrm{Ly}_2^{(A)}$ and $\mathrm{Ly}_2^{(B)}$ are presented in Figure 6. The shaded region around each curve represents the range of variation of the depicted quantity over the n_w parallel training environments. Both test metrics improve almost monotonically on average during the training, with just a few drops, which are related to the stochastic nature of the control policy. The width of the shaded regions suggests that the considered test metrics vary significantly among the different parallel training environments. This wide range of variation is probably a direct consequence of the high complexity of the optimization problem, featuring several locally optimal solutions.

Note that the network performance in term of cumulative reward worsen when a target orbit with different energy is considered. This is mainly due to the increase in propellant mass required for reaching the orbit $\mathrm{Ly}_2^{(B)}$, as the spacecraft must change its energy during the transfer. On the contrary, the minimum distance of the trajectory from the target orbit reaches similar values in the two scenarios.

Training the network took about 6 hours on a computer equipped with a 56-core Intel(R) Xeon(R) E5-2680 v4

@2.40 GHz, for either test case. However, it is here worthwhile to remark that, once the trained network is deployed (either on a target flight hardware or when used in simulations) the computational cost for evaluating the optimal control at any time step is almost negligible, being equivalent to that of a few small-size matrix multiplications.

B. Optimal Trajectories

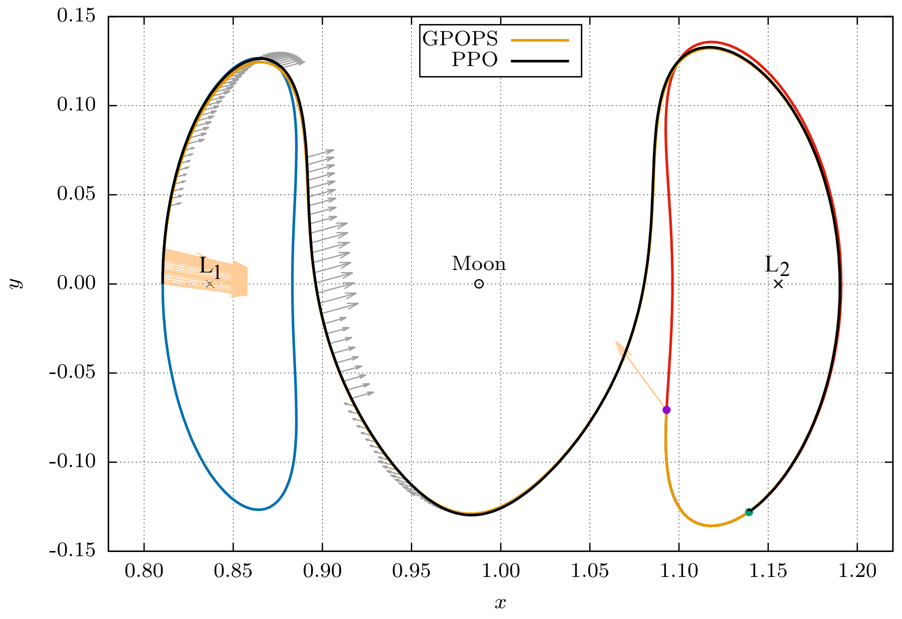


Fig. 7 Optimal trajectories found by PPO and GPOPS between same-C Lyapunov orbits.

Figures 7 and 8 show the optimal trajectories (black curves) found by PPO towards $\mathrm{Ly}_2^{(A)}$ and $\mathrm{Ly}_2^{(B)}$, respectively, and the corresponding control along the transfers (gray arrows). For the sake of comparison, the optimal solutions (light-orange curves and arrows) obtained by a direct optimization technique with GPOPS-II [42] are also reported. An overview of the obtained results is presented in Table 5, where the final trajectory return G, flight time t_f, consumed propellant mass m_p (by supposing $m_0 = 1000$ kg), and minimum distance from the target orbit $d_{\min}$ for both PPO and GPOPS solutions are reported. One can easily note that the trajectories found by PPO and GPOPS in the case of equal-C Lyapunov orbits are almost superimposed. However, the two control laws are quite different from each other. In particular, while the optimal control by GPOPS features a typical bang-off-bang profile with two clear burns, one at the departure and one at the arrival, the PPO solution front-loads the thrust in the first part of the trajectory but never reaches the maximum thrust level. On the contrary, the spacecraft is injected into a stable manifold of the target Lyapunov orbit and the second half of the transfer is purely ballistic. This difference in the control law has a major impact on the transfer time, which decreases by about 13% from GPOPS (5.99) to PPO (5.23), and on the overall propellant consumption, which is almost four times larger (+0.45 kg), but still remaining less than 0.1% of the initial

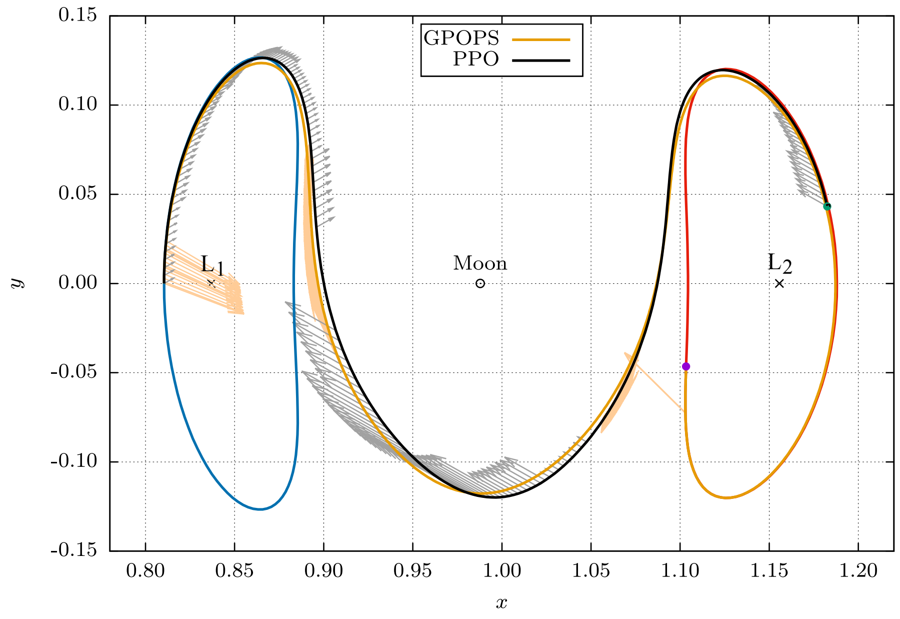


Fig. 8 Optimal trajectories found by PPO and GPOPS between different-C Lyapunov orbits.

spacecraft mass. When considering the transfer towards $\mathrm{Ly}_2^{(B)}$, the differences between PPO and GPOPS trajectories are more apparent. The considerations concerning the different use of the propulsion done for the previous test case still apply, but the terminal PPO burn arc is significantly larger. The overall consumption is also larger for PPO (almost double) with respect to GPOPS, but a solution with a shorter transfer time is indeed found. In either study case, the terminal error d_{min} is well below the desired threshold.

When considering the numerical differences between the PPO and GPOPS solutions, one should take into account the fact that the two dynamical models are not exactly the same. In fact, the RL framework requires a discrete-time model, and the thrust is assumed constant in magnitude and direction over each time step. On the opposite, GPOPS relies on a continuous-time formulation of the optimal control problem and the thrust can be switched on/off at any time. For this reason, GPOPS is able to find a solution where the spacecraft thrusts for arbitrarily short periods at maximum throttle, which is indeed the proper use of propulsion, and this explains the lower propellant consumption. In theory, the use of a smaller step size in the MDP formulation would make the dynamic model more similar to a continuous-time one, thus potentially improving the accuracy and optimality of the control policy. However, this also makes the problem progressively harder to be solved via RL as it increases the search space dimensions. As a result, a trade-off on the step size is typically necessary to find a good balance between the accuracy of the solution and the complexity of the resulting control problem.

Despite PPO being inherently unable to converge to the optimal control found by GPOPS, the trajectories found are quite close both in shape and in final spacecraft mass to the optimal solutions, and both of them converge, with satisfactory high precision, to the final target orbit. This confirms the validity of the reward function here devised, which is capable of guiding the network towards the solution of a time-free problem and with terminal periodic constraints without the need for any *a-priori* information about the putative optimal solution. Future works will attempt at defining the RL action space in such a way as to have thrusting arcs of arbitrary time length, with the aim of attaining a propellant consumption closer to that of the optimal solution.

A major advantage of the proposed procedure over a direct optimization method such as GPOPS-II is the computational time in inference, i.e., when the controller is deployed onboard. Indeed, the time required by a direct optimization method is highly dependent on the initial guess of the optimization procedure, and it generally varies from a few seconds to several minutes on the basis of how close this guess is to the optimal trajectory. When used onboard the spacecraft for guidance purposes, a direct method is typically run within a model predictive control (MPC) framework to compute a new optimal trajectory at each guidance step starting from an updated estimate of the spacecraft state. In this case, the initial guess of every optimization run is the optimal trajectory computed at the previous guidance step, which, depending on the step size and uncertainty level, is usually close enough to the new solution to maintain the computational time in the order of seconds and make the methodology suitable for real-time applications. Conversely, the onboard computational time of a neural network controller is only dependent on the neural network dimensions and on the hardware architecture, being just determined by one forward pass of the network per guidance step. This operation only involves matrix multiplications, thus it is extremely fast, usually in the order of a few milliseconds.

Table 5 Optimal trajectories overview.

Target orbit	G	t_f		m_p, kg		$d_{\min}$	
		PPO	GPOPS	PPO	GPOPS	PPO	GPOPS
$\mathrm{Ly}_2^{(A)}$	5.560×10^{-4}	5.233	5.986	0.556	0.108	5.940×10^{-4}	$< 1 \times 10^{-7}$
$\mathrm{Ly}_2^{(B)}$	1.441×10^{-3}	4.183	5.895	1.441	0.661	7.467×10^{-4}	$< 1 \times 10^{-7}$

C. Autonomous Guidance in Presence of Navigation Errors

During the actual space mission, the spacecraft trajectory may deviate significantly from the nominal one because of the presence of external disturbances and model uncertainties that may affect the system dynamics. One of the most common sources of uncertainty for the guidance system is the presence of navigation errors, which are related to measurement noise and/or inaccuracies in the orbital determination that lead to imperfect knowledge of the spacecraft state.

To assess the robustness of the obtained closed-loop control policies against navigation uncertainties, a moderate level of Gaussian noise has been added to the deterministic observation of the spacecraft position, velocity, and mass that the onboard guidance system receives at time t_h:

$$\tilde{\boldsymbol{x}}_h = [(\boldsymbol{x}_h + \delta \boldsymbol{x}_h)^T \; C_h \; t_h]^T \tag{31}$$

being:

$$\delta \boldsymbol{x}_h = [\delta \boldsymbol{r}_h^T \; \delta \boldsymbol{v}_h^T \; \delta m_h] \sim \mathcal{N}(\boldsymbol{0}_5, \boldsymbol{\Sigma}) \in \mathbb{R}^5 \tag{32}$$

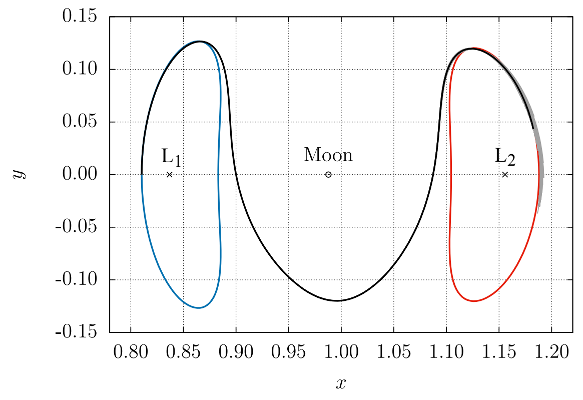
where $\boldsymbol{\Sigma} = \operatorname{diag}\left(\sigma_r^2 \boldsymbol{I}_2, \sigma_v^2 \boldsymbol{I}_2, \sigma_m^2\right)$ is the covariance matrix, with σ_r, σ_v and σ_m the standard deviations on the observed spacecraft position, velocity, and mass, and with $\boldsymbol{I}_n$ (respectively, $\boldsymbol{0}_n$) indicating an identity (respectively, null) matrix of dimension $n \times n$ (respectively, $n \times 1$). The Jacobi constant C_h is evaluated using the spacecraft state $\boldsymbol{x}_h + \delta \boldsymbol{x}_h$.

Table 6 Results of the Monte Carlo campaign.

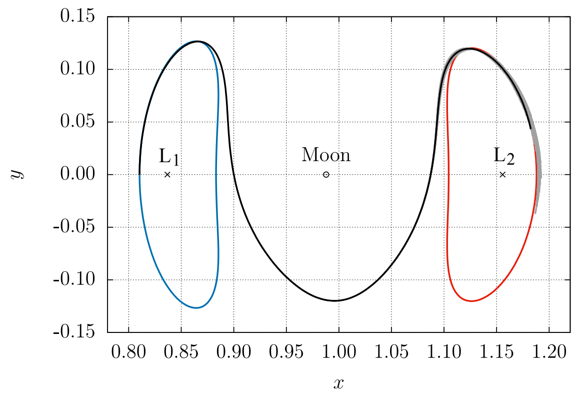
Target orbit	Uncertanty level	t_f		m_p, kg		$d_{\min}$, 10^{-3}			ΔC, 10^{-4}		
		mean	std	mean	std	1σ	2σ	3σ	1σ	2σ	3σ
$\mathrm{Ly}_2^{(A)}$	$\times 1$	4.84	0.22	0.56	2.00×10^{-4}	0.99	1.61	1.93	3.66	3.67	3.69
	$\times 2$	4.69	0.25	0.56	3.00×10^{-4}	1.49	2.35	2.94	3.66	3.69	3.72
	$\times 5$	4.46	0.27	0.56	8.00×10^{-4}	2.29	3.88	4.91	3.68	3.75	3.83
	$\times 10$	4.27	0.30	0.56	1.70×10^{-3}	3.46	5.96	7.15	3.70	3.84	4.00
$\mathrm{Ly}_2^{(B)}$	$\times 1$	4.20	0.14	1.46	8.56×10^{-2}	1.99	3.27	7.35	7.31	11.45	15.36
	$\times 2$	4.15	0.23	1.44	1.18×10^{-1}	3.21	7.95	11.24	10.66	16.79	18.11
	$\times 5$	3.90	0.29	1.36	1.13×10^{-1}	6.34	12.78	16.21	16.71	19.24	20.78
	$\times 10$	3.70	0.32	1.32	1.03×10^{-1}	10.38	19.03	24.50	17.89	21.91	24.58

By assuming a base error level defined by $\sigma_r = 10\,\mathrm{km}$, $\sigma_v = 10\,\mathrm{cm/s}$ and $\sigma_m = 100\,\mathrm{g}$, the controller was tested on four uncertainty levels, with magnitude 1, 2, 5, or 10 times that of the base level. For each study case, four Monte Carlo campaigns, involving the deployment of the optimal PPO policy in 1000 realizations of the corresponding observation-uncertain environment, have been performed. Table 6 reports, for each mission scenario and uncertainty level considered, the mean value and standard deviation of the transfer time t_f and of the consumed propellant mass m_p. The minimum distance from the target orbit $d_{\min}$ and the difference between the actual and the target Jacobi constant ΔC are also presented, in terms of the values below which fall the 68.3% (1σ), 95.5% (1σ), and 99.7% (1σ) of the trajectories, respectively.

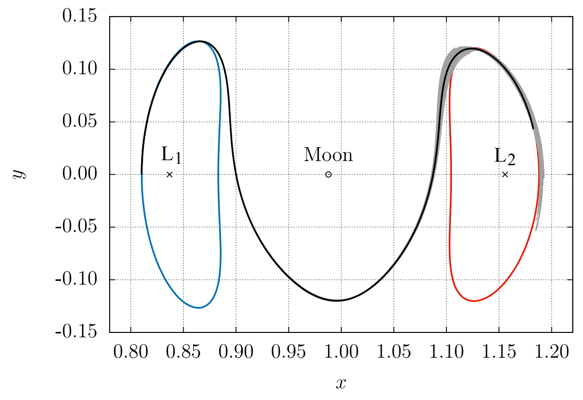
The first study case (i.e., mission towards $\mathrm{Ly}_2^{(A)}$) shows that the trained policy is quite robust to navigation errors, even when an uncertainty level 10 times the base one is accounted for. The trajectory never deviates much from the nominal one and the terminal error is consistently low in all rollouts. Since the differences between the Monte Carlo trajectories are not appreciable, the corresponding plots are here omitted for the sake of conciseness. The variations in propellant consumption are negligible too.



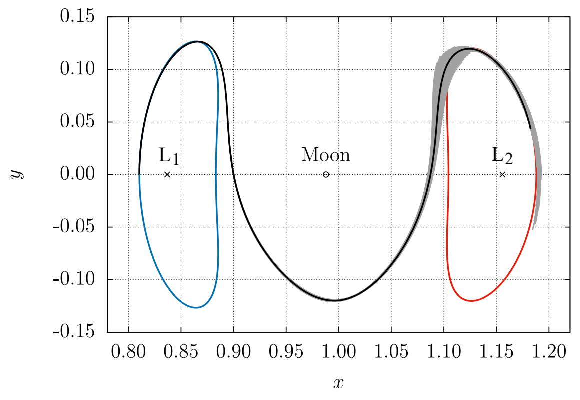
(a) Small uncertainty ($\times 1$).



(b) Medium uncertainty ($\times 2$).



(c) High uncertainty ($\times 5$).



(d) Very high uncertainty ($\times 10$).

Fig. 9 Monte Carlo simulations for the mission toward Ly$_2^{(B)}$ with different navigation error levels.

Conversely, in the second case study (mission towards $\text{Ly}_2^{(B)}$), the trained policy seems robust only to low levels of navigation errors. In fact, both the final error in the Jacobi constant and in the spacecraft state grows considerably when a 5x or 10x uncertainty level is considered. This is apparent in Fig. 9, which shows the Monte Carlo trajectories (light-gray curves), alongside the optimal trajectory by PPO (black curve) for different levels of navigation errors. In both study cases, the mean transfer time decreases as the level of uncertainty increases. This phenomenon is probably related to the increase in errors in terminal conditions, but further investigations are due for a better understanding of this unexpected effect.

As a final remark, it is worthwhile to stress that the main objective of this study was not to train a single control policy to solve different transfer problems when deployed onboard, but, instead, to devise a general training methodology capable of finding optimal control laws for different mission scenarios. For this reason, the same transfer environments were used both for training and evaluation. Nevertheless, the robustness of the guidance laws has been tested against the presence of navigation uncertainties not included in the training environment, which indeed represents a more realistic operative scenario for the proposed guidance algorithm.

V. Conclusion

A novel approach for generating an autonomous guidance law for cislunar orbit transfers based on reinforcement learning (RL) is presented in this paper. Proximal policy optimization (PPO) is used to find the optimal control law along a low-thrust orbit transfer from a Lyapunov orbit around the L_1 Lagrangian point to a Lyapunov orbit around the L_2 Lagrangian point in the Earth-Moon system.

The major contribution of this paper is the definition of a reward function of general validity, which does not rely on any previous knowledge about the problem solution. Being solely dependent on the minimum distance from the target orbit and on the fuel consumption, the proposed reward function allows for the solution of time-free problems with periodic terminal conditions, and can be easily extended to other cislunar transfer scenarios. This is a significant improvement over previous works, where the guidance law was heavily reliant on a preliminary-computed deterministic solution of the transfer problem, either obtained by using dynamical system theory or by solving an optimal control problem. Moreover, a peculiar definition of the action space is devised, which allows enforcing implicitly the constraint on the maximum thrust while overcoming discontinuity problems typical of more traditional definitions of the control (e.g., in spherical coordinates).

The preliminary numerical results shown in this paper confirm the validity of the proposed approach in two study cases, where the target orbit has the same energy level or slightly higher energy than the departure one, respectively. When considering the problem of finding a nominal optimal trajectory, RL converges to an interesting solution, quite different from the (putative) global optimum indicated by a direct optimization method (GPOPS-II). In particular, RL attempts to front-load the control effort in the first part of the trajectory, flying along a stable manifold of the target orbit in the second half of the transfer. While slightly sub-optimal in terms of propellant expenditure, this trajectory is close in shape to that obtained by GPOPS.

The optimality of the nominal solution is not the only aspect to take into account when considering the problem of autonomous guidance along cislunar orbit transfers. In this respect, a Monte Carlo analysis was carried out to show the robustness of the PPO-trained policy to navigation errors. To this end, several levels of uncertainties were considered. Very promising results were found; in fact, the transfer between equal-C Lyapunov orbits is almost insensitive to this source of errors when using as guidance law the policy trained by RL, thus justifying the increase in propellant of the nominal solution. Similar results were obtained for the study case concerning the transfer between Lyapunov orbits of different energy, but here medium/high levels of navigation errors may significantly affect the transfer, leading to non-negligible terminal errors. The robustness performance of the policy could of course be improved by training the policy directly in a stochastic environment featuring the same sources and level of perturbations as expected in the actual mission. Future investigations should be thus aimed at improving the robustness of the policy by considering the presence of other sources of uncertainties, among which are unmodeled errors in the dynamical model and control actuation errors. Also, the methodology should be extended to a three-dimensional scenario to support the investigation of transfers from/towards Halo or Near-Rectilinear orbits.

References

[1] Gardner, J. P., Mather, J. C., Clampin, M., Doyon, R., Greenhouse, M. A., Hammel, H. B., Hutchings, J. B., Jakobsen, P., Lilly, S. J., Long, K. S., et al., "The james webb space telescope," *Space Science Reviews*, Vol. 123, 2006, pp. 485–606. https://doi.org/10.1007/s11214-006-8315-7.

[2] Lo, M., and Ross, S., "The Lunar L1 Gateway : portal to the stars and beyond," , 2001. https://doi.org/2014/40516, URL https://hdl.handle.net/2014/40516.

[3] Gómez, G., and Masdemont, J., "Some zero cost transfers between libration point orbits," *Advances in the Astronautical Sciences*, Vol. 105, No. 2, 2000, pp. 1199–1216.

[4] Koon, W. S., Lo, M. W., Marsden, J. E., and Ross, S. D., "Heteroclinic connections between periodic orbits and resonance transitions in celestial mechanics," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, Vol. 10, No. 2, 2000, pp. 427–469. https://doi.org/10.1063/1.166509.

[5] Chupin, M., Haberkorn, T., and Trélat, E., "Low-Thrust Lyapunov to Lyapunov and Halo to Halo with L^2-Minimization," *arXiv preprint arXiv:1511.02089*, 2016.

[6] Trélat, E., "Optimal control and applications to aerospace: some results and challenges," *Journal of Optimization Theory and Applications*, Vol. 154, No. 3, 2012, pp. 713–758. https://doi.org/10.1007/s10957-012-0050-5.

[7] Gergaud, J., and Haberkorn, T., "Homotopy method for minimum consumption orbit transfer problem," *ESAIM: Control, Optimisation and Calculus of Variations*, Vol. 12, No. 2, 2006, pp. 294–310. https://doi.org/10.1051/cocv:2006003.

[8] Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics, revised edition*, American Institute of Aeronautics and Astronautics, 1999.

[9] Simó, C., Gómez, G., Llibre, J., Martinez, R., and Rodriguez, J., "On the optimal station keeping control of halo orbits," *Acta Astronautica*, Vol. 15, No. 6-7, 1987, pp. 391–397. https://doi.org/10.1016/0094-5765(87)90175-5.

[10] Howell, K. C., and Pernicka, H. J., "Station-keeping method for libration point trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 1, 1993, pp. 151–159. https://doi.org/10.2514/3.11440.

[11] Izzo, D., Märtens, M., and Pan, B., "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodynamics*, 2018, pp. 1–13. https://doi.org/10.1007/s42064-018-0053-6.

[12] Cheng, L., Wang, Z., Jiang, F., and Zhou, C., "Real-Time Optimal Control for Spacecraft Orbit Transfer via Multiscale Deep Neural Networks," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 55, No. 5, 2019, pp. 2436–2450. https://doi.org/10.1109/TAES.2018.2889571.

[13] Izzo, D., and Öztürk, E., "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. https://doi.org/10.2514/1.G005254.

[14] Sánchez-Sánchez, C., and Izzo, D., "Real-time optimal control via deep neural networks: study on landing problems," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 5, 2018, pp. 1122–1135. https://doi.org/10.2514/1.G002357.

[15] Furfaro, R., Bloise, I., Orlandelli, M., Di Lizia, P., Topputo, F., Linares, R., et al., "Deep learning for autonomous lunar landing," *Advances in the Astronautical Sciences*, Vol. 167, 2018, pp. 3285–3306.

[16] Ghilardi, L., D'Ambrosio, A., Scorsoglio, A., Furfaro, R., Linares, R., and Curti, F., "Image-based Optimal Powered Descent Guidance via Deep Recurrent Imitation Learning," *2020 AAS/AIAA Astrodynamics Specialist Conference*, Virtual Lake Tahoe, 2020.

[17] Shi, Y., and Wang, Z., "Onboard Generation of Optimal Trajectories for Hypersonic Vehicles Using Deep Learning," *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 400–414. https://doi.org/10.2514/1.A34670.

[18] Shi, Y., and Wang, Z., "A Deep Learning-Based Approach to Real-Time Trajectory Optimization for Hypersonic Vehicles," *AIAA Scitech 2020 Forum*, 2020. https://doi.org/10.2514/6.2020-0023.

[19] Scorsoglio, A., D'Ambrosio, A., Ghilardi, L., Gaudet, B., Curti, F., and Furfaro, R., "Image-Based Deep Reinforcement Meta-Learning for Autonomous Lunar Landing," *Journal of Spacecraft and Rockets*, Vol. 59, No. 1, 2022, pp. 153–165. https://doi.org/10.2514/1.A35072.

[20] Gaudet, B., Linares, R., and Furfaro, R., "Deep reinforcement learning for six degree-of-freedom planetary landing," *Advances in Space Research*, Vol. 65, No. 7, 2020, pp. 1723–1741. https://doi.org/10.1016/j.asr.2019.12.030.

[21] Furfaro, R., Scorsoglio, A., Linares, R., and Massari, M., "Adaptive generalized ZEM-ZEV feedback guidance for planetary landing via a deep reinforcement learning approach," *Acta Astronautica*, Vol. 171, 2020, pp. 156–171. https://doi.org/10.1016/j.actaastro.2020.02.051.

[22] Gaudet, B., Linares, R., and Furfaro, R., "Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations," *Acta Astronautica*, Vol. 171, 2020, pp. 1–13. https://doi.org/10.1016/j.actaastro.2020.02.036.

[23] Zavoli, A., and Federici, L., "Reinforcement Learning for Robust Trajectory Design of Interplanetary Missions," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 8, 2021, pp. 1440–1453. https://doi.org/10.2514/1.G005794.

[24] Boone, S., Bonasera, S., McMahon, J. W., Bosanac, N., and Ahmed, N. R., "Incorporating Observation Uncertainty into Reinforcement Learning-Based Spacecraft Guidance Schemes," *AIAA SCITECH 2022 Forum*, 2022. https://doi.org/10.2514/6.2022-1765.

[25] Federici, L., Scorsoglio, A., Zavoli, A., and Furfaro, R., "Meta-reinforcement learning for adaptive spacecraft guidance during finite-thrust rendezvous missions," *Acta Astronautica*, Vol. 201, 2022, pp. 129–141. https://doi.org/10.1016/j.actaastro.2022.08.047.

[26] Gaudet, B., Furfaro, R., Linares, R., and Scorsoglio, A., "Reinforcement Metalearning for Interception of Maneuvering Exoatmospheric Targets with Parasitic Attitude Loop," *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 386–399. https://doi.org/10.2514/1.A34841.

[27] Federici, L., Scorsoglio, A., Ghilardi, L., D'Ambrosio, A., Benedikter, B., Zavoli, A., and Furfaro, R., "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. https://doi.org/10.2514/1.G006832.

[28] Broida, J., and Linares, R., "Spacecraft rendezvous guidance in cluttered environments via reinforcement learning," *Advances in the Astronautical Sciences*, Vol. 168, 2019, pp. 1777–1788.

[29] Federici, L., Benedikter, B., and Zavoli, A., "Deep Learning Techniques for Autonomous Spacecraft Guidance During Proximity Operations," *Journal of Spacecraft and Rockets*, Vol. 58, No. 6, 2021, pp. 1774–1785. https://doi.org/10.2514/1.A35076.

[30] Schiassi, E., D'Ambrosio, A., Drozd, K., Curti, F., and Furfaro, R., "Physics-Informed Neural Networks for Optimal Planar Orbit Transfers," *Journal of Spacecraft and Rockets*, Vol. 59, No. 3, 2022, pp. 834–849. https://doi.org/10.2514/1.A35138.

[31] D'Ambrosio, A., Schiassi, E., Curti, F., and Furfaro, R., "Physics-Informed Neural Networks Applied to a Series of Constrained Space Guidance Problems," *31st AAS/AIAA Space Flight Mechanics Meeting*, 2021.

[32] LaFarge, N. B., Miller, D., Howell, K. C., and Linares, R., "Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment," *Acta Astronautica*, Vol. 186, 2021, pp. 1–23. https://doi.org/10.1016/j.actaastro.2021.05.014.

[33] Sullivan, C. J., and Bosanac, N., "Using Multi-Objective Deep Reinforcement Learning to Uncover a Pareto Front in Multi-Body Trajectory Design," *2020 AAS/AIAA Astrodynamics Specialist Conference*, Virtual Lake Tahoe, 2020.

[34] Sullivan, C. J., Bosanac, N., and Anderson, R. L., "Designing Low-Thrust Transfers near Earth–Moon L2 via Multi-Objective Reinforcement Learning," *Journal of Spacecraft and Rockets*, Vol. 60, No. 2, 2023, pp. 634–647. https://doi.org/10.2514/1.A35463.

[35] Sullivan, C. J., Bosanac, N., Anderson, R. L., Mashiku, A. K., and Stuart, J. R., "Exploring Transfers between Earth-Moon Halo Orbits via Multi-Objective Reinforcement Learning," *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–13. https://doi.org/10.1109/AERO50100.2021.9438267.

[36] Sullivan, C. J., Bosanac, N., Mashiku, A. K., and Anderson, R. L., "Multi-Objective Reinforcement Learning for Low-Thrust Transfer Design between Libration Point Orbits," *2021 AAS/AIAA Astrodynamics Specialist Conference*, 2021.

[37] Guzzetti, D., "Reinforcement learning and topology of orbit manifolds for stationkeeping of unstable symmetric periodic orbits," *Advances in the Astronautical Sciences*, Vol. 171, 2020, pp. 3747–3766.

[38] Bonasera, S., Bosanac, N., Sullivan, C. J., Elliott, I., Ahmed, N., and McMahon, J. W., "Designing Sun–Earth L2 Halo Orbit Stationkeeping Maneuvers via Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 2, 2023, pp. 301–311. https://doi.org/10.2514/1.G006783.

[39] Scorsoglio, A., Furfaro, R., Linares, R., and Massari, M., "Actor-critic reinforcement learning approach to relative motion guidance in near-rectilinear orbit," *Advances in the Astronautical Sciences*, Vol. 168, 2019, pp. 1737–1756.

[40] Sullivan, C. J., and Bosanac, N., "Using Reinforcement Learning to Design a Low-Thrust Approach into a Periodic Orbit in a Multi-Body System," *AIAA Scitech 2020 Forum*, 2020. https://doi.org/10.2514/6.2020-1914.

[41] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[42] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 41, No. 1, 2014, pp. 1–37. https://doi.org/10.1145/2558904.

[43] LaFarge, N. B., Miller, D., Howell, K. C., and Linares, R., "Guidance for Closed-Loop Transfers using Reinforcement Learning with Application to Libration Point Orbits," *AIAA Scitech 2020 Forum*, 2020. https://doi.org/10.2514/6.2020-0458.

[44] Mataric, M. J., "Reward Functions for Accelerated Learning," *Machine Learning Proceedings 1994*, Morgan Kaufmann, San Francisco (CA), 1994, pp. 181–189. https://doi.org/10.1016/B978-1-55860-335-6.50030-1.

[45] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[46] Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., et al., "Ray: A distributed framework for emerging AI applications," *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 561–577.