Acta Astronautica 227 (2025) 40–56

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Research paper

Propulsive landing of launchers' first stages with Deep Reinforcement Learning

Davide Iafrate [a,*], Andrea Brandonisio [b], Robert Hinz [c], Michèle Lavagna [b]

[a] Space Engineering, Politecnico di Milano, Italy
[b] Aerospace Department, Politecnico di Milano, Italy
[c] Deimos Space, Spain

ARTICLE INFO

Keywords:
Retropropulsive landing
Launchers
Machine learning
Reinforcement learning
GNC
Controls

ABSTRACT

The planetary landing problem is gaining relevance in the space sector, spanning a wide range of applications from unmanned probes landing on other planetary bodies to reusable first and second stages of launcher vehicles. In the existing methodology there is a lack of flexibility in handling complex non-linear dynamics, in particular in the case of non-convexifiable constraints. It is therefore crucial to assess the performance of novel techniques and their advantages and disadvantages. The purpose of this work is the development of an integrated 6-DOF guidance and control approach based on reinforcement learning of deep neural network policies for fuel-optimal planetary landing control, specifically with application to a launcher first-stage terminal landing, and the assessment of its performance and robustness. 3-DOF and 6-DOF simulators are developed and encapsulated in MDP-like (Markov Decision Process) industry-standard compatible environments. Particular care is given in thoroughly shaping reward functions capable of achieving the landing both successfully and in a fuel-optimal manner. A cloud pipeline for effective training of an agent using a PPO reinforcement learning algorithm to successfully achieve the landing goal is developed.

1. Introduction

The planetary landing problem has been rising in relevance in recent years in an effort to make space more economically accessible through reusable launchers, and to enable improved exploration of planetary bodies of the solar system, focusing on landing sites with high scientific importance. The problem consists of achieving a successful touchdown on a planetary body within prescribed location and velocity bounds, subject to terminal attitude constraints. This goal can be optimized in terms of time, propellant consumption and/or terminal error minimization. Throughout the years several techniques have been developed, starting from simple fixed-guidance solutions to advanced optimization techniques. The first applications were investigated for the lunar landing program during the first space race: the Apollo mission used a polynomial trajectory guidance that used different polynomials for each of the landing phases, limiting the overall computational resources required [1]. Afterwards, to improve the fuel efficiency of the trajectory, optimal *guidance* laws have emerged for point-mass dynamics, and, in the case of linear cases, non-atmospheric dynamics [2]: these approaches need a mapping of the guidance objective to actuator actions through a tracking controller. Nowadays, a critical improvement is the ability to optimize more complex dynamics and to enforce constraints, indeed, recent approaches to the problem characterized by these two aspects have used Second Order Cone Programming methods by *convexifying* the non-convex constraints through a relaxation [3–5]. This is accepted since it has been proven in [3] that the *fuel-optimal* solution to the relaxed problem coincides with the fuel-optimal solution of the original non-convex problem. The approach here developed is limited by two main factors: only linear dynamics can be optimized and not all the constraints can be convexified. Furthermore, this methodology usually exploits conical glideslope constraints, which are not appropriate for trajectories which need to avoid steep obstacles close to the landing pad. Despite these aforementioned drawbacks, this approach has been applied to point-mass dynamics such as in is G-FOLD [6], and successively extended to 6-DOF with *successive convexification (SCVX)* methods that iteratively optimize the convexified problems for guidance computation to generate 6-DOF optimal open-loop trajectories [7,8]. At present time the state-of-the-art approach splits the process into sequential steps, first computing a reference trajectory starting from an estimate of the state given by the navigation subsystem and then tracking it with a controller. This raises the question of whether it is possible to implement a unified approach to

* Corresponding author.
E-mail addresses: davide.iafrate@polimi.it (D. Iafrate), andrea.brandonisio@polimi.it (A. Brandonisio), robert.hinz@deimos-space.com (R. Hinz), michelle.lavagna@polimi.it (M. Lavagna).

https://doi.org/10.1016/j.actaastro.2024.11.028
Received 30 June 2024; Received in revised form 7 November 2024; Accepted 13 November 2024
Available online 22 November 2024

Nomenclature

$\vec{a}_{targ}$	Target acceleration [m/s^2]
C_A	Aerodynamic coefficients' matrix [−]
CoM	Center of Mass
CoP	Center of Pressure
$\mathcal{F}B$	Body-fixed Reference Frame
$\mathcal{F}I$	Inertial Reference Frame
g_0	Standard gravitational constant at sea level [m/s^2]
Isp	Specific impulse of the engine [s]
ISA	International Standard atmosphere
J	Inertia matrix of the rocket [kg m^2]
l	Length of the rocket [m]
m	Mass of the vehicle [kg]
$\vec{r}$	Position vector of the center of mass [m]
$Ra \rightarrow b$	Rotation matrix from reference frame $\mathcal{F}a$ to $\mathcal{F}b$
$R(\vec{s'}, \vec{s}, \vec{a})$	Reward function [−]
r	Base radius of the rocket [m]
S_A	Aerodynamic reference surface [m^2]
$\vec{v}$	Velocity vector of the center of mass [m/s]
$vtarg$	Target velocity [m/s]
$[\psi, \theta, \phi]$	Yaw, Pitch and Roll angles [rad]
$[\psi, \theta, \phi]$lim	Limit Yaw, Pitch and Roll angles [rad]
ρ	Density
$\vec{\omega}$	Angular velocity vector [rad/s]
Subscripts	
I	Inertial reference frame
B	Body reference frame

Guidance and Control, as well as the potential level of performance and robustness it can attain. One technique that allows to achieve this type of integrated guidance and control policies is Deep Reinforcement Learning, a framework that can be used to generate a *policy* directly mapping an observation (oftentimes a partial one) of the state of the system to a control action.

Deep Reinforcement Learning (DRL) offers a significant advantage over traditional model-based approaches for the guidance and control of reusable launchers by effectively managing complex and nonlinear dynamics without the need for predefined mathematical models. DRL's inherent adaptability and robustness enable controllers to adjust seamlessly to varying conditions and unforeseen disturbances, thereby enhancing system reliability and performance. Leveraging extensive data and simulation experiences, DRL can optimize multiple performance metrics simultaneously, such as fuel efficiency and trajectory accuracy, while efficiently handling high-dimensional state spaces. Additionally, DRL frameworks provide scalability and transferability across different launchers and mission profiles, reducing development time and costs. The successful application of DRL in various aerospace contexts, coupled with its potential for continuous improvement and seamless integration with existing systems, underscores its suitability for modern guidance and control challenges. By addressing key challenges related to safety, explainability, and computational resources, DRL-based controllers can meet the stringent requirements of reusable launcher systems, thereby strengthening the development and deployment of advanced reinforcement learning solutions in aerospace engineering.

In recent years, DRL is increasingly applied to several G&C problems both within and outside the aerospace sector [9]: in [10] the authors use a Model Predictive Control to follow trajectories generated through an RL policy, to effectively maneuver around obstacles in an environment. In particular, the field of UAVs has been greatly influenced by RL [11] to obtain the path planner subsystem of the GNC stack for high-level goals and lower-level local planning. In [12], a hierarchical structure is investigated for missile evasion and guidance: multiple low-level policies are developed and a policy selector agent selects in real-time the optimal one. In [13] a closed-loop controller learned through an RL approach is able to both directly guide a low-thrust spacecraft or augment a traditional guidance approach, despite large initial deviations from the target and in the presence of perturbations and significant non-linearities in the dynamics, while preserving real-time capabilities on limited computational resources. In [14] the task of in-orbit rendezvous and docking is tackled: a learned guidance strategy feeds velocity commands to a conventional controller to track, to lower the learning burden and facilitate sim-to-real transfer. The RL framework can be used to optimize trajectories for complex metrics, such as in [15,16] where an RL policy is used for path planning to maximize the imaged surface of an in-orbit uncooperative target.

For planetary landing RL enables streamlined guidance and control directly from sensor data; in [17,18], lunar rendezvous and landing using image-based guidance is performed using a reinforcement meta-learning algorithm that derives a *sensor-to-action policy*. Moreover, recent studies have used this technique on extraterrestrial planetary bodies and with different thruster configurations than the ones typically employed in launchers first stages [19–21]. Furthermore, there was limited dispersion in the initial mass of the lander and no analysis of the robustness of the developed policy to unmodeled dynamics and parametric uncertainties.

Contributions of the work. Starting from this background, in this paper, we focus on examining the atmospheric planetary landing of rocket first stages using Reinforcement Learning (RL), an area that has received limited attention in current state-of-the-art research. To address this, our primary contributions are focused on:

- Applying model-free reinforcement learning to consistently achieve successful atmospheric landing by directly controlling the launcher's gimbaled thruster.

- Developing an open-source, validated 6-DOF simulation environment, expandable with more actuators and including the relevant dynamics for atmospheric planetary landing.
- Investigating different reward functions for the task, including a novel approach with a two-phase training run using ablated reward functions.

In particular, the landing problem has been approached through subsequently more general cases, to gain a deeper understanding of the interaction between the change in the reward function, the choice of hyperparameters and the convergence of the algorithm. We first addressed a 3-DOF case with simplified initial conditions (lower initial velocity and height), then secondly moved to the 6-DOF environment. The 6-DOF simulator was validated using a Simulink model already validated with a Flight Environment Simulator. After satisfactory results were obtained, we investigated more realistic initial conditions taken from the flight profile of a Falcon 9, first addressing the problem in 3-DOF and then the more complex one in 6-DOF. The whole training phase was greatly facilitated by the software pipeline developed, which easily allowed training on Cloud Virtual Machines and monitoring the training metrics through an interactive web interface. In summary, after outlining the problem dynamics in Section 2, the environment implementation is detailed in Section 3. The results for the 3-DOF and 6-DOF cases are presented in 4 and 5 respectively. Finally, a robustness analysis to unmodeled dynamics and disturbances is briefly presented in 6.

2. Problem dynamics

The rocket is modeled as a rigid body, with aerodynamic effects and a uniform gravitational field, a reasonable approximation due to proximity to Earth's surface. The position of the CoM and CoP are considered constant and the inertia matrix is taken to be diagonal. The change in thrust due to atmospheric pressure variation is neglected, exploiting the ISA atmospheric density profile as defined in [22]. Only the aerodynamic force due to drag is computed, with no lift considered.

2.1. 3 degrees of freedom

The system is first studied on a *planar* 3-DOF model of the rocket. The dynamics equations are defined in an inertial reference frame with the x axis directed upwards, as reported in equations Eq (1), with the following simplifying assumptions:

- Only the axial aerodynamic force A is considered (no *normal force*);
- The inertia J is computed using an average value for the mass;
- Gravity g is uniform;
- The density ρ is constant and taken to be at sea level value.

$$\begin{cases} m\ddot{x} = T\sin(\theta+\delta) - A\sin\theta - mg \\ m\ddot{y} = T\cos(\theta+\delta) - A\cos\theta \\ J\dot{\omega} = -T\sin\delta(x_T - x_{\text{CoM}}) \\ \dot{m} = -\frac{T}{g_0 I_s} \end{cases} \tag{1}$$

The control authority is provided by the thrust T, gimbaled of an angle δ from the longitudinal direction of the rocket. The attitude is parametrized by the angle θ with angular velocity ω. The positions of the CoM and CoP along the longitudinal body axis are respectively x_{CoM} and x_{CoP}.

2.2. 6 degrees of freedom

As introduced before, the training analysis will also involve the study of a more complex 6-DOF scenario. The formulation adopted for this more realistic condition is presented in the following subsections. Two reference frames are employed to describe the overall dynamics of the system:

- **Inertial reference frame** $\mathcal{F}_I$: the origin is in the landing site and the x-axis pointing upward.
- **Body-fixed reference frame** $\mathcal{F}_B$: the reference frame is fixed with the vehicle's body, centered at its center of mass (CoM) and aligned with its x-axis along the longitudinal axis of the vehicle, pointing towards the opposite end with respect to the thruster. The other two axes are perpendicular to the x-axis.

2.2.1. Translational dynamics

The translational dynamics are defined in the inertial reference frame $\mathcal{F}_I$ by the following system of differential equations:

$$\begin{cases} \dot{\vec{r}}_I &= \vec{v}_I \\ \dot{\vec{v}}_I &= \frac{1}{m(t)}\vec{F}_I + \vec{g}_I \\ \dot{m} &= -\frac{\|\vec{T}_I\|}{I_{sp} g_0} \end{cases} \tag{2}$$

where:

- $\vec{g}_I = [-g_0, 0, 0]^T$ is the gravitational acceleration vector;
- $\vec{F}_I$ is the summation of the forces acting on the vehicle, as detailed in Section 2.3.

2.2.2. Rotational dynamics

The rotational dynamics is governed by Euler's rigid body equations and the kinematics are parametrized using the quaternion representation in the scalar-first convention (See q (34) in the Appendix).

The dynamics are expressed in the *body-fixed reference frame* and the quaternion parametrization thus represents the attitude of $\mathcal{F}_B$ with respect to $\mathcal{F}_I$. The kinematics and dynamics equations are reported in Eq. (3).

$$\begin{cases} \dot{\vec{q}} &= \frac{1}{2}\Omega\vec{q} \\ \dot{\vec{\omega}}_B &= J^{-1}(\vec{M}_B - \vec{\omega}_B \times J\vec{\omega}_B) \end{cases} \tag{3}$$

In these equations, the $\vec{M}_B$, J and Ω terms are defined as:

- $\vec{M}_B$ is the summation of the moments acting on the rocket expressed in the *body frame*;
- J is the inertia matrix of the rocket (computed at the beginning of each episode depending on the initial mass m and maintained constant):

$$J = \begin{bmatrix} \frac{1}{2}mr^2 & 0 & 0 \\ 0 & \frac{1}{12}m(l^2+3r^2) & 0 \\ 0 & 0 & \frac{1}{12}m(l^2+3r^2) \end{bmatrix} \tag{4}$$

- Ω is the skew-symmetric matrix :

$$\Omega = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{5}$$

2.3. Forces and moments

The forces considered to be acting on the rocket are the control force $\vec{T}$ and the aerodynamic force $\vec{A}$.

Table 1
Mean and range of simplified initial conditions for each state.

	$\vec{r}$	θ	$\vec{v}$	ω	m
$\vec{\mu}_{\vec{x}_0}$	[50, 500] m	π rad	[0, −50] m/s	0 rad/s	41e3 kg
$\vec{\Delta}_{\vec{x}_0}$	[5, 50] m	0 rad	[0, 0] m/s	0 rad/s	1000 kg

Table 2
Mean and range of simplified initial conditions.

	$\vec{r}$ [m]	$\vec{v}$ [m/s]	$\vec{q}$ [–]	$\vec{\omega}$ [rad/s]	m [kg]
$\vec{\mu}_{\vec{x}_0}$	[500, 100, 100]	[−50, 0, 0]	[1, 0, 0, 0]	[0, 0, 0]	41e3
$\vec{\Delta}_{\vec{x}_0}$	[50, 10, 10]	[10, 10, 10]	[0.1, 0.1, 0.1, 0.1]	[0.1, 0.1, 0.1]	1e3

2.3.1. Control force

The rocket engine provides the necessary control force, gimbaling about the z_B body axis of an angle δ_Y and about the y_B body axis of an angle δ_Z. The rotation matrix $R_{T\to B}$ rotates the thrust vector $\vec{T}_T = [T, 0, 0]^T$ from the thrust frame (fixed with the nozzle axis) to the body frame:

$$R_{T\to B} = \begin{bmatrix} cos(\delta_y)cos(\delta_z) & -sin(\delta_y) & -cos(\delta_y)sin(\delta_z) \\ sin(\delta_y)cos(\delta_z) & cos(\delta_y) & -sin(\delta_y)sin(\delta_z) \\ sin(\delta_z) & 0 & cos(\delta_z) \end{bmatrix} \tag{6}$$

Thus, the thrust vector in the body frame $\mathcal{F}_B$ can be computed as:

$$\vec{T}_B = R_{T\to B}\vec{T}_T \tag{7}$$

The engine is modeled after the Merlin 1D of the Falcon 9 [23], having saturations both in the gimbaling range and in the thrust magnitude.

2.3.2. Aerodynamic force

The aerodynamic force is computed as:

$$\vec{A}_B = -\frac{1}{2}\rho(x)\|\vec{v}_I\|S_A C_A R_{I\to B}\vec{v}_I \tag{8}$$

The aerodynamic coefficients matrix C_A is a diagonal matrix with coefficients for each body axis:

$$C_A = \begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & c_z \end{bmatrix} \tag{9}$$

where the aerodynamic coefficients are chosen to be equal, reflecting the so-called *spherical aerodynamic model*, in which the only aerodynamic force is the drag force [7]. The atmosphere is modeled with decaying density, which is computed according to the International Standard Atmosphere (ISA) [22].

2.4. Initial conditions

Two training sets of initial conditions have been used during training: first, a set of simplified initial conditions (lower height and velocity than typical landing trajectories, velocity directed only downwards) is employed as a stepping stone to shape the reward function and then a second set of initial conditions sourced from historic flight data of the Falcon 9 [24] is used to simulate a realistic landing on a downrange location, such as a barge in the middle of the ocean or a downrange landing pad. Both sets are centred around a mean vector $\vec{\mu}_{\vec{x}_0}$ and are sampled from a uniform distribution within a specified range $\vec{\Delta}_{\vec{x}_0}$.

Simplified initial conditions

For the simplified initial conditions case the mean and range are reported in Table 1 in the case of the 3-DOF environment and Table 2 for the 6-DOF case.

Table 3
Mean and range of initial conditions.

	$\vec{r}$	θ	$\vec{v}$	ω	m
$\vec{\mu}_{\vec{x}_0}$	[−1600, 2000] m	$\frac{3}{4}\pi$	[180, −90] m/s	0	41e3 kg
$\vec{\Delta}_{\vec{x}_0}$	[200, 10] m	0.1	[30, 30] m/s	0.05	1000 kg

Table 4
Mean and range of realistic initial conditions.

	$\vec{r}$ [m]	$\vec{v}$ [m/s]	$\vec{q}$ [–]	$\vec{\omega}$ [rad/s]	m [kg]
$\vec{\mu}_{\vec{x}_0}$	[2000, −1600, 0]	[−90, 180, 0]	[0.866, 0, 0, −0.5]	[0, 0, 0]	41e3
$\vec{\Delta}_{\vec{x}_0}$	[10, 200, 0]	[30, 30, 0]	[0.1, 0.1, 0.1, 0.1]	[0.05, 0.05, 0.05]	1e3

Realistic initial conditions

The realistic initial conditions are shown in Table 3 in the 3-DOF case and Table 4 for the 6-DOF.

3. Environment implementation

The reinforcement learning algorithm used to solve the optimization problem is the Proximal Policy Optimization (PPO) algorithm [25]. PPO is a policy gradient reinforcement learning method that seeks to directly optimize the policy by following the gradient of the expected return. It improves upon previous policy gradient methods like TRPO by using a clipped surrogate objective function to restrict the change in policy at each update step. Specifically, PPO optimizes the following loss function:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{10}$$

where $r_t(\theta)$ is the probability ratio between the new and old policies, $\hat{A}_t$ is an estimate of the advantage function, and ϵ is a hyperparameter that restricts how much the new policy can change per update. The loss function incentivizes the improvement of the policy while keeping the new policy close to the old one. PPO alternates between sampling data using the old policy and optimizing the loss function using stochastic gradient ascent. Moreover, other advantages are faster training, better sample complexity, and ease of implementation compared to prior policy gradient methods. The dynamics are integrated using a variable-step RK45 ODE integrator, with the relative and absolute tolerances respectively ($\epsilon_{\text{rel}} = 0.001$ and $\epsilon_{\text{abs}} = 1e$–06).

3.1. Observation space

The environment's observation space encompasses all possible values of the observations of the system state. It is a subset of the state space, consisting of the set of all possible system states. In our setting, both spaces are continuous as the variables are real-valued; the observation state is equal to the state vector, defined as follows, without the mass:

$$\vec{x} = \begin{bmatrix} \vec{r} \\ \vec{v} \\ \vec{q} \\ \vec{\omega} \\ m \end{bmatrix} \tag{11}$$

To enhance the performance of RL algorithms, it is common practice to normalize the observation space since some variables have natural bounds while others are theoretically unbounded. A normalization vector is derived based on reasonable maximum values for various state variables based on free-fall time, maximum free-fall velocity and maximum angular velocity achievable by the actuator. In both environments, 3 and 6-DOF, the state returned by the simulator is normalized element-wise by dividing each element by the corresponding value from the normalization vector ensuring that the normalized state vector lies within the desired range. The bounds of the Observation Space are enforced by terminating an episode if any of the state variable values exceed the normalized bounds.

Table 5
Normalization values for the action.

Action vector element	$\|\|\vec{T}_{max}\|\|$	δ_y	δ_z
Selected bound	981 kN	20 deg	20 deg

3.2. Action space

The action space comprises the space of all valid actions for the environment. In particular, the control vector for the 6-DOF rocket is:

$$\vec{u} = \begin{bmatrix} \vec{\delta}_y \\ \vec{\delta}_z \\ \vec{T} \end{bmatrix} \tag{12}$$

where the values are defined as:

- **thrust** $\|\vec{T}\|$, limited to the range $\|\vec{T}\| \in [0, 981]$ kN
- **gimbal angles** δ_y and δ_z, limited in the range $[-20\,\text{deg}, +20\,\text{deg}]$

In general, it is preferred to have each term of the action bound in the interval $[-1, +1]$, normalizing the control vector with the values in Table 5, obtaining the action vector $\vec{a}$.

In this way, the action space then simply becomes:

$$\mathcal{A} = \left\{ \vec{a} \in \mathbb{R}^3 \,:\, a_i \in [-1, +1], i = 1, 2, 3 \right\} \tag{13}$$

3.3. Reward functions

One of the most important elements in the RL paradigm is the reward function $R(\vec{s}, \vec{a})$. This function maps a tuple of current state $\vec{s}$ and action taken $\vec{a}$ to a scalar value r. This scalar value is used to map each state's value and make the policy π converge to the optimal one π^*. One of the biggest issues in reinforcement learning is choosing a reward function that is reasonably descriptive for the problem at hand, meaning that it is not too *sparse*, otherwise it could be hard for the algorithm to map correctly the value of each state (this is called *credit assignment problem*).

Indeed, if, in the landing problem, the reward is only given upon landing in a state within the correct bounds the algorithm fails to converge to a policy different from a random one. Several iterations of the reward function have been tested to reach a satisfactory result, based on giving the agent a hint of the correct behavior to reach the final goal state and getting a bonus when this goal is achieved while trying to minimize fuel consumption. The reward functions were first developed and tested on the 3-DOF environment and then tuned on the 6-DOF environment to reach satisfactory behavior in terms of convergence and performance of the obtained policy. In the following sections, both versions are detailed.

Reward function development progression. For both simplified and realistic initial conditions, various reward functions were developed and evaluated, building upon those proposed in [20,21]. For the first case, two reward functions are developed, with the second one as an *ablated* version of the first. Both reward functions incentivize the agent to follow a target velocity aiming for the landing site and give a final bonus in case of a successful landing. When the landing is consistently achieved the second reward function is used to optimize fuel consumption; this *ablated reward function* does not provide the agent with a target velocity, exploiting the fact that the value function has already mapped the landing states to the high reward. In this way, the first training run allows the agent to learn how to land successfully and, the second, to minimize fuel consumption.

In the case of realistic initial conditions, a reward function based on following the acceleration command resulting from the energy-optimal solution of a simplified optimal control problem is used. A recap of the steps is shown in Fig. 1

3.3.1. Target velocity reward

The first reward function that has been tested is defined in Eq (14), starting from [21]; it follows a Line-of-Sight heuristic to have at each time step a target velocity vector to hint the rocket where to go. Therefore the reward strongly incentivizes a successful landing by giving a final bonus when the landing conditions are satisfied and a penalty on thruster usage to minimize fuel consumption.

$$\begin{aligned} r =& \alpha \|\vec{v} - \vec{v}_{targ}\| + \beta \|\vec{T}_B\| + \gamma \cdot any([|\psi|, |\theta|, |\phi|] > [\psi, \theta, \phi]_{lim}) + \eta \\ &+ \kappa (x \le 0 \,\text{and}\, \|\vec{r}_f\| < r_{lim} \,\text{and}\, \|\vec{v}_f\| < v_{lim} \,\text{and}\, all(\vec{\omega} < \omega_{lim}) \\ &\text{and}\, all([\psi, \theta, \phi]_f < [\psi, \theta, \phi]_{lim_f})) \end{aligned} \tag{14}$$

Each term of the reward function is weighted and has a specific meaning:

- α: rewards having a velocity vector $\vec{v}$ close to the target velocity $\vec{v}_{targ}$;
- β: penalizes usage of the thrust, to reduce propellant consumption;
- γ: penalizes exceeding the attitude limits, represented by the $[\psi, \theta, \phi]_{\text{lim}}$ limit euler angles. These are not strict constraints and their purpose is to avoid the agent exploring excessively states which are unlikely to be visited in the optimal solution. In the 3-DOF case there is an additional penalty term for exceeding a θ_{mgn} angle, that guides the agent towards reasonable attitudes during training.
- η: this is a small positive constant to avoid early termination by the agent (as all other rewards except for the terminal one are negative);
- δ: this term, present only in the 3-DOF case, gives the agent a hint that it is approaching a limit in the attitude, defined by the θ_{mgn} angle;
- κ: multiplies the phterminal reward term, given to the agent only if the final landing conditions are satisfied.

The 3-DOF reward function has an additional attitude bounds term $-\delta \cdot max(0, |\theta| - \theta_{mgn})$. The target velocity magnitude follows a decaying exponential shape as shown in Eq. (15) starting from the initial velocity $v_0 = \|\vec{v}(t_0)\|$, based on the time-to-go computed in (16). This target velocity targets a waypoint at $z^{\text{waypoint}} = 50$ m and is made to be only vertical when below this height thanks to the shape of $\hat{\vec{r}}$ as in Eq. (17) and of $\hat{\vec{v}}$ as in Eq. (18). The decay speed is controlled by the scaling time τ which also changes in value at the waypoint, as in (19).

The weights have been carefully tuned through subsequent runs of the algorithm starting from the values in [21], aiming to have each term with the same order of magnitude, except for the terminal bonus coefficient k, and are detailed in Table 6. Instead, the values of the attitude and final landing state parameters are shown in Table 7.

$$\vec{v}_{targ} = -v_0 \left(\frac{\vec{r}}{\|\hat{\vec{r}}\|} \right) \left(1 - exp\left(-\frac{t_{go}}{\tau} \right) \right) \tag{15}$$

$$t_{go} = \frac{\hat{\vec{r}}}{\hat{\vec{v}}} \tag{16}$$

$$\hat{\vec{r}} = \begin{cases} \vec{r} - \left[0\ 0\ z^{\text{waypoint}}\right], & \text{if } r_z > z^{\text{waypoint}} \\ \left[0\ 0\ r_z\right], & \text{otherwise} \end{cases} \tag{17}$$

$$\hat{\vec{v}} = \begin{cases} \vec{v} - [0\ 0\ 2], & \text{if } r_z > z^{\text{waypoint}} \\ \vec{v} - [0\ 0\ 1], & \text{otherwise} \end{cases} \tag{18}$$

$$\tau = \begin{cases} \tau_1, & \text{if } r_z > z^{\text{waypoint}} \\ \tau_2 & \text{otherwise} \end{cases} \tag{19}$$

3.3.2. Ablated reward function

To improve fuel efficiency, a second phase of the training run discards the target velocity-tracking term $\alpha\|\vec{v} - \vec{v}_{targ}\|$ (see Table 8).

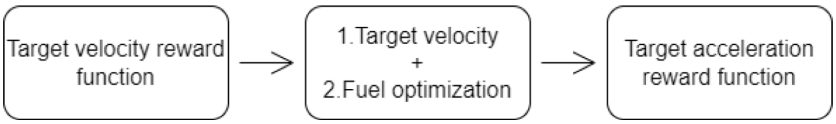
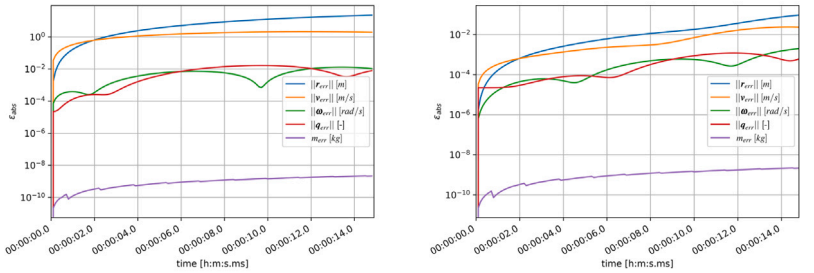


Fig. 1. Steps in the development of reward functions: starting from targeting a *heuristic* velocity aiming for the landing site, we move to a two phases training to optimize more the fuel consumption. Finally, we move to a different reward function as the environment becomes more challenging.



(a) ISA atmospheric model (b) Exponential atmospheric model

Fig. 2. Absolute errors between validated and developed simulator with different atmospheric models.

Table 6
Target velocity reward function coefficients.

Coefficient	α	β	γ	δ	κ	η	τ_1	τ_2
Value	−0.01	−1e−7	−10	−5	10	0.05	20 s	100 s

Table 7
Target velocity reward function parameters.

Coefficient	$\lvert\theta_{lim}\rvert$	θ_{mgn}	ω_{lim}	r_{lim}	v_{lim}	$[\psi,\theta,\phi]_{\lim}$	$[\psi,\theta,\phi]_{\lim_f}$
Value	360°	180°	0.2 rad/s	30 m	10 m/s	[85, 85, 360]°	[10, 10, 360]°

Table 8
Ablated reward function coefficients.

Coefficient	ξ	γ	κ	δ	η
Value	0.004	−10	10	−5	0.05

3.3.3. Target acceleration reward

The target velocity reward function works well in the simplified initial conditions case but does not reach a successful landing in the case of realistic initial conditions, with an excessive vertical component of the velocity vector and a tilt angle at landing too high. A different reward based on a target acceleration has been developed to reach a correct landing with these initial conditions. The reward function gives a hint to the agent to follow a target acceleration $\vec{a_{targ}}$, which is detailed in the following paragraphs.

Eq. (20) defined the reward function defined for the 3-DOF case instead, Eq. (21) the one for the 6-DOF case:

$$\begin{aligned} r &= \alpha\|\vec{a}-\vec{a}_{targ}\|\Delta_h + \beta\|\vec{T}\| + \eta \\ &+ \gamma(|\theta| > \theta_{lim}) - \delta\cdot max(0, |\theta| - \theta_{mgn}) + \eta \\ &+ w_f \cdot max\left(1-\frac{r_f}{r_f^{\max}},0\right)\cdot max\left(1-\frac{v_f}{v_f^{\max}},0\right) \\ &+ \kappa(x \le 0\, and\, \|\vec{r}\| < r_{lim}\, and\, \|\vec{v}\| < v_{lim}\, and\, and\, |\omega| < \omega_{lim}\, and\, |\theta| < \theta_{lim_f}) \end{aligned} \tag{20}$$

Table 9
Target acceleration reward function coefficients.

Coefficient	α	β	γ	δ	η	w_f	$v_f^{\max}$	$v_f^{\max}$	κ
Value	−0.01	−1e−8	−1	−5	0.1	50	50 m/s	100 m/s	10

$$\begin{aligned} r &= \alpha\|\vec{a}-\vec{a}_{targ}\|\Delta_h + \beta\|\vec{T}\| + \eta \\ &+ \gamma any([|\psi|,|\theta|,|\phi|] > [\psi,\theta,\phi]_{lim}) \\ &+ w_f \cdot max\left(1-\frac{r_f}{r_f^{\max}},0\right)\cdot max\left(1-\frac{v_f}{v_f^{\max}},0\right) \\ &+ \kappa(x \le 0\, and\, \|\vec{r}\| < r_{lim}\, and\, \|\vec{v}\| < v_{lim}\, all(\vec{\omega} < \omega_{lim}) \text{and}\, all(\vec{q} < \vec{q}_{\lim})) \end{aligned} \tag{21}$$

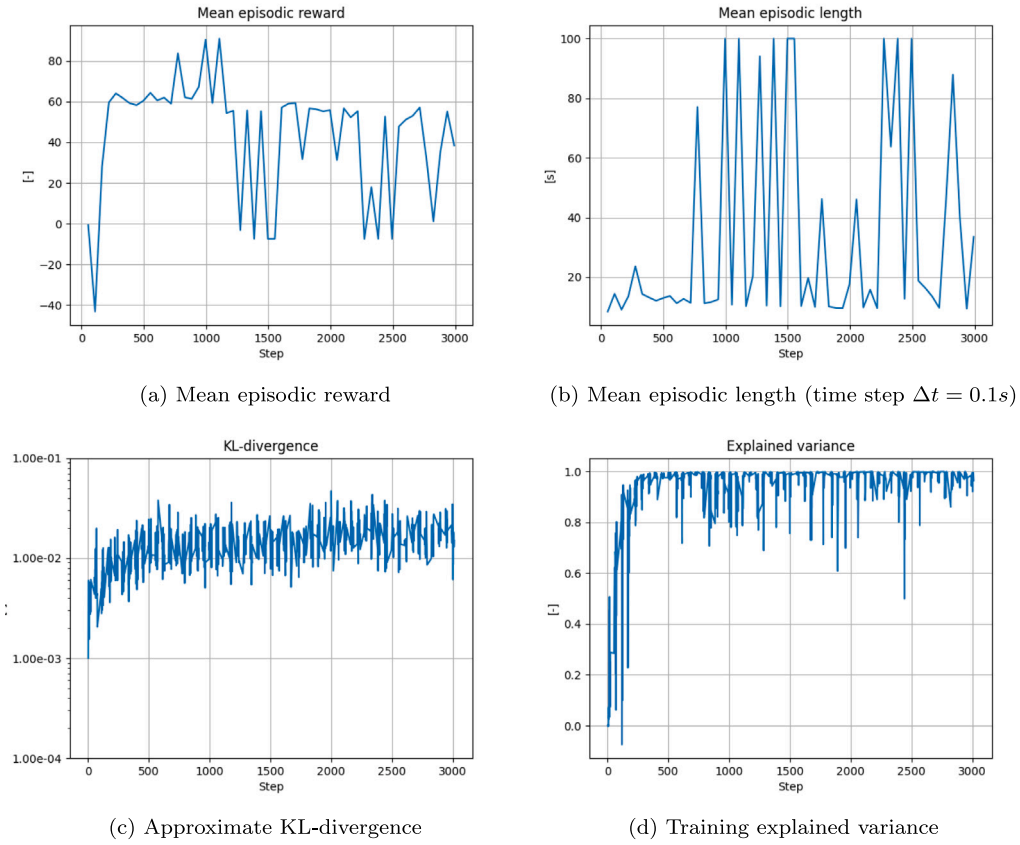
This approach tries again to achieve a balance between giving the agent a hint to achieve a successful landing and minimizing fuel consumption. In this case, θ_{mgn} is a *management* angle, that guides the agent towards reasonable attitude bounds in the 3-DOF case The reward coefficients are detailed in Table 9.

In this case, two further terms can be considered to improve convergence giving a linearly decaying reward as the final velocity and position errors increase within a certain interval. They are multiplied together and scaled by the weight w_f. Other kinds of functions have been tested to shape the terminal reward, such as quadratic and exponential functions but the linear ones resulted in the smallest landing errors. The target acceleration reward term is given down to a certain waypoint height (set to $x = 50$ m) and then set to 0 to let the agent explore more, in fact, this can be modeled as being multiplied by a step function of the height Δ_h:

$$\Delta_h = \begin{cases} 1 & \text{if } x \ge waypoint \\ 0 & \text{if } x < waypoint \end{cases} \tag{22}$$

The target acceleration a_{targ} is the solution of the simplified problem which minimizes the energy performance index in Eq (23).

$$J = \frac{1}{2}\int_{t_0}^{t_f} \vec{a}^T\vec{a}\, dt \tag{23}$$



(a) Mean episodic reward

(b) Mean episodic length (time step $\Delta t = 0.1s$)

(c) Approximate KL-divergence

(d) Training explained variance

Fig. 3. Trend of RL training metrics through training (simplified initial conditions), showing convergence to a policy maximizing mean reward and minimizing episodic length.

subject to the following dynamics:

$$\begin{cases} \dot{\vec{r}} &= \vec{v} \\ \dot{\vec{v}} &= \vec{g} + \vec{a} \\ \vec{a} &= \dfrac{\vec{T}_I}{m} \end{cases} \tag{24}$$

In this paper, both terminal position and velocity are null, $\vec{r}_f = \vec{0}$ and $\vec{v}_f = \vec{0}$ thus resulting in the following analytical solution:

$$\mathbf{a}_{targ} = -\frac{6\mathbf{r}}{t_{go}^2} - \frac{4\mathbf{v}}{t_{go}} - \mathbf{g} \tag{25}$$

with the time-to-go t_{go} computed as the real positive solution of the quartic equation:

$$g^2 t_{go}^4 - 4\|\mathbf{v}\|^2 t_{go}^2 - 24\mathbf{r}^T\mathbf{v}t_{go} - 36\|\mathbf{r}\|^2 = 0 \tag{26}$$

This problem does not account for the presence of the atmosphere and is not fuel-optimal; however, it is used as a proxy to achieve a successful landing by computing the optimal acceleration at each time step and using it as a target to balance the other terms of the reward function. To avoid an excessive acceleration command for the engine the actual target acceleration is clipped as:

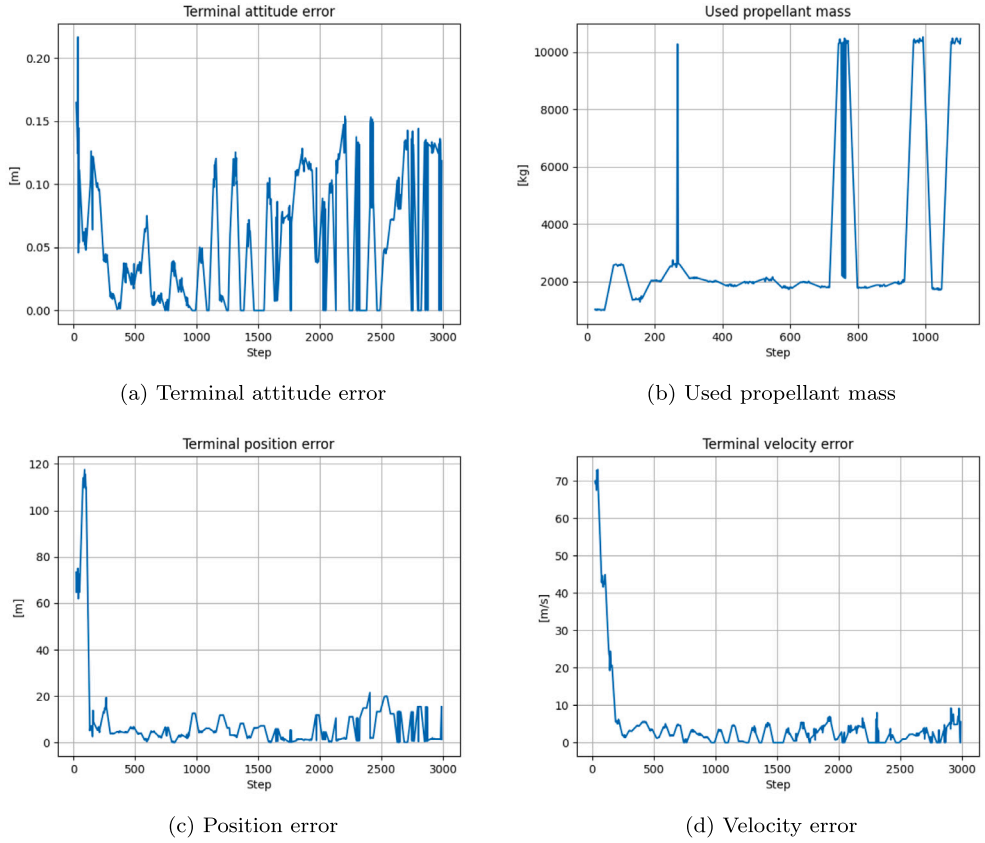
$$\vec{a}_{targ} = sat_{T_{max}/m}(\vec{a}_{targ}) = \begin{cases} \vec{a}_{targ} & \text{if } \|\vec{a}\| \leq U \\ \vec{a}_{targ}\dfrac{T_{max}/m}{\|\vec{a}_{targ}\|} & \text{if } \|\vec{a}_{targ} > T_{max}/m\| \end{cases} \tag{27}$$

A detailed solution to the problem can be found in [26]. This target acceleration has been exploited also in [20], albeit with a different action space and a different learning algorithm.

3.4. Environment validation

The environment has been validated using as baseline the one developed in [8] which had in turn been validated using the FES (Flight Environment Simulator) internal software from Deimos Space. The validation was performed by simulating several trajectories with different control actions applied and checking the error between the two simulators. The samples shown in 2 input a constant actuator command $\vec{u} = [5°, 5°, 98000N]^T$ and are shown to highlight discrepancies in a limit scenario in terms of trajectory (high rotational velocities reached). Other trajectories were tested to validate the behavior on more realistic test cases (low angular velocities) and the errors are several orders of magnitude lower than the (already small) ones reported here. Overall the simulator manages to capture faithfully the relevant behavior of the launcher.

To validate the simulator the same atmospheric model used in [8] is first employed to correctly assess the magnitude of the errors. The errors, shown in Fig. 2(b), are quite tiny. Through a deeper analysis, it has been assessed that the difference stems mainly from differences in the normalization of the quaternion. This is reflected in a *slight* discrepancy in the rotational behavior which is however emphasized in this validation run due to the extremely high angular velocity reached, a behavior which would result in premature termination of the episode during training.



(a) Terminal attitude error

(b) Used propellant mass

(c) Position error

(d) Velocity error

Fig. 4. Trajectory metrics during training (simplified initial conditions). The position and velocity error steeply go down and the used mass trends downwards when outliers are not considered.

The simulator employed for the training phase uses a more appropriate atmospheric model (ISA atmosphere). As shown in Fig. 2(b) the atmospheric model does not have a significant effect on the behavior of the launcher and the error from the validated simulator is still low.

4. Results 3-DOF environment

In this chapter the results of the 3-DOF case are shown, quantifying both the performance of the RL algorithm and the performance of the obtained trajectory in the case of simplified and realistic initial conditions. In this setting the policy can achieve quick convergence with both sets of initial conditions, indicating that the dense reward is sufficient to provide a strong learning signal to the algorithm. This suggests that expanding the analysis of this scenario to a full reentry trajectory would be feasible.

4.1. Simplified initial conditions

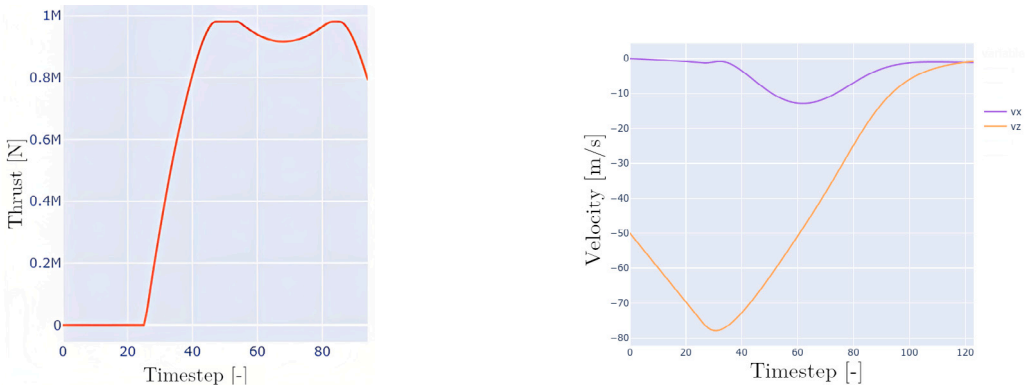
For the simplified initial conditions a lower initial height and initial velocity are selected. Furthermore, the velocity is *nominally* in the vertical direction only. This makes the credit-assignment problem easier due to the reduced episodic length and does not require the policy to shed any initial horizontal velocity. The mean and range of the distribution are reported in Table 1.

In Fig. 3 the convergence behavior of the algorithm is highlighted. Due to the multi-phase training process, there are discontinuities in the mean reward and episodic lengths at the $1k$ Steps mark (on the x axis the *step* refers to *evaluation* step. The exploratory behavior of the algorithm results in spikes due to deviation from the locally optimal policies, which are crucial to explore the state and action spaces and converge towards a *global* optimum. Moreover, the convergence can also be analyzed by looking at the end-of-episode metrics (i.e. the terminal errors and used propellant mass) in Fig. 4 showing quick convergence towards low terminal errors in both position (Fig. 4(c)), velocity (Fig. 4(d)) and attitude (Fig. 4(a)), with a gradual optimization of the used propellant mass as outlined in Fig. 4(b).

It is possible to see that during the training phase, the agent first learns to achieve a correct landing location (Fig. 4(c)), with the episode length (Fig. 3(b)) not changing significantly (meaning that the rocket is still mostly free falling). Successively the terminal velocity reward term incentivizes the agent to learn to slow down, decreasing the velocity error 4(d) and thus increasing the episodic length but also significantly increasing the mean reward 3(a). Following these two steps, a longer-used mass minimization trend can be observed in Fig. 4(b). Moreover, it is interesting to analyze the profiles of velocity, in Fig. 5(b), and thrust, in Fig. 5(a), of a sample episode after convergence. The two figures show that the agent tries to maximize the reward by using the thrusters at a minimum level, thus gaining speed, and then performing a high-thrust final burn. This reflects the ideal thrust profile, as the launcher avoids having high gravity losses during the landing burn by employing the atmospheric drag to do part of the work required along the trajectory.

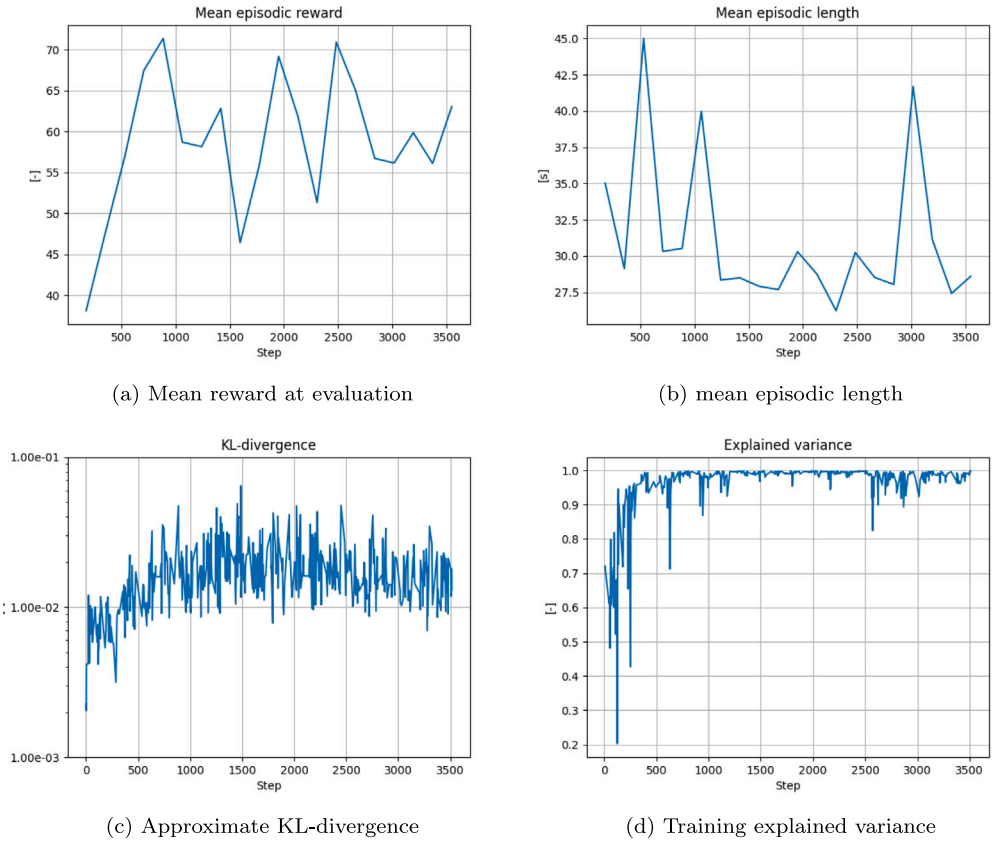
4.2. Realistic initial conditions

Concerning the realistic initial conditions case, the analogous results are depicted in Fig. 6, showing the training metrics measuring the overall performance of the algorithm. In this simulation, the reward is evaluated periodically from a batch of trajectory runs during training which outlines a rapid convergence to a reward value close to the theoretical maximum. This theoretical maximum can be approximated by envisioning an ideal scenario where the target acceleration is perfectly



(a) Thrust profile of a sample episode (b) Speed profile of a sample episode

Fig. 5. Thrust and velocity profiles of the converged policy, showing the agent using a bang–bang profile to minimize gravity losses during the final burn.



(a) Mean reward at evaluation (b) mean episodic length

(c) Approximate KL-divergence (d) Training explained variance

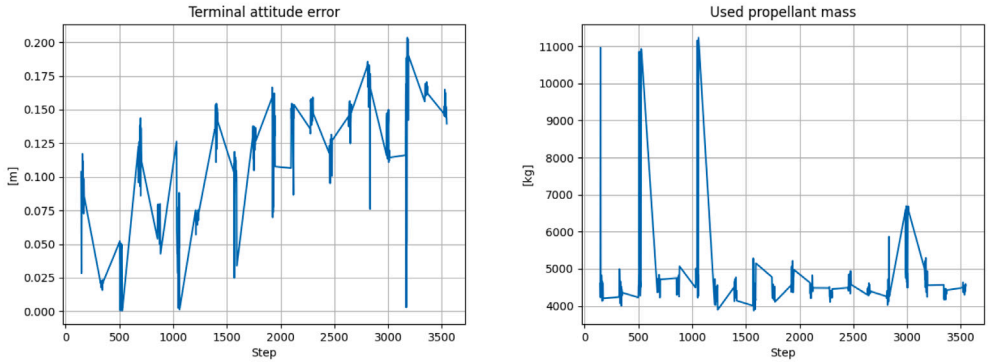
Fig. 6. Trend of RL training metrics through training (realistic initial conditions). The mean reward trends upwards with episodic length going down to minimize propellant consumption.

tracked and the thruster remains unused. Consequently, the policy receives solely the terminal bonuses.

Moreover, by looking at the terminal errors and the used mass combined with the reward trend, we can see that the algorithm manages to converge to a robust policy, with low terminal velocity Fig. 7(d) and position errors Fig. 7(c). The attitude error at touchdown is also acceptable (Fig. 7(a)).

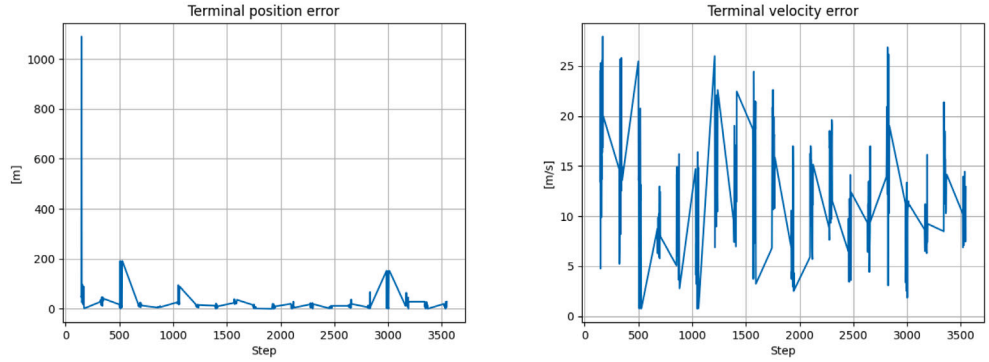
4.3. Comparison of activation functions

Before concluding the discussion on the simulation environment based on the 3-DOF, it may be interesting to analyze the two types of activation functions which have been tested within this environment framework: the Rectified Linear Unit, *ReLU*, and the Hyperbolic Tangent, *tanh*. As shown in Fig. 8(a), using ReLUs results in quicker convergence than using the hyperbolic tangent, however, there is a policy unlearning behavior with the reward peaking after a certain number of training steps. This effect could be overcome by sampling the mean reward obtained by saving the highest-reward policy network, through periodic evaluation of the policy. If monotonic behavior in training is desired for the ReLU activation function, it can be (roughly) obtained by limiting the KL-divergence, as shown in Fig. 8(b). This aims to limit the change in policy parameters at each update, by using early stopping of the neural network optimizer. A good value for the limit



(a) Terminal attitude error

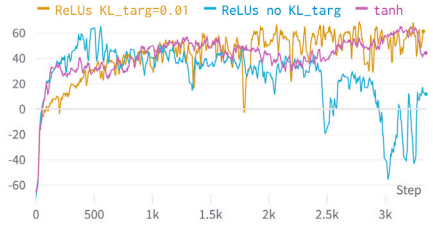
(b) Used propellant mass



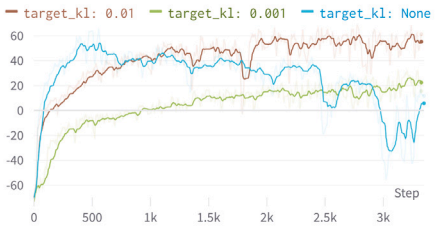
(c) Terminal position error

(d) Terminal velocity error

Fig. 7. Trajectory metrics at each evaluation step (realistic initial conditions).



(a) Comparison of different activation functions showing the higher convergence speed of ReLU but also the need to limit the policy updates (through the KL-divergence target) to avoid policy-unlearning.



(b) Effects of constraining the KL-divergence for ReLUs activation functions: if left unconstrained the policy suffers an un-learning behavior after reaching a maximum in the reward.

Fig. 8. Mean reward through a training trajectory.

is $KL_{\text{targ}} = 0.01$, which does not slow down excessively the learning process but still allows it to reach the maximum episodic reward.

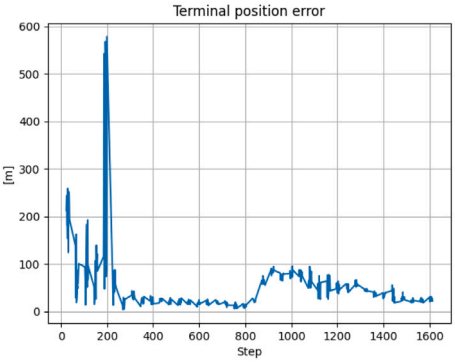
For the 6-DOF environment, it was found that the instability in convergence prevented successfully finding good policies.

5. Results 6-DOF environment

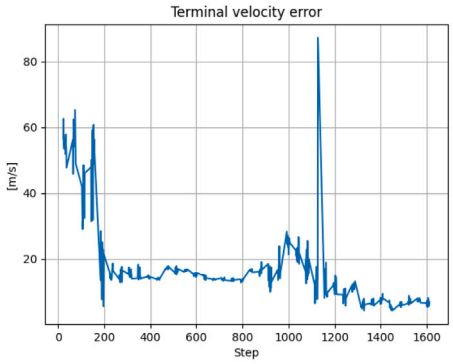
In this section, the analysis of the RL policy for a 6-DOF environment is presented. As happened in Section 4 two training runs are shown: the first for a set of simplified initial conditions, the second for a set of initial conditions sourced from historic flight data of the Falcon 9.

5.1. Simplified initial conditions

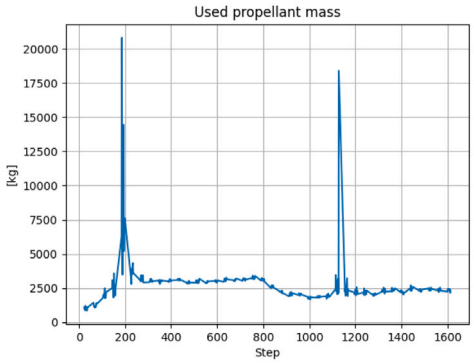
In this instance, the *policy entropy coefficient* is set to a value of $c_{\text{entropy}} = 0.01$. This setting serves the purpose of fostering exploration and mitigating premature fixation of the policy onto a local optimum. The implications of the two phases approach are evident when examining Fig. 9. Initially, the algorithm attains policy convergence during the initial training phase, and then a turning point emerges around time step 800, marked by a modification in the reward function. This alteration initiates a subsequent exploratory phase, which, in turn, leads to convergence toward a more efficient policy, characterized by reduced propellant consumption and a decreased terminal velocity. The lower consumption of propellant is also highlighted by the sharp reduction in average episodic length, as shown in Fig. 10(b).



(a) $||\vec{r}_f||$



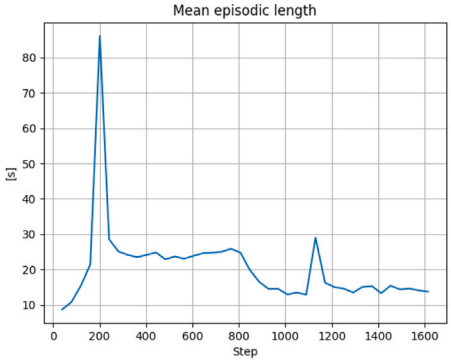
(b) $||\vec{v}_f||$



(c) $m(t_f) - m(t_0)$

Fig. 9. Trend of trajectory metrics during training, showing successful convergence.

(a) Evaluation mean episodic reward



(b) Mean episodic length

Fig. 10. Reinforcement Learning training metrics (simplified initial conditions). The downward trend in episode duration around 10^3 steps shows the strong minimization of fuel consumption due to the two-phase training.

The behavior of some metrics of the PPO algorithm can be assessed to verify convergence from an algorithmic perspective. In particular the mean reward 10(a) has an increasing trend, with a dip at training step 800 due to the switch in reward functions (this implies that there will be a different maximum achievable reward) and the explained variance converging to a value of 1 mean shows a good approximation of the value function by the value network.

5.1.1. Montecarlo analysis of the policy

A Montecarlo analysis is carried out to test the robustness of the policy. Here, since two different reward functions have been used for the training phase, it is interesting to compare the differences between the two training phases. The terminal velocity angle, computed as in Eq. (28), measures the deviation of the terminal velocity from vertical at touchdown, which would nominally be directed downwards to prevent the lander from tipping over.

$$\phi = 180 - \arcsin\left(\frac{v_x(t_f)}{\|\vec{v}(t_f)\|}\right) \tag{28}$$

The statistics for the terminal errors of phase 1 and phase 2 are reported in Tables 10 and 11.

It is quite interesting to analyze the differences between the two controllers: indeed, comparing the position errors, the velocity errors and the used mass, it is clear that the first controller aims to minimize the terminal position error while having a relatively high-velocity error and propellant consumption, while the second controller minimizes

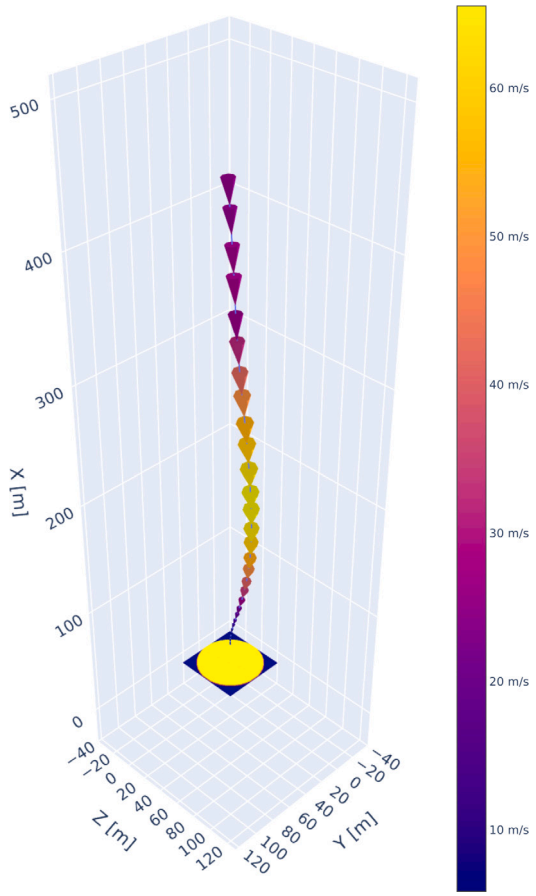


Fig. 11. Trajectory of an episode using the optimal network. The lander decelerates rapidly in the terminal part of the trajectory, to minimize fuel consumption.

Table 10
Terminal errors and used mass statistics, the first phase (mean μ and standard deviation σ).

	Position	Velocity	Attitude	Angular velocity	Used mass	Velocity angle
μ	12.5 m	13.9 m/s	4.3°	0.01°/s	3214 kg	4.6°
σ	4.3 m	0.3 m/s	2.7°	0.01°/s	96 kg	1.4°

Table 11
Terminal errors and used mass statistics, second phase.

	Position	Velocity	Attitude	Angular velocity	Used mass	Velocity angle
μ	25.1 m	6.78 m/s	3.6°	0.03°/s	2467 kg	48°
σ	3.5 m	1.2 m/s	2.0°	0.0°/s	88.4 kg	9.9°

successfully the used propellant mass (using about 30% less propellant), lowering at the same time the average velocity error to about 6.5 m/s, down from about 14 m/s. This is however detrimental to the position error which grows from an average of 14 m to about 26 m. Furthermore, the terminal velocity angle increases significantly, although given the low terminal velocity, this might not affect the stability of the lander. Both attitude and angular velocity errors have comparable statistics across the two training phases and are within acceptable bounds for the landing.

A sample trajectory from the second model is shown in Fig. 11: the lander first rotates to thrust in the landing direction, then when a sufficiently low altitude is reached the thruster performs a burn to reduce the velocity. Finally, once over the landing pad, a vertical attitude is reached and successful landing is achieved.

5.2. Realistic initial conditions

In this case, two important hyperparameters to tweak were the *batch size* and the *number of steps per rollout*; due to the increased length of the average episode and the size of the action space, these had to be increased significantly, in order to have a larger number of samples available, capable of stabilizing the update of the policy and value networks. Therefore, since the algorithm needs to explore larger action and observation spaces, the time to convergence increases, requiring significantly more episodes, clocking in at around 24 h of run-time. In Fig. 12 a more detailed interpretation of the convergence behavior can be observed: comparing the mean episodic reward in Fig. 13(a) with the other metrics, it is evident that the algorithm first explores tracking the target acceleration, then around step 1500, it discovers the terminal bonus, successfully landing with low speed and position errors. Then, there is an exploratory phase in which it attempts to further optimize the reward, which finally converges to a solution with a higher terminal reward due to the decrease in used mass.

The robustness of the PPO algorithm is evident in the trends of its training metrics as shown in Fig. 13. The explained variance, measuring the accuracy of the value function in predicting the cumulative rewards reaches almost 100%, showing how the reward landscape is thoroughly sampled and quantified.

It can be seen that the behavior is noisy and that there are spikes in the terminal errors. This could be caused both by an excessive residual degree of exploration or by the need for the policy to be more robust, and could benefit from further increasing the batch and sample sizes. From the trajectory of an episode with good landing behavior shown in Fig. 14, it can be seen that the rocket manages to reach successfully the landing zone (yellow circle in the figure) with low terminal velocity and a vertical profile for the terminal descent.

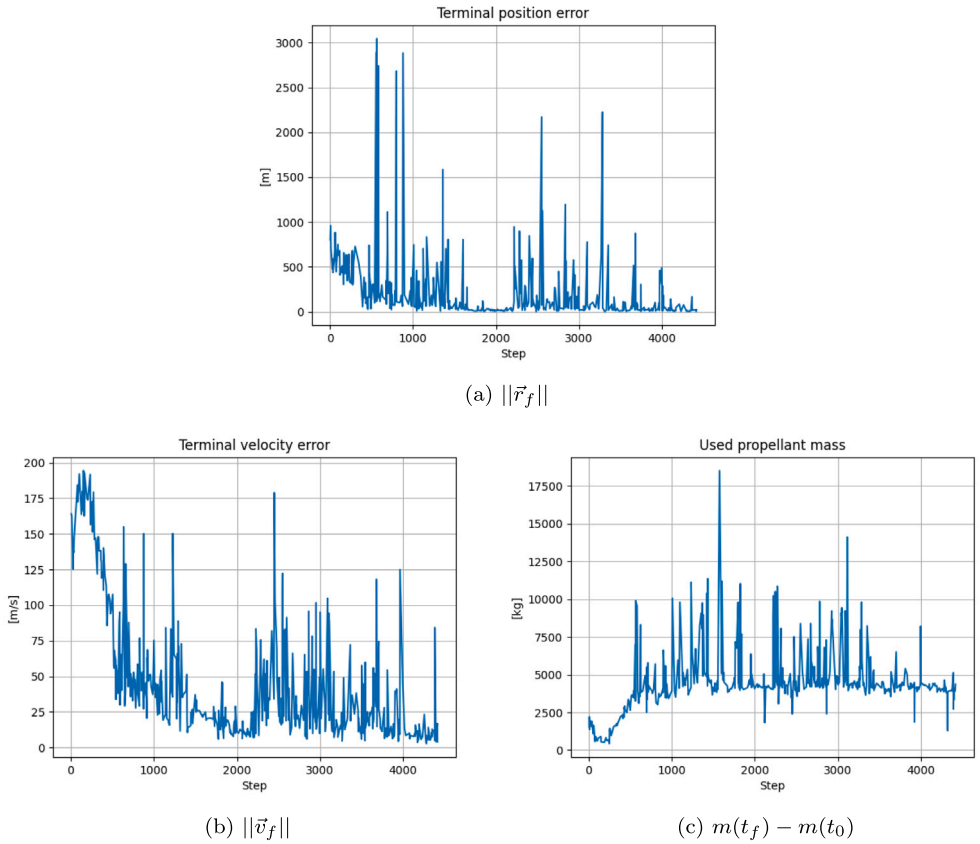


Fig. 12. Trajectory metrics at each evaluation step in the case of realistic initial conditions.

Table 12
Mean and range of initial conditions in the case of Monte Carlo analysis.

	$\vec{r}$ [m]	$\vec{v}$ [m/s]	$\vec{q}$ [–]	$\vec{\omega}$ [rad/s]	m [kg]
$\vec{\mu}_{\vec{x}_0}$	[2000, −1600, 0]	[−90, 180, 0]	[0.866, 0, 0, −0.5]	[0, 0, 0]	41e3
$\vec{\Delta}_{\vec{x}_0}$	[100, 200, 50]	[30, 30, 10]	[0.1, 0.1, 0.1, 0.1]	[0.05, 0.05, 0.05]	1e3

Table 13
Terminal errors and used mass statistics.

	Position	Velocity	Attitude	Angular velocity	Used mass	Final velocity angle
μ	10.0 m	9.42 m/s	4.7°	0.06°/s	4219 kg	18.9°
σ	4.3 m	2.3 m/s	2.7°	0.04°/s	103 kg	9.5°

5.2.1. Montecarlo analysis of the policy

Also in this 6-DOF case, a Montecarlo analysis is carried out on the best-performing policy network to test the robustness of the trained policy. The Montecarlo is performed on mean initial conditions and range of initial conditions as shown in Table 12.

Instead, the statistic for the terminal errors are reported in Table 13.

The performance of the controller is compared to a baseline obtained by solving the fuel-optimal problem disregarding the rotational dynamics of the problem, meaning that the body is treated as a point mass and the constrained (null terminal velocity and distance from landing pad) optimization problem is solved. The RL solution is also compared to a *successive convexification MPC* approach. The results are reported in Table 14 and highlight that the optimal solution is about 25% more efficient than the RL one. This is due to both the acceleration-tracking shaping and the need to perform attitude control as well. The RL controller requires about 60% more of the propellant used by the successive convexification solution, thus being significantly more efficient.

Table 14
Propellant consumption comparison of the obtained policy, the 3-DOF (point mass) optimal solution and 6-DOF successive convexification (SCVX) approaches.

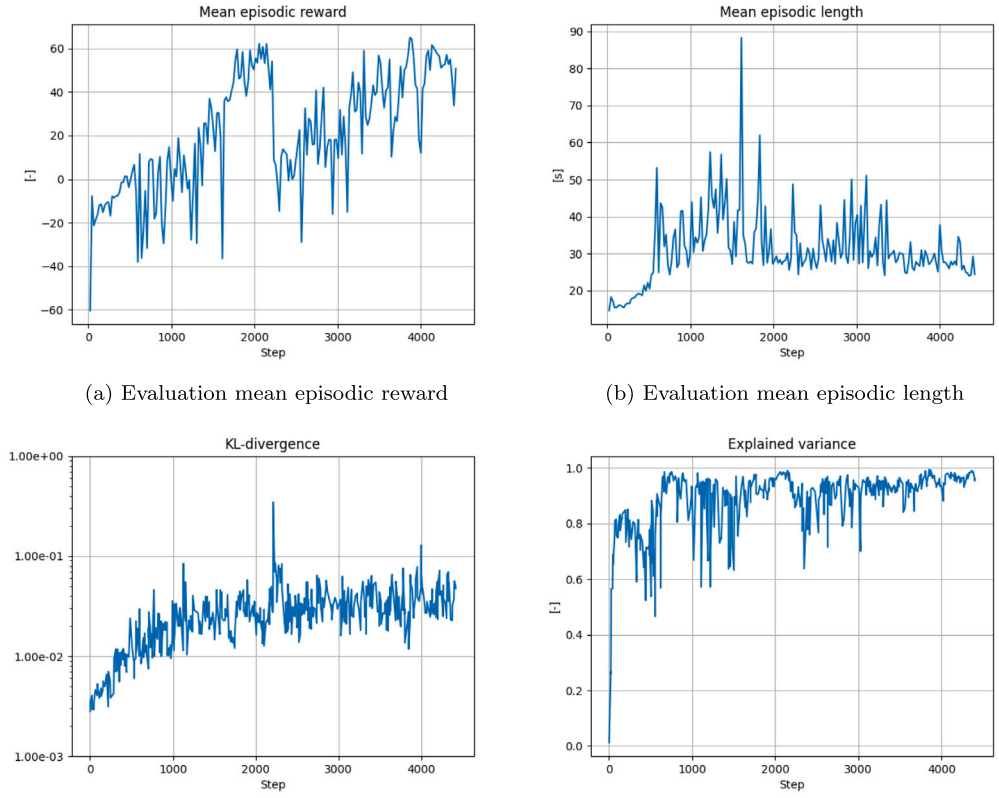
	RL controller (6-DOF)	3-DOF optimal solution	SCVX (6-DOF)
$\lvert m_0 - m_f \rvert$	4250 kg	3545 kg	7525 kg

6. Robustness to unmodeled dynamics and disturbances

This section analyzes the robustness of the control policy to various unmodeled dynamics and external disturbances. The key factors considered include:

1. **Error in the Position of the Center of Mass (CoM):**
   - **Variation in CoM Position:** The dry vehicle's CoM position varies within ±3% of its nominal position (15 m).
   - **Propellant Consumption Effect:** The shift in CoM due to propellant consumption is modeled, accounting for the time-varying masses of fuel and oxidizer through the mixture ratio (O/F).
   - **CoM Calculation:**

$$x_{\text{CG}} = \frac{m_{\text{dry}} x_{\text{CG}}^{\text{dry}} + m_{\text{ox}} x_{\text{CG}}^{\text{ox}} + m_{\text{fuel}} x_{\text{CG}}^{\text{fuel}}}{m_{\text{dry}} + m_{\text{ox}} + m_{\text{fuel}}} \tag{29}$$



(a) Evaluation mean episodic reward

(b) Evaluation mean episodic length

(c) Approximate KL-divergence

(d) Training explained variance

Fig. 13. Reinforcement learning training metrics. The upwards trend in episodic reward show successful learning of a good policy. The decrease in training loss also signals convergence, as well as explained variance showing that the agent has explored the environment.

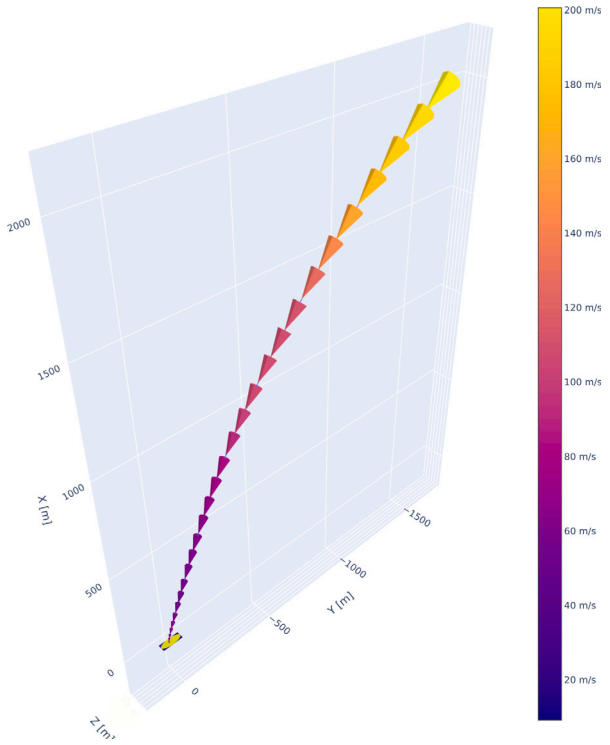


Fig. 14. Trajectory of an episode using the optimal network. We can notice the decrease in velocity and the successful pinpoint landing. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2. **Flexible Modes of the Structure:**

   - **Modeling Flexibility:** Introduces perturbing forces and torques to account for structural flexibility.
   - **Modal Dynamics:**

   $$\ddot{q}_i = -\omega_i^2 q_i - 2\xi\omega_i \dot{q}_i - T_i t_{p,i} \tag{30}$$

   where q_i are modal variables, ω_i are eigenfrequencies, ξ is the damping coefficient, and T_i represents torques.

3. **Uncertainty in the Inertia Moments:**

   - **Time-Varying Inertia:** Inertia moments are recalculated at each time step due to changing propellant mass.
   - **Inertia Uncertainty:** A $\pm 1\%$ uncertainty is introduced, sampled from a uniform distribution.

4. **Real Dynamics of the Actuators:**

   - **TVC Gimbal Actuators:** Modeled as second-order low-pass filters with natural frequency ω_{act} and damping coefficient ξ_{act}.

   $$\frac{\delta_i}{\delta_i^{\text{cmd}}}(s) = \frac{\omega_{\text{act}}^2}{s^2 + 2\xi_{\text{act}}\omega_{\text{act}} s + \omega_{\text{act}}^2} \tag{31}$$

   - **Thruster Delay:** Modeled as a first-order low-pass filter with a characteristic time $\tau_{\text{thrust}} = 2\,\text{s}$.

   $$\frac{\|\mathbf{T}\|}{\|\mathbf{T}_{\text{cmd}}\|}(s) = \frac{\omega_{\text{thrust}}}{\omega_{\text{thrust}} + s} \tag{32}$$

5. **Misalignment of the Thrust:**

   - **Thrust Vector Misalignment:** Small offsets ϵ_i (sampled uniformly from $[-0.5°, 0.5°]$) are applied using a rotation matrix R_ϵ to simulate misalignment.

6. **Wind Gusts and Wind Layers Model:**

   - **Wind Gusts:** Modeled as sinusoidal gusts with amplitude $\mathbf{A}_{\text{gust}} = [15, 15, 15]\,\text{m/s}$ over specific height ranges.

   $$\mathbf{V}_{\text{gust}} = \mathbf{A}_{\text{gust}} \left(1 - \cos\left(\frac{\pi(x - h_1)}{0.5\Delta h}\right)\right) \tag{33}$$

   - **Wind Layers:** Implemented using the Horizontal Wind Model 14 (HWM14) from the U.S. Naval Research Laboratory [27].

6.1. Sensitivity results

Excluding outliers, the controller showed robustness to every source of disturbances, maintaining low position and velocity errors comparable to disturbance-free scenarios as shown in Fig. 16.

Some simulation runs resulted in divergence (*outliers*) as evident in Fig. 15, characterized by the vehicle hovering or ascending near the landing pad instead of landing (see Table 15).

The control policy exhibits robustness against various unmodeled dynamics and disturbances. The low dispersion in landing positions and minimal final errors demonstrate the effectiveness of the controller under realistic operational conditions. Outlier cases suggest specific scenarios where control refinement may be necessary, but overall performance remains within acceptable limits.

Table 15
Number of outliers (runs where there is a divergence of the controller) out of 100 runs for each disturbance.

Disturbance	Outliers (out of 100 runs)
Center of mass error	0
Flexible modes	0
Inertia moments error	1
Offset CoM	1
Real actuator dynamics	2
Thrust misalignment	0
Wind gusts	0
Wind layers model	1

7. Conclusions

In this work, PPO is used to develop a control policy algorithm for the task of landing a reusable launcher's first stage. Starting from a simplified 3-DOF planar environment, several aspects were analyzed, from how hyperparameters selection affects convergence to how the reward and activation functions influence performance. Successively we moved to studying the 6-DOF problem and developing a policy to land successfully in this more challenging case. Dense reward functions were used in both cases, as they were found to provide a strong learning signal to the RL algorithm. To achieve this, first, a target-velocity tracking reward function was used. Different initial conditions were tested in both cases, starting on a simplified set and then moving to realistic conditions taken from the Falcon 9 flight profile. This change required modifying the reward function, moving from tracking a heuristic target velocity to tracking a target acceleration. Both reward functions had to be extensively tuned to achieve satisfactory performance. The batch size sampled from the replay buffer to train the policy network was found to be a critical hyperparameter in achieving good convergence in terms of landing errors (terminal velocity and position errors).

Both in 3DOF and 6DOF cases, the learned policy manages to achieve the landing objective with low fuel consumption, even if in the latter case the learning algorithm necessitates a much longer time to converge to a quasi-optimal policy.

The simplicity of the control policy algorithm is counter-weighted by the need for an exhaustive grasp of domain-specific knowledge when developing and tuning the reward function. Furthermore, the policy's *explainability* is low, due to the black-box nature of the neural net used to approximate it, this being an aspect that gains particular significance in the context of reentry and landing procedures, especially when considering Earth landings and potentially manned vehicles.

Nonetheless, Reinforcement Learning proves to be a valid option for learning a landing policy and opens up the door for integrating additional actuators and dynamics by simply modifying the environment, without any changes to the learning algorithm.

CRediT authorship contribution statement

Davide Iafrate: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Andrea Brandonisio:** Writing – review & editing, Methodology, Conceptualization. **Robert Hinz:** Investigation, Conceptualization. **Michèle Lavagna:** Writing – review & editing, Supervision, Investigation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Davide Iafrate reports financial support was provided by DEIMOS Engenharia, SA. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

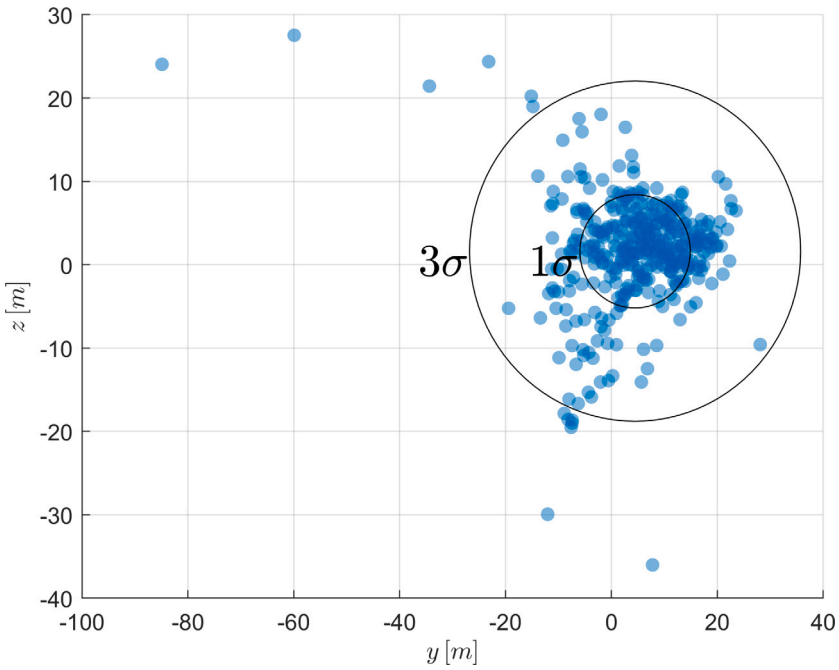
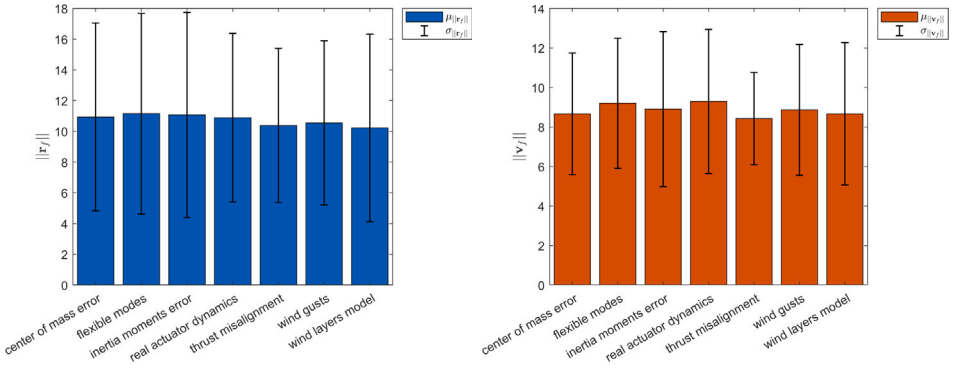


Fig. 15. Dispersion plot with disturbances and unmodeled dynamics. Most of the simulations result in a successful landing within a 20 m radius, however, there are a few significant outliers.



(a) Final position error (b) Final velocity error

Fig. 16. Sensitivity to unmodeled dynamics and disturbances. The policy proves to be robust by having low errors and dispersion in the presence of different disturbances and uncertainties.

Acknowledgments

The author wants to thank Deimos Space for supporting the development of the work, with a special thank you to the Madrid and Lisbon GNC Competence Center.

Appendix

Quaternion parametrization. The quaternion parametrization uses the scalar-first convention as in Eq. (34): the angle θ represents *double* the magnitude of the rotation around the rotation axis expressed by the vector part of the quaternion $[q_x, q_y, q_z]$.

$$\vec{q} = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} cos\frac{\theta}{2} \\ \vec{i}sin\frac{\theta}{2} \\ \vec{j}sin\frac{\theta}{2} \\ \vec{k}sin\frac{\theta}{2} \end{bmatrix} \tag{34}$$

The rotation matrix from the inertial to the body reference frame can be computed knowing the elements of the quaternion as:

$$R_{I\to B} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{35}$$

References

[1] A.R. Klumpp, Apollo lunar descent guidance, Automatica 10 (2) (1974) 133–146.
[2] C. D'Souza, C. D'Souza, An Optimal Guidance Law for Planetary Landing, AIAA, 1997, http://dx.doi.org/10.2514/6.1997-3709, arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.1997-3709 URL: https://arc.aiaa.org/doi/abs/10.2514/6.1997-3709.
[3] B. Acikmese, S.R. Ploen, Convex programming approach to powered descent guidance for mars landing, J. Guid. Control Dyn. 30 (5) (2007) 1353–1366, http://dx.doi.org/10.2514/1.27553, arXiv:https://doi.org/10.2514/1.27553.
[4] L. Blackmore, B. Açikmeşe, D.P. Scharf, Minimum-landing-error powered-descent guidance for mars landing using convex optimization, J. Guid. Control Dyn. 33 (4) (2010) 1161–1171, http://dx.doi.org/10.2514/1.47202, arXiv:https://doi.org/10.2514/1.47202.

[5] B.A. kmeşe, L. Blackmore, Lossless convexification of a class of optimal control problems with non-convex control constraints, Automatica 47 (2) (2011) 341–347, http://dx.doi.org/10.1016/j.automatica.2010.10.037, URL: https://www.sciencedirect.com/science/article/pii/S0005109810004516.

[6] B. Açı kmeşe, J. Casoliva, J. Carson, L. Blackmore, G-FOLD: A Real-Time Implementable Fuel Optimal Large Divert Guidance Algorithm for Planetary Pinpoint Landing, LPI Contributions, 2012, p. 4193.

[7] M. Szmuk, T.P. Reynolds, B. Açıkmeşe, Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints, J. Guid. Control Dyn. 43 (8) (2020) 1399–1413, http://dx.doi.org/10.2514/1.G004549, arXiv:https://doi.org/10.2514/1.G004549.

[8] J. Guadagnini, M. Lavagna, P. Rosa, Model predictive control for reusable space launcher guidance improvement, Acta Astronaut. 193 (2022) 767–778, http://dx.doi.org/10.1016/j.actaastro.2021.10.014, URL: https://www.sciencedirect.com/science/article/pii/S0094576521005567.

[9] S. Silvestrini, L.P. Cassinis, R. Hinz, D. Gonzalez-Arjona, M. Tipaldi, P. Visconti, F. Corradino, V. Pesce, A. Colagrossi, Chapter fifteen - modern spacecraft GNC, in: V. Pesce, A. Colagrossi, S. Silvestrini (Eds.), Modern Spacecraft Guidance, Navigation, and Control, Elsevier, 2023, pp. 819–981, http://dx.doi.org/10.1016/B978-0-323-90916-7.00015-9, URL: https://www.sciencedirect.com/science/article/pii/B9780323909167000159.

[10] C. Greatwood, A.G. Richards, Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control, Auton. Robots 43 (2019) 1681–1693.

[11] A.T. Azar, A. Koubaa, N. Ali Mohamed, H.A. Ibrahim, Z.F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A.M. Khamis, I.A. Hameed, G. Casalino, Drone deep reinforcement learning: A review, Electronics 10 (9) (2021) http://dx.doi.org/10.3390/electronics10090999, URL: https://www.mdpi.com/2079-9292/10/9/999.

[12] M. Yan, R. Yang, Y. Zhang, et al., A hierarchical reinforcement learning method for missile evasion and guidance, Sci. Rep. 12 (2022) 18888, http://dx.doi.org/10.1038/s41598-022-21756-6.

[13] N.B. LaFarge, D. Miller, K.C. Howell, R. Linares, Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment, Acta Astronaut. 186 (2021) 1–23, http://dx.doi.org/10.1016/j.actaastro.2021.05.014, URL: https://www.sciencedirect.com/science/article/pii/S0094576521002460.

[14] K. Hovell, S. Ulrich, On deep reinforcement learning for spacecraft guidance, 2020, http://dx.doi.org/10.2514/6.2020-1600, arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2020-1600 URL: https://arc.aiaa.org/doi/abs/10.2514/6.2020-1600.

[15] A. Brandonisio, M. Lavagna, D. Guzzetti, Reinforcement learning for uncooperative space objects smart imaging path-planning, J. Astronaut. Sci. 68 (4) (2021) 1145–1169, http://dx.doi.org/10.1007/s40295-021-00288-7.

[16] A. Brandonisio, L. Capra, M. Lavagna, Deep reinforcement learning spacecraft guidance with state uncertainty for autonomous shape reconstruction of uncooperative target, Adv. Space Res. (2023) http://dx.doi.org/10.1016/j.asr.2023.07.007.

[17] A. Scorsoglio, R. Furfaro, R. Linares, B. Gaudet, Image-based deep reinforcement learning for autonomous lunar landing, 2020, http://dx.doi.org/10.2514/6.2020-1910, arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2020-1910 URL: https://arc.aiaa.org/doi/abs/10.2514/6.2020-1910.

[18] A. Scorsoglio, A. D'Ambrosio, L. Ghilardi, B. Gaudet, F. Curti, R. Furfaro, Image-based deep reinforcement meta-learning for autonomous lunar landing, J. Spacecr. Rockets 59 (1) (2022) 153–165, http://dx.doi.org/10.2514/1.A35072, arXiv:https://doi.org/10.2514/1.A35072.

[19] B. Gaudet, R. Furfaro, Adaptive pinpoint and fuel efficient mars landing using reinforcement learning, IEEE/CAA J. Autom. Sin. 1 (4) (2014) 397–411, http://dx.doi.org/10.1109/JAS.2014.7004667.

[20] R. Furfaro, A. Scorsoglio, R. Linares, M. Massari, Adaptive generalized ZEM-ZEV feedback guidance for planetary landing via a deep reinforcement learning approach, Acta Astronaut. 171 (2020) 156–171, http://dx.doi.org/10.1016/j.actaastro.2020.02.051, URL: https://www.sciencedirect.com/science/article/pii/S0094576520301247.

[21] B. Gaudet, R. Linares, R. Furfaro, Deep reinforcement learning for six degree-of-freedom planetary powered descent and landing, 2018, arXiv e-prints arXiv:1810.08719.

[22] I.S..S. atmosphere, ISO 2533:1975 Standard Atmosphere, International Organization for Standardization, Geneva, Switzerland, 1975.

[23] SpaceX, Falcon 9 User Guide, Technical Report, SpaceX, 2020, URL: https://web.archive.org/web/20201202093334/https://www.spacex.com/media/falcon_users_guide_042020.pdf.

[24] D. Murphy, Flightclub, 2022, URL: https://flightclub.io/.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, http://dx.doi.org/10.48550/ARXIV.1707.06347, URL: https://arxiv.org/abs/1707.06347.

[26] Y. Guo, M. Hawkins, B. Wie, Waypoint-optimized zero-effort-miss/zero-effort-velocity feedback guidance for mars landing, J. Guid. Control Dyn. 36 (3) (2013) 799–809, http://dx.doi.org/10.2514/1.58098.

[27] D.P. Drob, et al., An update to the horizontal wind model (HWM): The quiet time thermosphere, Earth Space Sci. 2 (2015) 301–319, http://dx.doi.org/10.1002/2014EA000089.